# Elements of Sequential Monte Carlo

Christian A. Naesseth, Fredrik Lindsten and Thomas B. Schon

The self-archived postprint version of this journal article is available at Linköping University Institutional Repository (DiVA):
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-164491

N.B.: When citing this work, cite the original publication.
Naesseth, C. A., Lindsten, F., Schon, T. B., (2019), Elements of Sequential Monte Carlo, *FOUNDATIONS AND TRENDS IN MACHINE LEARNING*, 12(3), 187-306.
https://doi.org/10.1561/2200000074

Original publication available at:
https://doi.org/10.1561/2200000074

Copyright: Now Publishers
http://www.nowpublishers.com/

Tweet

LINKÖPING UNIVERSITY

arXiv:1903.04797v1 [stat.ML] 12 Mar 2019

# Elements of Sequential Monte Carlo

Christian A. Naesseth
Linköping University
christian.a.naesseth@liu.se

Fredrik Lindsten
Linköping University
fredrik.lindsten@liu.se

Thomas B. Schön
Uppsala University
thomas.schon@it.uu.se

March 13, 2019

**Abstract**

A core problem in statistics and probabilistic machine learning is to compute probability distributions and expectations. This is the fundamental problem of Bayesian statistics and machine learning, which frames all inference as expectations with respect to the posterior distribution. The key challenge is to approximate these intractable expectations. In this tutorial, we review sequential Monte Carlo (SMC), a random-sampling-based class of methods for approximate inference. First, we explain the basics of SMC, discuss practical issues, and review theoretical results. We then examine two of the main user design choices: the *proposal distributions* and the so called *intermediate target distributions*. We review recent results on how variational inference and amortization can be used to learn efficient proposals and target distributions. Next, we discuss the SMC estimate of the normalizing constant, how this can be used for pseudo-marginal inference and inference evaluation. Throughout the tutorial we illustrate the use of SMC on various models commonly used in machine learning, such as stochastic recurrent neural networks, probabilistic graphical models, and probabilistic programs.

# Contents


**1 Introduction** **4**
- 1.1 Historical Background . . . . . 4
- 1.2 Probabilistic Models and Target Distributions . . . . . 5
- 1.3 Applications . . . . . 10
- 1.4 Example Code . . . . . 13
- 1.5 Outline . . . . . 13

**2 Importance Sampling to Sequential Monte Carlo** **15**
- 2.1 Importance Sampling . . . . . 15
  - 2.1.1 Sequential Importance Sampling . . . . . 17
  - 2.1.2 Shortcomings of Importance Sampling . . . . . 20
- 2.2 Sequential Monte Carlo . . . . . 20
  - 2.2.1 Low-variance Resampling . . . . . 27
  - 2.2.2 Effective Sample Size and Adaptive Resampling . . . . . 29
- 2.3 Analysis and Convergence . . . . . 30

**3 Learning Proposals and Twisting Targets** **34**
- 3.1 Designing the Proposal Distribution . . . . . 34
  - 3.1.1 Locally Optimal Proposal Distribution . . . . . 34
  - 3.1.2 Approximations to the Optimal Proposal Distribution . . 35
  - 3.1.3 Learning a Proposal Distribution . . . . . 41
- 3.2 Adapting the Target Distribution . . . . . 47
  - 3.2.1 Twisting the Target . . . . . 47
  - 3.2.2 Designing the Twisting Potentials . . . . . 51
- 3.A Taylor and Unscented Transforms . . . . . 56

**4 Random but Proper Weights: Algorithms and Applications** **59**
- 4.1 The Unbiasedness of the Normalization Constant Estimate . . . . 59
- 4.2 Particle Metropolis-Hastings . . . . . 61
  - 4.2.1 Particle Marginal Metropolis-Hastings . . . . . 62
  - 4.2.2 Sampling the Latent Variables . . . . . 65
  - 4.2.3 Correlated Pseudo-Marginal Methods . . . . . 69
- 4.3 Proper Weights and Sequential Monte Carlo . . . . . 69
  - 4.3.1 Random Weights Sequential Monte Carlo . . . . . 71
  - 4.3.2 Nested Sequential Monte Carlo . . . . . 72
- 4.4 Distributed Sequential Monte Carlo . . . . . 76

4.4.1 Importance Weighted Independent SMC Samplers . . . . 76
4.4.2 The Island Particle Filter . . . . . 77
4.A Proof of Unbiasedness . . . . . 80

**5 Conditional SMC: Algorithms and Applications 83**
5.1 Conditional Sequential Monte Carlo . . . . . 83
5.2 The Unbiasedness of the Inverse Normalization Constant Estimate 86
5.3 The Expected SMC Empirical Distribution . . . . . 88
5.4 Parallelization . . . . . 88
5.5 Evaluating Inference . . . . . 90
5.A Proof of Unbiasedness . . . . . 95
5.B Proof of SMC Distribution Representation . . . . . 96
5.C Jeffreys Divergence Bound . . . . . 98

**6 Discussion 99**

# 1 Introduction

A key strategy in machine learning is to break down a problem into smaller and more manageable parts, then process data or unknown variables recursively. Well known examples of this are message passing algorithms for graphical models and annealing for optimization or sampling. Sequential Monte Carlo (SMC) is a class of methods that are tailored to solved statistical inference problems recursively. These methods have mostly received attention in the signal processing and statistics communities. With well over two decades of research in SMC, they have enabled inference in increasingly complex and challenging models. Recently, there has been an emergent interest in this class of algorithms from the machine learning community. We have seen applications to probabilistic graphical models (PGMs) [Naesseth et al., 2014, Paige and Wood, 2016], probabilistic programming [Wood et al., 2014], variational inference (VI) [Maddison et al., 2017, Naesseth et al., 2018, Le et al., 2018], inference evaluation [Grosse et al., 2015, Cusumano-Towner and Mansinghka, 2017], Bayesian nonparametrics [Fearnhead, 2004] and many other areas.

We provide a unifying view of the SMC methods that have been developed since their conception in the early 1990s [Gordon et al., 1993, Stewart and McCarty, 1992, Kitagawa, 1993]. In this introduction we provide relevant background material, introduce a running example, and discuss the use of code snippets throughout the tutorial.

## 1.1 Historical Background

SMC methods are generic tools for performing approximate (statistical) inference, predominantly Bayesian inference. They use a weighted sample set to iteratively approximate the posterior distribution of a probabilistic model. Ever since the dawn of Monte Carlo methods (see e.g. Metropolis and Ulam [1949] for an early discussion), random sample-based approximations have been recognized as powerful tools for inference in complex probabilistic models. Parallel to the development of Markov chain Monte Carlo (MCMC) methods [Metropolis et al., 1953, Hastings, 1970], sequential importance sampling (SIS) [Handschin and Mayne, 1969] and sampling/importance resampling [Rubin, 1987] laid the foundations for what would one day become SMC.

SMC methods where initially known as particle filters [Gordon et al., 1993, Stewart and McCarty, 1992, Kitagawa, 1993]. Particle filters where conceived as algorithms for online inference in nonlinear state space models (SSMs) [Cappé

et al., 2005]. Since then there has been a flurry of work applying SMC and particle filters to perform approximate inference in ever more complex models. While research in SMC initially focused on SSMs, we will see that SMC can be a powerful tool in a much broader setting.

## 1.2 Probabilistic Models and Target Distributions

As mentioned above, SMC methods were originally developed as an approximate solution to the so called filtering problem, which amounts to online inference in dynamical models. Several overview and tutorial articles focus on particle filters, i.e. the SMC algorithms specifically tailored to solve the online filtering problem [Arulampalam et al., 2002, Doucet and Johansen, 2009, Fearnhead and Künsch, 2018]. In this tutorial we will take a different view and explain how SMC can be used to solve more general "offline" problems. We shall see how this viewpoint opens up for many interesting applications of SMC in machine learning that do not fall in the traditional filtering setup, and furthermore how it gives rise to new and interesting design choices. We consider a generic probabilistic model given by a joint probability distribution function (PDF) of latent variables $\mathbf{x}$ and observed data $\mathbf{y}$,

$$p(\mathbf{x}, \mathbf{y}). \tag{1}$$

We focus on Bayesian inference, where the key object is the posterior distribution

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})}, \tag{2}$$

where $p(\mathbf{y})$ is known as the marginal likelihood.

The target distributions are a sequence of probability distributions that we recursively approximate using SMC. We define each target distribution $\gamma_t(x_{1:t})$ in the sequence as a joint PDF of latent variables $x_{1:t} = (x_1, \ldots, x_t)$, where $t = 1, \ldots, T$. The PDF is denoted by

$$\gamma_t(x_{1:t}) := \frac{1}{Z_t} \widetilde{\gamma}_t(x_{1:t}), \quad t = 1, \ldots, T, \tag{3}$$

where $\widetilde{\gamma}_t$ is a positive integrable function and $Z_t$ is the normalization constant, ensuring that $\gamma_t$ is indeed a PDF.

We connect the target distributions to the probabilistic model through a requirement on the final target distribution $\gamma_T(x_{1:T})$. We enforce the condition that $\gamma_T(x_{1:T})$ is either equivalent to the posterior distribution, or that it contains the posterior distribution as a marginal distribution. The *intermediate* target distributions, i.e. $\gamma_t(x_{1:t})$ for $t < T$, are useful only insofar they help us approximate the final target $\gamma_T(x_{1:T})$. This approach is distinct from previous tutorials on particle filters and SMC that traditionally focus on the intermediate targets, i.e. the filtering distributions. We stress that it is not necessarily the case that $x_{1:T}$ (the latent variables of the target distribution) is equal to $\mathbf{x}$ (the latent variables of the probabilistic model of interest), as the former can include additional auxiliary variables.

Below we introduce a few examples of probabilistic models and some straightforward choices of target distributions. We introduce and illustrate our running example which will be used throughout. We will return to the issue of choosing the sequence of intermediate targets in Section 3.2.

**State Space Models** The state space model (or hidden Markov model) is a type of probabilistic models where the latent variables and data satisfy a Markov property. For this model we typically have $\mathbf{x} = x_{1:T}$. Often the measured data can also be split into a sequence of the same length ($T$) as the latent variables, i.e. $\mathbf{y} = y_{1:T}$. The model is defined by a transition PDF $f$ and an observation PDF $g$,

$$x_t \mid x_{t-1} \sim f(\cdot \mid x_{t-1}), \tag{4a}$$

$$y_t \mid x_t \sim g(\cdot \mid x_t). \tag{4b}$$

The joint PDF is

$$p(\mathbf{x}, \mathbf{y}) = p(x_1)g(y_1 \mid x_1) \prod_{t=2}^{T} f(x_t \mid x_{t-1})g(y_t \mid x_t), \tag{5}$$

where $p(x_1)$ is the prior on the initial state $x_1$. This class of models is especially common for data that has an inherent temporal structure such as in the field of signal processing. A common choice is to let the target distributions follow the same sequential structure as in Eq. (5):

$$\widetilde{\gamma}_t(x_{1:t}) = p(x_1)g(y_1 \mid x_1) \prod_{k=2}^{t} f(x_k \mid x_{k-1})g(y_k \mid x_k), \tag{6}$$

which means that the final normalized target distribution satisfies $\gamma_T(x_{1:T}) = p(\mathbf{x}|\mathbf{y})$ as required. This is the model class and target distributions which are studied in the classical filtering setup.

**Non-Markovian Latent Variable Models** The non-Markovian latent variable models (LVMs) are characterized by either no, or higher order, Markov structure between the latent variables $\mathbf{x}$ and/or data $\mathbf{y}$. This can be seen as a non-trivial extension of the SSM, see Eq. (4), which has a Markov structure. Also for this class of models it is common to have $\mathbf{x} = x_{1:T}$ and $\mathbf{y} = y_{1:T}$.

Unlike the SSM, the non-Markovian LVM in its most general setting requires access to all previous latent variables $x_{1:t-1}$ to generate $x_t, y_t$

$$x_t \,|\, x_{1:t-1} \sim f_t(\cdot \,|\, x_{1:t-1}), \tag{7a}$$
$$y_t \,|\, x_{1:t} \sim g_t(\cdot \,|\, x_{1:t}), \tag{7b}$$

where we again refer to $f_t$ and $g_t$ as the transition PDF and observation PDF, respectively. The joint PDF is given by

$$p(\mathbf{x}, \mathbf{y}) = p(x_1)g(y_1 \,|\, x_1)\prod_{t=2}^{T} f_t(x_t \,|\, x_{1:t-1})g_t(y_t \,|\, x_{1:t}), \tag{8}$$

where $p(x_1)$ is again the prior on $x_1$. A typical target distribution is given by

$$\widetilde{\gamma}_t(x_{1:t}) = \widetilde{\gamma}_{t-1}(x_{1:t-1})f_t(x_t \,|\, x_{1:t-1})g_t(y_t \,|\, x_{1:t}), \quad t > 1, \tag{9}$$

with $\widetilde{\gamma}_1(x_1) = p(x_1)g_1(y_1 \,|\, x_1)$. Another option is

$$\begin{aligned}
\widetilde{\gamma}_1(x_1) &= p(x_1), \\
\widetilde{\gamma}_t(x_{1:t}) &= \widetilde{\gamma}_{t-1}(x_{1:t-1})f_t(x_t \,|\, x_{1:t-1}), \quad 1 < t < T, \\
\widetilde{\gamma}_T(x_{1:T}) &= \widetilde{\gamma}_{T-1}(x_{1:T-1})f_T(x_T \,|\, x_{1:T-1})\prod_{t=1}^{T} g_t(y_t \,|\, x_{1:t}).
\end{aligned}$$

For both these sequences of target distributions the final iteration $T$ is the posterior distribution, i.e. $\gamma_T(x_{1:T}) = p(x_{1:T} \,|\, y_{1:T}) = p(\mathbf{x}|\mathbf{y})$. However, the former one will often lead to more accurate inferences. This is because we introduce information from the data at an earlier stage in the SMC algorithm.

Throughout the monograph we will exemplify the different methods using a Gaussian special case of Eq. (7), see Example 1. We let the prior on $x_{1:t}$, defined by the transition PDF $f_t$, be Markovian and introduce the non-Markov property instead through the observation PDF $g_t$.

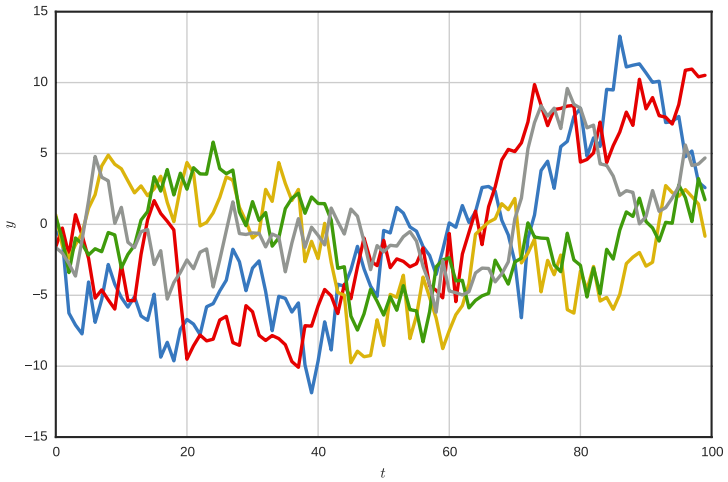



**Figure 1:** Five sample paths of $y_{1:T}$ from our running example for $T = 100$.

**Example 1** (Non-Markovian Gaussian Sequence Model)**.** *As our running example for illustration purposes we use a non-Markovian Gaussian sequence model. It is*

$$x_t \,|\, x_{1:t-1} \sim f_t(\cdot \,|\, x_{t-1}), \qquad\qquad y_t \,|\, x_{1:t} \sim g_t(\cdot \,|\, x_{1:t}), \tag{10}$$

*with observed variables $y_t$ (data), and where*

$$f_t(x_t \,|\, x_{t-1}) = \mathcal{N}(x_t \,|\, \phi\, x_{t-1}, q),$$
$$g_t(y_t \,|\, x_{1:t}) = \mathcal{N}\left(y_t \,|\, \sum_{k=1}^{t} \beta^{t-k} x_k, r\right).$$

*We let the prior at $t = 1$ be $p(x_1) = \mathcal{N}(x_1 \,|\, 0, q)$. Artificial data was generated using $(\phi, q, \beta, r) = (0.9, 1, 0.5, 1)$. The distribution of interest is the posterior distribution $p(x_{1:T} \,|\, y_{1:T})$. We illustrate a few sample paths of $y_{1:T}$ in Fig. 1 for $T = 100$.*

*We can adjust the strength of the dependence on previous latent variables in the observations, $y_t$, through the parameter $\beta \in [0, 1]$. If we set $\beta = 0$ we obtain a linear Gaussian* SSM*, since the data depends only on the most recent latent $x_t$. On the other hand if we let $\beta = 1$, this signifies that $x_k$ for $k < t$ has equally strong effect on $y_t$ as does $x_t$.*

Another example of a non-Markovian LVM often encountered in machine learning is the stochastic recurrent neural network (RNN). We define the stochastic RNN below and will return to it again in Section 3.

**Example 2** (Stochastic Recurrent Neural Network)**.** *A stochastic* RNN *is a non-Markovian* LVM *where the parameters of the transition and observation models are defined by* RNN*s. A common example is using the conditional Gaussian distribution to define the transition* PDF

$$f_t(x_t \mid x_{1:t-1}) = \mathcal{N}(x_t \mid \mu_t(x_{1:t-1}), \Sigma_t(x_{1:t-1})),$$

*where the functions* $\mu_t(\cdot), \Sigma_t(\cdot)$ *are defined by* RNN*s.*

**Conditionally independent models** A common model in probabilistic machine learning is to assume that the datapoints $y_k$ in the dataset $\mathbf{y} = \{y_k\}_{k=1}^K$ are conditionally independent given the latent $\mathbf{x}$. This means that the joint PDF is given by

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}) \underbrace{\prod_{k=1}^{K} g_k(y_k \mid \mathbf{x})}_{p(\mathbf{y} \mid \mathbf{x})}, \tag{11}$$

where $p(\mathbf{y} \mid \mathbf{x})$ is the likelihood. For this class of models it might not be immediately apparent that we can define a useful sequence of target distributions. However, as we shall see, we can make use of auxiliary variables to design target distributions that may help with inference.

We will discuss two approaches to design the sequence of target distributions: using data tempering and likelihood tempering, respectively. Both of these will make use of an auxiliary variable technique, where each $x_t$ is a random variable on the same space as $\mathbf{x}$.

*Data tempering:* Using data tempering we add the data $y_k$ to the target distribution one by one. In this case the data index $k$ coincides with the target index $t$. We define the target distribution

$$\widetilde{\gamma}_t(x_{1:t}) = p(x_t) \prod_{k=1}^{t} g_k(y_k \mid x_t) \cdot \prod_{k=1}^{t-1} s_k(x_k \mid x_{k+1}), \tag{12}$$

where the distributions $s_k(x_k \mid x_{k+1})$ are a design choice, known as backward kernels. With this choice, we have that the marginal distribution of $x_T$ at the final iteration is exactly the posterior distribution, i.e. $\gamma_T(x_T) = p(\mathbf{x} \mid \mathbf{y})$. In fact, at each step we have that the target distribution is a partial posterior $\gamma_t(x_t) = p(\mathbf{x} \mid y_{1:t})$.

*Likelihood tempering:* With likelihood tempering, instead of adding the data one by one, we change the likelihood $p(\mathbf{y} \mid \mathbf{x})$ through a sequence of positive variables. We define the target distribution

$$\widetilde{\gamma}_t(x_{1:t}) = p(x_t)p(\mathbf{y} \mid x_t)^{\tau_t} \cdot \prod_{k=1}^{t-1} s_k(x_k \mid x_{k+1}), \tag{13}$$

where $0 = \tau_1 < \ldots < \tau_T = 1$, and again make use of the user chosen backward kernels $s_k(x_k \mid x_{k+1})$. In this setting all data is considered at each iteration. Since $\tau_T = 1$, we have that the final marginal target distribution is again equal to the posterior $\gamma_T(x_T) = p(\mathbf{x} \mid \mathbf{y})$.

Applying SMC methods to tempered (and similar) target distributions has been studied by e.g. Chopin [2002], Del Moral et al. [2006]. We refer to these works for a thorough discussion on the choice of backward kernels $s_k(x_k \mid x_{k+1})$. Another well known example is annealed importance sampling by Neal [2001].

**Models and Targets** We have seen several probabilistic models with examples of corresponding target distributions. While not limited to these, this illustrates the wide range of the applicability of SMC. In fact, as long as we can design a sequence of target distributions such that $\gamma_T$ coincides with the distribution of interest, we can leverage SMC for inference.

## 1.3 Applications

Sequential Monte Carlo and importance sampling methods have already seen a plethora of applications to machine learning and statistical inference problems. Before we turn to the fundamentals of the various algorithms it can be helpful to understand some of these applications. We present and discuss a few select examples of applications of SMC and importance sampling (IS) to probabilistic graphical models, Bayesian nonparametric models, probabilistic programming, and inference evaluation.

**Probabilistic Graphical Models** Probabilistic graphical models (PGM; see e.g. Koller et al. [2009], Wainwright et al. [2008]) are probabilistic models where the conditional independencies in the joint PDF are described by edges in a graph. The graph structure allows for easy and strong control on the type of prior information that the user can express. The main limitation of the PGM

is that exact inference is often intractable and approximate inference is challenging.

The PGM is a probabilistic model where the PDF factorizes according to an underlying graph described by a set of cliques $C \in \mathscr{C}$, i.e. fully connected subsets of the vertices $V \in \mathscr{V}$ where $\mathscr{V}$ contains all individual components of $\mathbf{x}$ and $\mathbf{y}$. The undirected graphical model can be denoted by

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_{C \in \mathscr{C}} \psi_C(x_C), \tag{14}$$

where $Z$ is a normalization constant ensuring that the right hand side is a proper PDF.

SMC methods have recently been successfully applied to the task of inference in general PGMs, see e.g. Naesseth et al. [2014], Paige and Wood [2016], Lindsten et al. [2017, 2018] for representative examples.

**Probabilistic Programming** Probabilistic programming languages are programming languages designed to describe probabilistic models and to automate the process of doing inference in those models. We can think of probabilistic programming as that of automating statistical inference, particularly Bayesian inference, using tools from computer science and computational statistics. Developing a syntax and semantics to describe and denote probabilistic models and the inference problems we are interested in solving is key to the probabilistic programming language. To define what separates a probabilistic programming language from a standard programming language we quote Gordon et al. [2014]: "Probabilistic programs are usual functional or imperative programs with two added constructs: (1) the ability to draw values at random from distributions, and (2) the ability to condition values of variables in a program via observations." This aligns very well with the notion of Bayesian inference through the posterior distribution Eq. (2); through the syntax of the language we define $\mathbf{x}$ to be the random values we sample and $\mathbf{y}$ our observations that we condition on through the use of Bayes rule. The probabilistic program then essentially defines our joint probabilistic model $p(\mathbf{x}, \mathbf{y})$.

One of the main challenges of probabilistic programming is to develop algorithms that are general enough to enable inference for any model (probabilistic program) that we could conceivably write down using the language. Recently Wood et al. [2014] have shown that SMC-based approaches can be used as inference back-ends in probabilistic programs.

For a more thorough treatment of probabilistic programming we refer the interested reader to the recent tutorial by van de Meent et al. [2018] and the survey by Gordon et al. [2014].

**Bayesian nonparametric models** Nonparametric models are characterized by having a complexity which grows with the amount of available data. In a Bayesian context this implies that the usual latent random variables (i.e., parameters) of the model are replaced by latent stochastic processes. Examples include Gaussian processes, Dirichlet processes, and Beta processes; see e.g. Hjort et al. [2010] for a general introduction.

Sampling from these latent stochastic processes, conditionally on observed data, can be done using SMC. To give a concrete example, consider the Dirichlet process mixture model, which is a clustering model that can handle an unknown and conceptually infinite number of mixture components. Let $y_t$, $t = 1, 2, \ldots$ be a stream of data. Let $x_t$, $t = 1, 2, \ldots$ be a sequence of latent integer-valued random variables, such that $x_t$ is the index of the mixture component to which datapoint $y_t$ belongs. A generative representation of the mixture assignment variables is given by

$$p(x_{t+1} = j \mid x_{1:t}) = \begin{cases} \frac{n_{t,j}}{t+\alpha} & \text{for } j = 1, \ldots, J_t, \\ \frac{\alpha}{t+\alpha} & \text{for } j = J_t + 1, \end{cases}$$

where $J_t := \max\{x_{1:t}\}$ is the number of distinct mixture components represented by the first $t$ datapoints, and $n_{t,j} := \sum_{k=1}^{t} \mathbb{I}\{x_k = j\}$ is the number of datapoints among $y_{1:t}$ that belong to the $j$th component.

The model is completed by specifying the distribution of the data, conditionally on the mixture assignment variables:

$$\begin{aligned} \theta_k &\sim F(\theta), \quad k = 1, 2, \ldots \\ p(y_t \mid x_t, \{\theta_k\}_{k \geq 1}) &= G(y_t \mid \theta_{x_t}), \end{aligned}$$

where $G(\,\cdot \mid \theta)$ is an emission probability distribution parameterized by $\theta$ and $F(\theta)$ is a prior distribution over the mixture parameters $\theta$.

Note that the mixture assignment variables $x_t$, $t = 1, 2, \ldots$ evolve according to a latent stochastic process. Solving the clustering problem amounts to computing the posterior distribution of this stochastic process, conditionally on the observed data. One way to address this problem is to use SMC; see Fearnhead [2004] for an efficient implementation tailored to the discrete nature of the problem.

**Inference Evaluation** An important problem when performing approximate Bayesian inference is to figure out when our approximation is âĂIJgood enoughâĂİ? Is it possible to give practical guarantees on the approximation we obtain? We need ways to evaluate how accurate our approximate inference algorithms are when compared to the true target distribution that we are trying to approximate. We will refer to the process of evaluating and validating approximate inference methods as *inference evaluation*.

Inference evaluation is mainly concerned with measuring how close our approximation is to the true object we are trying to estimate, often a posterior distribution. For simulated data, Grosse et al. [2015, 2016] have shown that we can make use of SMC and IS to bound the symmetrized Kullback-Leibler (KL) divergence between our approximate posterior and the true posterior. In another related work Cusumano-Towner and Mansinghka [2017] have shown that SMC-based methods show promise in estimating the symmetric KL divergence between the approximate posterior and a gold standard algorithm.

## 1.4 Example Code

We will be making use of inline Python code snippets throughout the manuscript to illustrate the algorithms and methods. Below we summarize the modules that are necessary to import to run the code snippets:

```
import numpy as np
import numpy.random as npr
from scipy.misc import logsumexp
from scipy.stats import norm
```

**Example Code 1:** Necessary imports for Python code examples.

## 1.5 Outline

The remainder of this tutorial is organized as follows. In Section 2, we first introduce IS, a foundational building block for SMC. Then, we discuss the limitations of IS and how SMC resolves these. Finally, the section concludes with discussing some practical issues and theoretical results relevant to SMC methods.

Section 3 is focused on the two key design choices of SMC: the *proposal* and *target* distributions. Initially we focus on the proposal, discussing various ways

of adapting and learning good proposals that will make the approximation more accurate. Then we discuss the sequence of target distributions; how we can learn intermediate distributions that help us when we try to approximate the posterior.

Section 4 focuses on pseudo marginal (PM) methods and other SMC methods that rely on a concept known as *proper weights*. First, we provide a simple and straightforward proof of the unbiasedness property of the SMC normalization constant estimate. Then, we describe and illustrate the combination of MCMC and SMC methods through PM algorithms. We move on to detail properly weighted SMC, a concept that unites and extends the random weights and nested SMC algorithms. Finally, we conclude the chapter by considering a few approaches for distributed and parallel SMC.

In Section 5 we introduce conditional sequential Monte Carlo (CSMC), and related methods for simulating from and computing expectations with respect to a target distribution. First, we introduce the basic CSMC algorithm and provide a straightforward proof of the unbiasedness of the inverse normalization constant estimate. Then, we show how SMC and CSMC can be combined to leverage multi-core and parallel computing architectures in the interacting particle Markov chain Monte Carlo (IPMCMC) algorithm. Finally, we discuss recent applications of CSMC for evaluating approximate inference methods.

The tutorial concludes with a discussion and outlook in Section 6.

# 2 Importance Sampling to Sequential Monte Carlo

Typical applications require us to be able to evaluate or sample from the target distributions $\gamma_t$, as well as compute their normalization constants $Z_t$. For most models and targets this will be intractable, and we need approximations based on e.g. Monte Carlo methods.

In this section, we first review IS and some of its shortcomings. Then, we introduce the SMC method, the key algorithm underpinning this monograph. Finally, we discuss some key theoretical properties of the SMC algorithm.

## 2.1 Importance Sampling

Importance sampling is a Monte Carlo method that constructs an approximation using samples from a proposal distribution, and corrects for the discrepancy between the target and proposal using (importance) weights.

Most applications of Bayesian inference can be formulated as computing the expectation of some generic function $h_t$, referred to as a test function, with respect to the target distribution $\gamma_t$,

$$\gamma_t(h_t) := \mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right]. \tag{15}$$

Examples include posterior predictive distributions, Bayesian p-values, and point estimates such as the posterior mean. Computing Eq. (15) is often intractable, but by a clever trick we can rewrite it as

$$\mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right] = \frac{1}{Z_t}\mathbb{E}_{q_t}\left[\frac{\widetilde{\gamma}_t(x_{1:t})}{q_t(x_{1:t})}h_t(x_{1:t})\right] = \frac{\mathbb{E}_{q_t}\left[\frac{\widetilde{\gamma}_t(x_{1:t})}{q_t(x_{1:t})}h_t(x_{1:t})\right]}{\mathbb{E}_{q_t}\left[\frac{\widetilde{\gamma}_t(x_{1:t})}{q_t(x_{1:t})}\right]}. \tag{16}$$

The PDF $q_t$ is a user chosen proposal distribution, we assume it is simple to sample from and evaluate. We can now estimate the right hand side of Eq. (16) using the Monte Carlo method,

$$\mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right] \approx \frac{\frac{1}{N}\sum_{i=1}^{N}\widetilde{w}_t(x_{1:t}^i)h_t(x_{1:t}^i)}{\frac{1}{N}\sum_{j=1}^{N}\widetilde{w}_t(x_{1:t}^j)}, \tag{17}$$

where $\widetilde{w}_t(x_{1:t}) := {}^{\widetilde{\gamma}_t(x_{1:t})}/_{q_t(x_{1:t})}$ and $x_{1:t}^i$ are simulated iid from $q_t$. We will usually write Eq. (17) more compactly as

$$\mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right] \approx \sum_{i=1}^{N} w_t^i h_t(x_{1:t}^i), \quad x_{1:t}^i \overset{\text{iid}}{\sim} q_t, \tag{18}$$

where the normalized weights $w_t^i$ are defined by

$$w_t^i := \frac{\widetilde{w}_t^i}{\sum_j \widetilde{w}_t^j},$$

with $\widetilde{w}_t^i$ a shorthand for $\widetilde{w}_t(x_{1:t}^i)$. The estimate in Eq. (18) is strongly consistent, converging (almost surely) to the true expectation as the number of samples $N$ tend to infinity. An alternate view of IS is to consider it an (empirical) approximation[1] of $\gamma_t$,

$$\gamma_t(x_{1:t}) \approx \sum_{i=1}^{N} w_t^i \delta_{x_{1:t}^i}(x_{1:t}) =: \widehat{\gamma}_t(x_{1:t}), \tag{19}$$

where $\delta_X$ denotes the Dirac measure at $X$. Furthermore, IS provides an approximation of the normalization constant,

$$Z_t \approx \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_t^i =: \widehat{Z}_t \tag{20}$$

Because the weights depend on the random samples, $x_{1:t}^i$, they are themselves random variables. One of the key properties is that it is *unbiased*, This can be easily seen by noting that $x_{1:t}^i$ are iid draws from $q_t$ and therefore

$$\mathbb{E}[\widehat{Z}_t] = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}\left[\frac{\widetilde{\gamma}_t(x_{1:t}^i)}{q_t(x_{1:t}^i)}\right] = \frac{1}{N} \sum_{i=1}^{N} \int \frac{\widetilde{\gamma}_t(x_{1:t}^i)}{q_t(x_{1:t}^i)} q_t(x_{1:t}^i)\, \mathrm{d}x_{1:t}^i = Z_t, \tag{21}$$

since $Z_t = \int \widetilde{\gamma}_t(x_{1:t})\, \mathrm{d}x_{1:t}$ is nothing by the normalization constant for $\widetilde{\gamma}_t$. This property will be important for several of the more powerful IS and SMC-based methods considered in this monograph.

We summarize the importance sampling method in Algorithm 1. This algorithm is sometimes referred to as *self-normalized* IS, because we are normalizing each individual weight using all samples.

The straightforward implementation of the IS method we have described thus far is impractical for many of the example models and targets in Section 1.2. It is challenging to design good proposals for high-dimensional models. A good proposal is typically more heavy-tailed than the target; if it is not,

[1] This should be interpreted as an approximation of the underlying probability distribution, and not of the density function itself.

**Algorithm 1:** Importance sampling (IS)

**input :** Unnormalized target distribution $\widetilde{\gamma}_t$, proposal $q_t$, number of samples $N$.

**output:** Samples and weights $\left\{\left(x_{1:t}^i, w_t^i\right)\right\}_{i=1}^N$ approximating $\gamma_t$.

**for** $i = 1$ **to** $N$ **do**

  Sample $x_{1:t}^i \sim q_t$

  Set $\widetilde{w}_t^i = \frac{\widetilde{\gamma}_t(x_{1:t}^i)}{q_t(x_{1:t}^i)}$

**end**

Set $w_t^i = \frac{\widetilde{w}_t^i}{\sum_j \widetilde{w}_t^j}$, for $i = 1, \ldots, N$

the weights can have infinite variance. Another favorable property of a proposal is that it should cover the bulk of the target probability mass, putting high probability on regions of high probability under the target distribution. Even Markovian models, such as the SSM, can have a prohibitive computational complexity without careful design of the proposal. In the next section we will describe how we can alleviate these concerns using SIS, a special case of IS, with a kind of divide-and-conquer approach to tackle the high-dimensionality in $T$.

### 2.1.1 Sequential Importance Sampling

Sequential importance sampling is a variant of IS were we select a proposal distribution that has an autoregressive structure, and compute importance weights recursively. By choosing a proposal defined by

$$q_t(x_{1:t}) = q_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})$$

we can decompose the proposal design problem into $T$ conditional distributions. This means we obtain samples $x_{1:t}^i$ by reusing $x_{1:t-1}^i$ from the previous iteration, and append a new sample, $x_t^i$, simulated from $q_t(x_t \mid x_{1:t-1}^i)$. The unnormalized weights can be computed recursively by noting that

$$\begin{aligned}\widetilde{w}_t(x_{1:t}) &= \frac{\widetilde{\gamma}_t(x_{1:t})}{q_t(x_{1:t})} = \frac{\widetilde{\gamma}_{t-1}(x_{1:t-1})}{q_{t-1}(x_{1:t-1})} \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})} \\ &= \widetilde{w}_{t-1}(x_{1:t-1}) \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})}.\end{aligned}$$

**Algorithm 2:** Sequential importance sampling (SIS)

**input :** Unnormalized target distributions $\widetilde{\gamma}_t$, proposals $q_t$, number of samples $N$.

**output:** Samples and weights $\left\{\left(x_{1:t}^i, w_t^i\right)\right\}_{i=1}^N$ approximating $\gamma_t$, for $t = 1, \ldots, T$.

**for** $t = 1$ **to** $T$ **do**
  **for** $i = 1$ **to** $N$ **do**
    Sample $x_t^i \sim q_t(x_t \mid x_{1:t-1}^i)$
    Append $x_{1:t}^i = \left(x_{1:t-1}^i, x_t^i\right)$
    Set $\widetilde{w}_t^i = \widetilde{w}_{t-1}^i \frac{\widetilde{\gamma}_t(x_{1:t}^i)}{\widetilde{\gamma}_{t-1}(x_{1:t-1}^i) q_t(x_t^i \mid x_{1:t-1}^i)}$
  **end**
  Set $w_t^i = \frac{\widetilde{w}_t^i}{\sum_j \widetilde{w}_t^j}$, for $i = 1, \ldots, N$
**end**

We summarize the SIS method in Algorithm 2, where $q_1(x_1 \mid x_{1:0}) = q_1(x_1)$ and $\widetilde{w}_0 = \widetilde{\gamma}_0 = 1$.

If we need to evaluate the normalization constant estimate $\widehat{Z}_t$, analogously to IS we make use of Eq. (20). However, we may also obtain a (strongly) consistent estimate of the ratio of normalization constants $Z_t / Z_{t-1}$

$$\begin{aligned}\frac{Z_t}{Z_{t-1}} &= \mathbb{E}_{\gamma_t(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})}\left[\frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})}\right] \\ &\approx \sum_{i=1}^N w_{t-1}^i \frac{\widetilde{w}_t^i}{\widetilde{w}_{t-1}^i}.\end{aligned}$$

While the estimate of the *ratio* is consistent, it is in general not unbiased. However, SIS is a special case of IS. This means that the SIS estimate of the normalization constant for $\widetilde{\gamma}_t$, i.e. $\widehat{Z}_t$ in Eq. (20), is still unbiased and consistent.

In Example 3 we detail a first example proposal $q_t$ for the running example, and derive the corresponding weights $\widetilde{w}_t$. Furthermore, we include a code snippet that illustrates how to implement the sampler in Python.

**Example 3** (Sequential importance sampling for Example 1)**.** *We revisit our*

*running non-Markovian Gaussian example. The target distribution is*

$$\widetilde{\gamma}_t(x_{1:t}) = p(x_1)g(y_1 \mid x_1)\prod_{k=2}^{t} f(x_k \mid x_{k-1})g(y_k \mid x_{1:k}),$$

*with* $p(x_1) = \mathcal{N}(x_1 \mid 0, q)$ *and*

$$f(x_k \mid x_{k-1}) = \mathcal{N}(x_k \mid \phi x_{k-1}, q), \quad g(y_k \mid x_{1:k}) = \mathcal{N}\left(y_t \mid \sum_{l=1}^{k} \beta^{k-l} x_l, r\right).$$

*A common approach is to set the proposal to be the prior (or transition) distribution* $f$*. A sample from the proposal* $q_t(x_t \mid x_{t-1}) = f(x_t \mid x_{t-1})$ *is generated as follows*

$$x_t = \phi x_{t-1} + \sqrt{q}\varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, 1). \tag{22}$$

*We refer to this proposal simply as the* prior *proposal. The corresponding weight update is*

$$\begin{aligned}\widetilde{w}_t(x_{1:t}) &= \widetilde{w}_{t-1}(x_{1:t-1})\frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_{t-1}(x_{1:t-1})q_t(x_t \mid x_{1:t-1})} \\ &= \widetilde{w}_{t-1}(x_{1:t-1})\mathcal{N}\left(y_t \mid \sum_{k=1}^{t} \beta^{t-k} x_k, r\right),\end{aligned} \tag{23}$$

*where* $\widetilde{w}_0 = 1$*. We provide Example Code 2 to illustrate how to implement* SIS *with the prior proposal for this model in Python.*

```
x = np.zeros((N,T))
logw = np.zeros(N)
mu = np.zeros(N)
for t in range(T):
    x[:,t]= phi*x[:,t-1]+np.sqrt(q)*npr.randn(N)
    mu = beta*mu + x[:,t]
    logw += norm.logpdf(y[t], mu, np.sqrt(r))
w = np.exp(logw-logsumexp(logw))
```

**Example Code 2:** Sequential importance sampling for Example 1.

*For improved numerical stability we update the log-weights* $\log \widetilde{w}_t$ *and subtract the logarithm of the sum of weights (the weight normalization) before exponentiating.*

SIS can be implemented efficiently for a large class of problems, the computational complexity is usually linear in $N$ and $T$. Even so, the IS methods suffer from severe drawbacks limiting their practical use for many high-dimensional problems.

### 2.1.2 Shortcomings of Importance Sampling

The main drawback of IS is that the variance of the estimator scales unfavorably with the dimension of the problem; the variance generally increases exponentially in $T$. Because SIS is a special case of IS it inherits this unfavorable property.

One way in which this problem manifests itself in practice is through the normalized weights $w_t^i$. The maximum of the weights, $\max_i w_t^i$, will quickly approach one as $t$ increases, and consequently all other weights approach zero; a phenomena known as *weight degeneracy*. This means that, effectively, we approximate the target distribution using a single sample.

We illustrate weight degeneracy in Example 4 using the running example.

**Example 4** (SIS weight degeracy)**.** *We return to our running example, set the length $T = 6$, number of particles $N = 5$, and $(\phi, q, \beta, r) = (0.9, 1.0, 0.5, 1.0)$. Fig. 2 shows the particles and the normalized weights $w_t^i$, where the area of the discs correspond to the size of the weights. We can see that as $t$ increases nearly all mass concentrates on the fourth particle $x_t^4$. This means that the normalized weight of the particle is almost one, $w_t^4 \approx 1$. The remaining particles have normalized weights that are all close to zero, and thus have a negligible contribution to the approximation. This concentration of mass for* SIS *to a single particle happens very quickly. Even for very simple Markovian models the variance of e.g. our normalization constant estimator can increase exponentially fast as a function of $T$.*

Sequential Monte Carlo methods tackle the weight degeneracy issue by choosing a proposal that leverages information contained in $\widehat{\gamma}_{t-1}$, the previous iteration's target distribution approximation.

## 2.2 Sequential Monte Carlo

Sequential Monte Carlo methods improve upon IS by mitigating the weight degeneracy issue through a clever choice of proposal distribution. For certain sequence models the weight degeneracy issue can be resolved altogether,

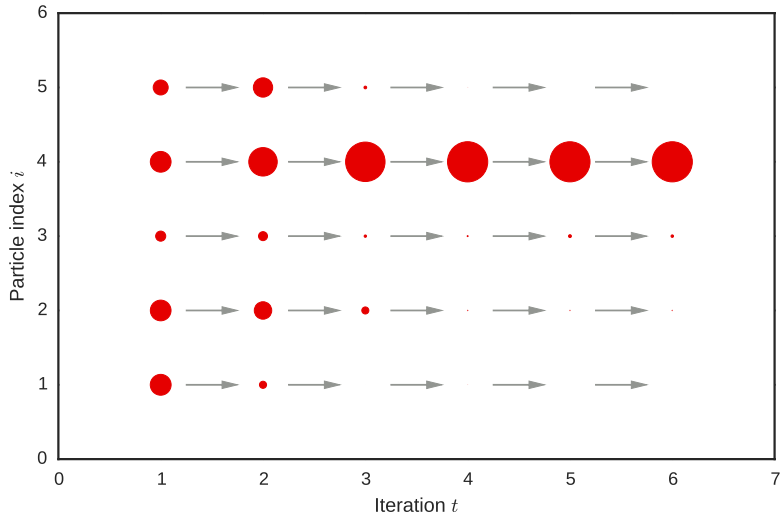



**Figure 2:** Weight degeneracy of the SIS method. The size of the disks represent the size of the corresponding weights $w_t^i$.

providing estimators to the final marginal distribution $\gamma_T(x_T)$ that do not deteriorate for increasing $T$. For other sequence models, SMC still tends to provide more accurate estimates in practice compared to IS.

Just like in SIS we need a sequence of proposal distributions $q_t(x_t \mid x_{1:t-1})$ for $t = 1, \ldots, T$. This is a user choice that can significantly impact the accuracy of the SMC approximation. For now we assume that the proposal is given and return to this choice in Section 3. Below, we detail the iterations (or steps) of a basic SMC algorithm.

**Step 1:** The first iteration of SMC boils down to approximating the target distribution $\gamma_1$ using standard IS. Simulating $N$ times independently from the first proposal

$$x_1^i \overset{\text{iid}}{\sim} q_1(x_1), \quad i = 1, \ldots, N, \tag{24}$$

and assigning corresponding weights

$$\widetilde{w}_1^i = \frac{\widetilde{\gamma}_1(x_1^i)}{q_1(x_1^i)}, \quad w_1^i = \frac{\widetilde{w}_1^i}{\sum_{j=1}^N \widetilde{w}_1^j}, \quad i = 1, \ldots, N, \tag{25}$$

lets us approximate $\gamma_1$ (cf. Eq. (19)) by

$$\widehat{\gamma}_1(x_1) = \sum_{i=1}^{N} w_1^i \delta_{x_1^i}(x_1). \tag{26}$$

The key innovation of the SMC algorithm is that it takes advantage of the information provided in $\widehat{\gamma}_1(x_1)$, Eq. (26), when constructing a proposal for the next target distribution $\gamma_2$.

**Step 2:** In the second iteration of SMC we sample $x_{1:2}$ from the proposal $\widehat{\gamma}_1(x_1)q_2(x_2 \mid x_1)$, rather than from $q_1(x_1)q_2(x_2 \mid x_1)$ like SIS. We sample $N$ times independently from

$$x_{1:2}^i \overset{\text{iid}}{\sim} \widehat{\gamma}_1(x_1)q_2(x_2 \mid x_1), \quad i = 1, \ldots, N, \tag{27}$$

and assign weights

$$\widetilde{w}_2^i = \frac{\widetilde{\gamma}_2(x_{1:2}^i)}{\widetilde{\gamma}_1(x_1^i)q_2(x_2^i \mid x_1^i)}, \quad w_2^i = \frac{\widetilde{w}_2^i}{\sum_{j=1}^N \widetilde{w}_2^j}, \quad i = 1, \ldots, N. \tag{28}$$

Simulating $x_{1:2}^i$, Eq. (27), can be broken down into parts: *resampling* $x_1^i \sim \widehat{\gamma}_1(x_1)$, *propagation* $x_2^i \mid x_1^i \sim q_2(x_2 \mid x_1^i)$, and concatenation $x_{1:2}^i = (x_1^i, x_2^i)$. Note the overloaded notation for $x_1^i$. We replace the initial sample set $\{x_1^i\}_{i=1}^N$ from Step 1, with the resampled set $x_1^i \sim \widehat{\gamma}_1(x_1)$, $i = 1, \ldots, N$.

Resampling can refer to a variety of methods in statistics, for our purpose it is simple (weighted) random sampling with replacement from $x_1^{1:N} = \{x_1^i\}_{i=1}^N$ with weights $w_1^{1:N} = \{w_1^i\}_{i=1}^N$. Resampling $N$ times independently means that the number of times each particle is selected is multinomially distributed. This resampling algorithm is known as *multinomial resampling*, see Example Code 3. In Sections 2.2.1 and 2.2.2 we revisit resampling and present alternative resampling algorithms, increasing efficiency by correlation and adaptation.

```
def multinomial_resampling(w, x):
    u = npr.rand(*w.shape)
    bins = np.cumsum(w)
    return x[np.digitize(u,bins)]
```

**Example Code 3:** Sampling $N$ times independently from $\sum_i w^i \delta_{x^i}$.

Propagation generates new samples independently from the proposal, i.e. $x_2^i \sim q_2(x_2 \mid x_1^i)$ for each $i = 1, \ldots, N$. By concatenating $x_{1:2}^i = (x_1^i, x_2^i)$ we obtain a complete sample from the proposal $\widehat{\gamma}_1(x_1)q_2(x_2 \mid x_1)$.

Finally, new importance weights are computed according to (28). This expression can be understood as a standard importance sampling weight—the target divided by the proposal. Note, however, that we use $\widetilde{\gamma}_1$ in the denominator: when computing the weights we "pretend" that the proposal is given by $\gamma_1(x_1)q_2(x_2 \mid x_1)$ rather than by its approximation $\widehat{\gamma}_1(x_1)q_2(x_2 \mid x_1)$. This is of course just an interpretation and the motivation for using this particular weight expression is given by the convergence analysis for SMC; see Section 2.3.

The resulting approximation of $\gamma_2$ is

$$\widehat{\gamma}_2(x_{1:2}) = \sum_{i=1}^{N} w_2^i \delta_{x_{1:2}^i}(x_{1:2}).$$

SMC can in essence be described as a synthesis of SIS and resampling, which explains its alternate names sequential importance resampling (SIR) or sequential importance sampling and resampling (SISR).

**Step $t$:** The remaining iterations follow the recipe outlined in step 2. First, the proposal is the product of the previous empirical distribution approximation and a conditional distribution

$$q_t(x_{1:t}) = \widehat{\gamma}_{t-1}(x_{1:t-1})q_t(x_t \mid x_{1:t-1}). \tag{29}$$

Samples for $i = 1, \ldots, N$ are generated as follows

$$\begin{aligned}
&\textit{resample} && x_{1:t-1}^i \sim \widehat{\gamma}_{t-1}(x_{1:t-1}),\\
&\textit{propagate} && x_t^i \mid x_{1:t-1}^i \sim q_t(x_t \mid x_{1:t-1}^i),\\
&\textit{concatenate} && x_{1:t}^i = \left(x_{1:t-1}^i, x_t^i\right).
\end{aligned}$$

Finally, we assign the weights

$$\widetilde{w}_t^i = \frac{\widetilde{\gamma}_t(x_{1:t}^i)}{\widetilde{\gamma}_{t-1}(x_{1:t-1}^i)q_t(x_t^i \mid x_{1:t-1}^i)}, \quad w_t^i = \frac{\widetilde{w}_t^i}{\sum_{j=1}^N \widetilde{w}_t^j}, \quad i = 1, \ldots, N, \tag{30}$$

and approximate $\gamma_t$ by

$$\widehat{\gamma}_t(x_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_{1:t}^i}(x_{1:t}).$$

**Algorithm 3:** Sequential Monte Carlo (SMC)

**input :** Unnormalized target distributions $\widetilde{\gamma}_t$, proposals $q_t$, number of samples $N$.

**output:** Samples and weights $\left\{\left(x_{1:t}^i, w_t^i\right)\right\}_{i=1}^N$ approximating $\gamma_t$, for $t = 1, \ldots, T$.

**for** $t = 1$ **to** $T$ **do**

  **for** $i = 1$ **to** $N$ **do**

    Sample $x_{1:t}^i \sim \widehat{\gamma}_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})$ (see Eq. (29))

    Set $\widetilde{w}_t^i = \frac{\widetilde{\gamma}_t(x_{1:t}^i)}{\widetilde{\gamma}_{t-1}(x_{1:t-1}^i) q_t(x_t^i \mid x_{1:t-1}^i)}$ (see Eq. (30))

  **end**

  Set $w_t^i = \frac{\widetilde{w}_t^i}{\sum_j \widetilde{w}_t^j}$, for $i = 1, \ldots, N$

**end**

The normalization constant $Z_t$ can be estimated by

$$\widehat{Z}_t = \prod_{k=1}^{t} \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_k^i. \tag{31}$$

We summarize the full sequential Monte Carlo sampler in Algorithm 3, where $\widehat{\gamma}_0 = \widetilde{\gamma}_0 = 1$, and $q_1(x_1 \mid x_{1:0}) = q_1(x_1)$.

The SMC method typically achieves drastic improvements compared to SIS. In Example 5 we return to our running example, using the same settings as in Example 4, to study the sample diversity and quality of the basic SMC method compared to SIS.

**Example 5** (SMC sample diversity)**.** *We illustrate the weights and resampling dependencies in Fig. 3. The grey arrows represent what sample from iteration $t-1$ that generated the current sample at iteration $t$, referred to as its* ancestor*. We can see that the weights tend to be more evenly distributed for* SMC*. The algorithm dynamically chooses to focus computational effort on more promising samples through the resampling step. Particles with low weights tend to not be resampled, and particles with high weights are resampled more frequently.*

*Not only do we get more diversity in our sample set,* SMC *also tends to find areas of higher probability. We illustrate this phenomenon in Table 1. We fix the number of particles $N = 10$, then study the average log-probability of our*

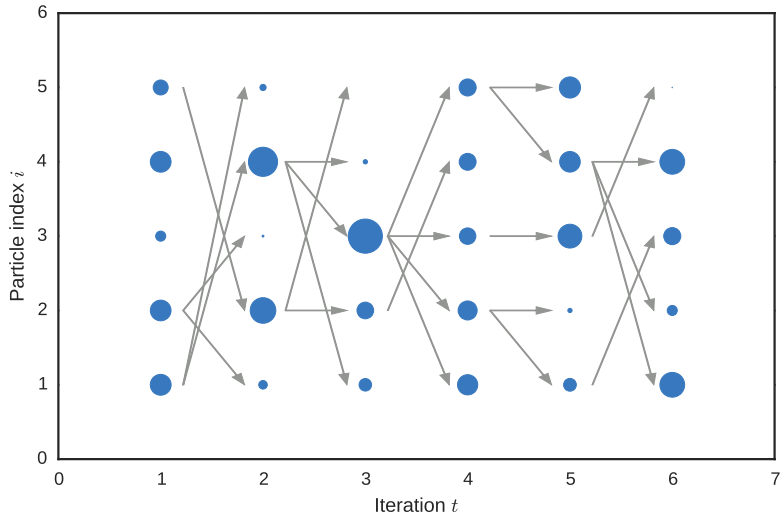



**Figure 3:** Diversity of samples in the SMC method. The size of the disks represent the size of the weights $w_t^i$, and the grey arrows represent resampling.

| | $\mathbb{E}_{\widehat{\gamma}_{10}}[\log \widetilde{\gamma}_{10}/10]$ | $\mathbb{E}_{\widehat{\gamma}_{20}}[\log \widetilde{\gamma}_{20}/20]$ | $\mathbb{E}_{\widehat{\gamma}_{40}}[\log \widetilde{\gamma}_{40}/40]$ |
|---|---|---|---|
| SIS | $-2.76$ | $-3.35$ | $-9.86$ |
| SMC | $\mathbf{-2.47}$ | $\mathbf{-2.51}$ | $\mathbf{-2.77}$ |

**Table 1:** Average log-probability values of the unnormalized target distribution with respect to the sampling distributions of SIS and SMC. The number of particles $N = 10$ is fixed for both methods.

*target distribution* $\log \widetilde{\gamma}_T$*, under the sampling distributions* $\widehat{\gamma}_T$*, normalized by the number of iterations* $T$*.*

While SMC methods do not suffer from weight degeneracy, the fundamental difficulty of simulating from a high-dimensional target still remains. For SMC this instead manifests itself in what is known as *path degeneracy*. We illustrate this property in Example 6.

**Example 6** (SMC path degeneracy)**.** *In Fig. 4 we reiterate the result from our previous example, Example 5. However, this time we only include the arrows corresponding to samples that have been consistently resampled and form our final approximation* $\widehat{\gamma}_T$*. We can see that our approximation for early iterations collapses back to a single ancestor, e.g. we have* $N = 5$ *identical copies of* $x_1^1$ *in Fig. 4.*

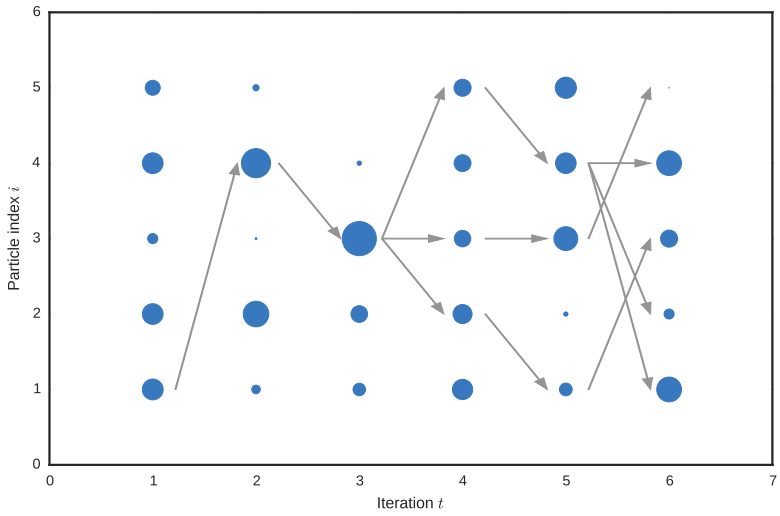



**Figure 4:** Path degeneracy of the SMC method for smoothing approximation. The size of the disks represent the size of the weights $w_t^i$, and the grey arrows represent resampling.

*This phenomena is known as path degeneracy and occurs because of the resampling mechanism. In Jacob et al. [2015] the authors study the impact this has on the memory requirements. They show that for state space models, under suitable conditions on the observation* PDF *$g$, the expected distance from the current iteration to a coalescence of the paths is bounded from above by $\mathcal{O}(N \log N)$.*

*In Fig. 5 we study the impact that increasing dependence on earlier iterations has on our* SMC *estimate of the log-normalization constant. We let $N = 20$, $T = 100$ be fixed and vary the value of $\beta \in (0, 1)$, where increasing values of $\beta$ correspond to more long-range dependencies in our non-Markovian* LVM*. We can see that for modest values of $\beta$ the* SMC *method achieves accurate estimates, whereas for higher values the drop-off in efficiency is significant. This manifests itself as an increase in the Monte Carlo (*MC*) variance in the estimate (width of bars), as well as a negative bias. As we will discuss in Section 2.3, this negative bias in the estimate of $\log Z_T$ is typical for* SMC *and* IS*.*

In Sections 2.2.1 and 2.2.2 we will discuss two standard practical approaches that help alleviate (but not solve) the issue of path degeneracy and improve overall estimation accuracy. The first is low-variance resampling — we lower the variance in the resampling step by correlating random variables. The second is adaptive resampling, which essentially means that we do not resample

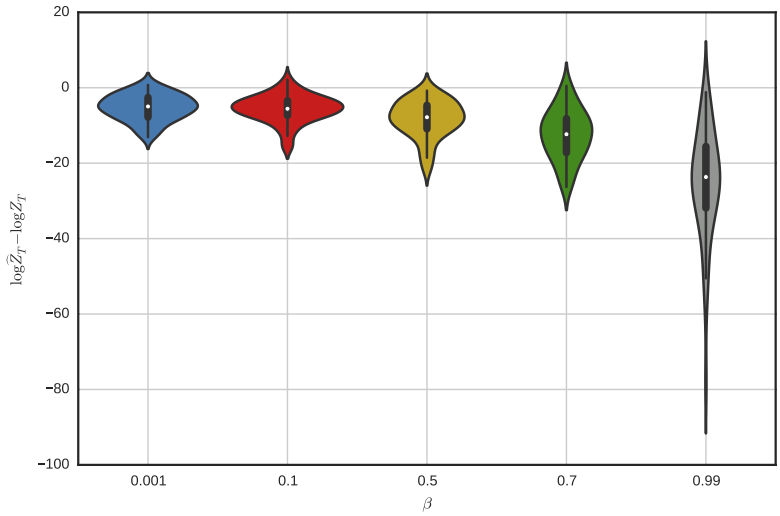



**Figure 5:** Violinplots for the log of the SMC estimate of the normalization constant divided by the true value, i.e. $\log \widehat{Z}_T - \log Z_T$. The number of particles are $N = 20$, length of the data is $T = 100$, and we study five different settings of $\beta = (0.001, 0.1, 0.5, 0.7, 0.99)$.

at each iteration but only on an as-needed basis. In Section 3 we go a step further and show how to choose, or learn, proposal and target distributions that lead to even further improvements.

### 2.2.1 Low-variance Resampling

The resampling step introduces extra variance in the short-term, using $\widehat{\gamma}_{t-1}$ to estimate $\gamma_{t-1}$ is better than using the resampled particle set. However, discarding improbable particles, and putting more emphasis on promising ones is crucial to the long-term performance of SMC. To keep long-term performance but minimize the added variance, we can employ standard variance reduction techniques based on correlating samples drawn from $\widehat{\gamma}_{t-1}$. First, we explain a common technique for implementing multinomial resampling based on inverse transform sampling. Then, we explain two low-variance resampling alternatives, stratified and systematic resampling.

To sample from $\widehat{\gamma}_{t-1}$ we use inverse transform sampling based on the weights $w^i_{t-1}$. We have that

$$x_{1:t-1} \sim \widehat{\gamma}_{t-1} \Leftrightarrow x_{1:t-1} = x^a_{1:t-1},$$

where $a$ is an integer random variable on $\{1,\ldots,N\}$, such that the following is true

$$\sum_{i=1}^{a-1} w_{t-1}^i \leq u < \sum_{i=1}^{a} w_{t-1}^i, \tag{32}$$

for $u \sim \mathcal{U}(0,1)$. Repeating the above process independently $N$ times gives multinomial resampling. That is, we draw $u^i \sim \mathcal{U}(0,1)$ and find the corresponding $a^i$ for each $i = 1,\ldots,N$. In the context of SMC the index variables $\{a^i\}_{i=1}^N$ determine the ancestry of the particles and they are therefore often referred to as *ancestor indices*.

**Stratified Resampling** One way to improve resampling is by stratification on the uniform random numbers $u^i$. This means we divide the unit interval into $N$ strata, $(0, {}^1/_N), ({}^1/_N, {}^2/_N), \ldots, (1 - {}^1/_N, 1)$. Then, we generate $u^i \sim \mathcal{U}({}^{i-1}/_N, {}^i/_N)$ for each strata $i = 1,\ldots,N$. Finally, the corresponding ancestor indices $a^i$ are given by studying Eq. (32).

Example Code 4 shows how this can be implemented in Python. The main change compared to multinomial resampling, Example Code 3, is the way the $u^i$'s are generated.

```
def stratified_resampling(w, x):
    N = w.shape[0]
    u = (np.arange(N) + npr.rand(N))/N
    bins = np.cumsum(w)
    return x[np.digitize(u,bins)]
```

**Example Code 4:** Sampling $N$ times from $\sum_i w^i \delta_{x^i}$ using stratification.

**Systematic Resampling** We can take correlation to the extreme by only generating a single uniform number $u \sim \mathcal{U}(0,1)$ to set all the $u^i$'s. Systematic resampling means that we let

$$u^i = \frac{i-1}{N} + \frac{u}{N},$$

where the random number $u \sim \mathcal{U}(0,1)$ is identical for all $i = 1,\ldots,N$. Then, we find corresponding indices $a^i$ by again studying Eq. (32). Note that just like in stratified resampling, systematic resampling also generates one $u^i$ in each strata $({}^{i-1}/_N, {}^i/_N)$. However, in this case the $u^i$'s are based on a single random

value $u$. This means that systematic resampling will be more computationally efficient than stratifies and multinomial resampling.

The code change to implement systematic resampling is simple. We only need to change a single line in Example Code 4. We replace line 3 by: `u = ( np.arange(N) + npr.rand())/N`.

Both systematic and stratified resampling are heavily used in practice. In many cases systematic resampling achieves slightly better results [Hol et al., 2006]. However, systematic resampling can sometimes lead to non-convergence [Gerber et al., 2017], depending on the ordering of the samples.

For a more thorough discussion on these, and other, resampling methods see e.g. Douc and Cappé [2005], Hol et al. [2006].

### 2.2.2 Effective Sample Size and Adaptive Resampling

The resampling step introduces extra variance by eliminating particles with low weights, and replicating particles with high weights. If the variance of the normalized weights is low, this step might be unnecessary. By tracking the variability of the normalized weights, and trigger a resampling step only when the variability crosses a pre-specified threshold, we can alleviate this issue. This is known as *adaptive resampling* in the SMC literature. Often we study the effective sample size (ESS) to gauge the variability of the weights, which for iteration $t$ is

$$\mathrm{ESS}_t = \frac{1}{\sum_{i=1}^{N} \left(w_t^i\right)^2}. \tag{33}$$

The ESS is a positive variable taking values in the continuous range between 1 and $N$. For IS the ESS is an approximation of the number of exact samples from $\gamma_t$ that we would need to achieve a comparable estimation accuracy [Doucet and Johansen, 2009]. A common approach is to resample only at iterations when the ESS falls below $N/2$.

Adaptive resampling can be implemented by slight alterations to the proposal and weight updates in Eqs. (29) and (30). If $\mathrm{ESS}_{t-1}$ is above the pre-specified threshold we simply omit the resampling step in Eq. (29) and obtain

$$\begin{aligned} &\textit{propagate} && x_t^i \,|\, x_{1:t-1}^i \sim q_t(x_t \,|\, x_{1:t-1}^i), \\ &\textit{concatenate} && x_{1:t}^i = \left(x_{1:t-1}^i, x_t^i\right). \end{aligned}$$

The fact that resampling is omitted at some iterations needs to be accounted for when computing the weights. For iterations where we do not resample the weight update is,

$$\widetilde{w}_t^i = \frac{w_{t-1}^i}{1/N} \cdot \frac{\widetilde{\gamma}_t(x_{1:t}^i)}{\widetilde{\gamma}_{t-1}(x_{1:t-1}^i) q_t(x_t^i \mid x_{1:t-1}^i)}, \quad w_t^i = \frac{\widetilde{w}_t^i}{\sum_{j=1}^N \widetilde{w}_t^j}, \quad i = 1, \ldots, N. \quad (34)$$

This can be thought of as adding an extra importance correction, where the previous weights $\{w_{t-1}^i\}_{i=1}^N$ define a "target distribution" for the ancestor indices and the factor $1/N$ is the "proposal" corresponding to not resampling. When the ESS falls below our pre-set threshold, we use the standard update Eqs. (29) and (30) with resampling instead.

Adaptive resampling is usually combined with the low-variance resampling techniques explained above for further variance reduction.

*Remark.* The constant factors in the weight expression will cancel when normalizing the weights. However, these constants still need to be included for the expression for the normalizing constant estimate Eq. (31) to be valid. That is, the extra importance correction added at those iterations when we do not resample should be the ratio of the *normalized weights* from the previous iteration, divided by the constant $1/N$. An alternative approach appearing in the literature is to neglect the constants when computing the weights, but then modify the expression for the normalizing constant estimate Eq. (31) instead.

## 2.3 Analysis and Convergence

Since its conception in the 1990s, significant effort has been spent on studying the theoretical properties of SMC methods. We will review and discuss a few select results in this section. For an early review of the area, see e.g. Del Moral [2004]. The theorems we discuss below all hold for a number of conditions on the proposal, probabilistic model, and test functions. For brevity we have omitted these exact conditions and refer to the cited proofs for details.

**Unbiasedness** One of the key properties of SMC approximations is that they provide *unbiased* approximations of integrals of functions $h_t$ with respect to the *unnormalized* target distribution $\widetilde{\gamma}_t$. We formalize this in Theorem 1.

**Theorem 1** (Unbiasedness)**.**

$$\mathbb{E}\left[\prod_{k=1}^{t}\left(\frac{1}{N}\sum_{j=1}^{N}\widetilde{w}_k^j\right)\cdot\sum_{i=1}^{N}w_t^i h_t(x_{1:t}^i)\right]=\int h_t(x_{1:t})\widetilde{\gamma}_t(x_{1:t})\,\mathrm{d}x_{1:t}$$

*Proof.* See Del Moral [2004, Theorem 7.4.2]. For the special case $h_t \equiv 1$ see also Section 4.1 and Appendix 4.A. □

A particularly important special case is when $h_t \equiv 1$ and we approximate the normalization constant of $\widetilde{\gamma}_t$. We have that $\mathbb{E}[\widehat{Z}_t] = Z_t$, where the expectation is taken with respect to all the random variables generated by the SMC algorithm. If we instead consider the more numerically stable $\log \widehat{Z}_t$, we have by Jensen's inequality that $\mathbb{E}\left[\log \widehat{Z}_t\right] \leq \log Z_t$. This means that the estimator of the log-normalization constant is negatively biased. This is illustrated by the violin plot discussed in Example 6.

We will delve deeper into the applications of the unbiasedness property and its consequences in Section 4.

**Laws of Large Numbers** While integration with respect to unnormalized distributions can be estimated unbiasedly, this is unfortunately not true when estimating expectations with respect to the normalized target distribution $\gamma_t$. However, SMC methods are still strongly consistent, leading to exact solutions when the number of particles $N$ tend to infinity. We formalize this law of large numbers in Theorem 2.

**Theorem 2** (Law of Large Numbers)**.**

$$\widehat{\gamma}_t(h_t) := \sum_{i=1}^{N} w_t^i h_t(x_{1:t}^i) \xrightarrow{a.s.} \gamma_t(h_t) = \int h_t(x_{1:t})\gamma_t(x_{1:t})\,\mathrm{d}x_{1:t},\ N \to \infty$$

*Proof.* See Del Moral [2004, Theorem 7.4.3]. □

**Central Limit Theorem** While the law of large numbers from the previous section shows that SMC approximations are exact in the limit of infinite computation, it tells us nothing about the quality of our estimate. The central limit theorem (CLT) in Theorem 3 tells us about the limiting distribution of our SMC estimate and its asymptotic variance. This gives us a first approach to get an understanding for the precision of our SMC approximation to $\gamma_t(h_t)$.

**Theorem 3** (Central Limit Theorem)**.**

$$\sqrt{N}\left(\widehat{\gamma}_t(h_t) - \gamma_t(h_t)\right) \xrightarrow{d} \mathcal{N}\left(0, V_t(h_t)\right),\ N \to \infty,$$

*where $V_t(\cdot)$ is defined recursively for a measurable function $h$,*

$$\begin{aligned}
V_t(h) &= \widetilde{V}_t\left(w'_t(x_{1:t})\left(h(x_{1:t}) - \gamma_t(h)\right)\right),\ t \geq 1,\\
\widetilde{V}_t(h) &= \widehat{V}_{t-1}\left(\mathbb{E}_{q_t(x_t \mid x_{1:t-1})}[h(x_{1:t})]\right) + \mathbb{E}_{\gamma_{t-1}}\left[Var_{q_t(x_t \mid x_{1:t-1})}(h)\right],\ t > 1,\\
\widehat{V}_t(h) &= V_t(h) + Var_{\gamma_t}(h),\ t \geq 1,
\end{aligned}$$

*initialized with $\widetilde{V}_1(h) = Var_{q_1}(h)$ and where*

$$\begin{aligned}
w'_t(x_{1:t}) &= \frac{\gamma_t(x_{1:t})}{\gamma_{t-1}(x_{1:t-1})q_t(x_t \mid x_{1:t-1})},\\
w'_1(x_{1:t}) &= \frac{\gamma_1(x_1)}{q_1(x_1)}.
\end{aligned}$$

*Proof.* See Chopin [2004]. □

**Sample Bounds** Another way of looking at the SMC method, disregarding test functions, is as a direct approximation to the target distribution itself. With this point of view we can study bounds on the difference between the distribution of a sample drawn from the SMC approximation compared to that of a sample drawn from the target distribution. Specifically, assume that we generate a sample $x'_{1:t}$ by first running an SMC sampler to generate an approximation $\widehat{\gamma}_t$ of $\gamma_t$, and then simulate $x'_{1:t} \sim \widehat{\gamma}_t$. Then, the marginal distribution of $x'_{1:t}$ is $\mathbb{E}[\widehat{\gamma}_t]$, where the expectation is taken with respect to all random variables generated by the SMC algorithm. It is worth noting that this distribution, $\mathbb{E}[\widehat{\gamma}_t]$, may be continuous despite the fact that $\widehat{\gamma}_t$ is a point-mass distribution by construction. In Theorem 4 we restate a generic sample bound on the KL divergence from the expected SMC approximation $\mathbb{E}[\widehat{\gamma}_t]$ to the target distribution $\gamma_t$.

**Theorem 4** (Sample Bound)**.**

$$KL\left(\mathbb{E}[\widehat{\gamma}_t] \| \gamma_t\right) \leq \frac{\mathscr{C}}{N},$$

*for a finite constant $\mathscr{C}$.*

*Proof.* See Del Moral [2004, Theorem 8.3.2]. □

From this result we can conclude that in fact the SMC approximation tends to the true target in distribution as the number of particles increase.

Recently there has been an increased interest for using SMC as an approximation to the target distribution, rather than just as a method for estimating expectations with respect to test functions. This point of view has found applications in e.g. probabilistic programming and variational inference [Wood et al., 2014, Naesseth et al., 2018, Huggins and Roy, 2015].

# 3 Learning Proposals and Twisting Targets

The two main design choices of sequential Monte Carlo methods are the proposal distributions $q_t$, $t \in \{1, \ldots, T\}$ and the intermediate (unnormalized) target distributions $\widetilde{\gamma}_t$, $t \in \{1, \ldots, T-1\}$. Carefully choosing these can drastically improve the efficiency of the algorithm and the accuracy of our estimation. Adapting both the proposal and the target distribution is especially important if the latent variables are high-dimensional.

In the first section below we discuss how to choose, or learn, the proposal distribution for a fixed target distribution. Then, in the final section we discuss how we can design a good sequence of intermediate target distributions for a given probabilistic model.

## 3.1 Designing the Proposal Distribution

The choice of the proposal distribution is perhaps the most important design choice for an efficient sequential Monte Carlo algorithm. A common choice is to propose samples from the model prior; this is simply known as the prior proposal. However, using the prior can lead to poor approximations for a small number of particles, especially if the latent space is high-dimensional.

We will in this section derive the locally (one-step) *optimal* proposal distribution, or *optimal* proposal for short. Because it is typically intractable, we further discuss various ways to either emulate it or approximate it directly. Finally, we discuss alternatives to learn efficient proposal distributions using an end-to-end variational perspective.

### 3.1.1 Locally Optimal Proposal Distribution

The locally optimal proposal distribution is the distribution we obtain if we assume that we have a perfect approximation at iteration $t-1$. Then, choose the proposal $q_t$ that minimizes the KL divergence from the joint distribution $\gamma_{t-1}(x_{1:t-1})q_t(x_t \mid x_{1:t-1})$ to $\gamma_t(x_{1:t})$. We formalize this result in Proposition 1. Equivalently, we can view this as the proposal minimizing the variance of the incremental weights, i.e. $\widetilde{w}_t^i / \widetilde{w}_{t-1}^i$, with respect to the newly generated samples $x_t^i$.

**Proposition 1** (Locally optimal proposal distribution)**.** *The optimal proposal*

$q_t^\star(x_t \mid x_{1:t-1})$ *minimizing* $KL(\gamma_{t-1}(x_{1:t-1})q_t(x_t \mid x_{1:t-1}) \| \gamma_t(x_{1:t}))$ *is given by*

$$q_t^\star(x_t \mid x_{1:t-1}) = \gamma_t(x_t \mid x_{1:t-1}) = \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_t(x_{1:t-1})}, \tag{35}$$

*where* $\widetilde{\gamma}_t(x_{1:t-1}) = \int \widetilde{\gamma}_t(x_{1:t}) \, \mathrm{d}x_t$.

*Proof.* If we let "const" denote terms constant with respect to the proposal distribution $q_t(x_t \mid x_{1:t-1})$, we get

$$\begin{aligned}
&\mathrm{KL}(\gamma_{t-1}(x_{1:t-1})q_t(x_t \mid x_{1:t-1}) \| \gamma_t(x_{1:t})) \\
&= \mathbb{E}_{\gamma_{t-1}q_t}[\log q_t(x_t \mid x_{1:t-1}) - \log \gamma_t(x_{1:t})] + \text{const} \\
&= \mathbb{E}_{\gamma_{t-1}q_t}[\log q_t(x_t \mid x_{1:t-1}) - \log \gamma_t(x_t \mid x_{1:t-1})] + \text{const} \\
&= \mathbb{E}_{\gamma_{t-1}(x_{1:t-1})}[\mathrm{KL}(q_t(x_t \mid x_{1:t-1}) \| \gamma_t(x_t \mid x_{1:t-1}))] + \text{const},
\end{aligned}$$

where the inner (conditional) Kullback-Leibler divergence is zero if and only if $q_t(x_t \mid x_{1:t-1}) \equiv \gamma_t(x_t \mid x_{1:t-1})$. □

In Example 7 we show that the optimal proposal is analytically tractable for our running non-Markovian Gaussian example.

**Example 7** (Optimal proposal for Example 1)**.** *If we let the target distribution be* $\widetilde{\gamma}_t(x_{1:t}) = \widetilde{\gamma}_{t-1}(x_{1:t-1}) f(x_t \mid x_{t-1}) g(y_t \mid x_{1:t})$*, then the (locally) optimal proposal for our running example is analytically tractable. It is*

$$\begin{aligned}
q_t^\star(x_t \mid x_{1:t-1}) = \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_t(x_{1:t-1})} &\propto f(x_t \mid x_{t-1}) g(y_t \mid x_{1:t}) \\
&\propto \mathcal{N}\left(x_t \mid \frac{r\phi x_{t-1} + q y_t - q \sum_{k=1}^{t-1} \beta^{t-k} x_k}{q+r}, \frac{qr}{q+r}\right).
\end{aligned}$$

In most practical cases the optimal proposal distribution is not a feasible alternative. The resulting importance weights are intractable, or simulating random variables from the optimal proposal is too computationally costly. Below we discuss various common approaches for approximating it.

### 3.1.2 Approximations to the Optimal Proposal Distribution

There have been a number of suggestions over the years on how to approximate the optimal distribution. We will review three analytic approximations based on a Gaussian assumption, as well as briefly describe an *exact approximation*.

**Laplace Approximation** We obtain the Laplace approximation to the optimal proposal by a second-order Taylor approximation of the log-PDF around a point $\bar{x}$ [Doucet et al., 2000]. If we let $l_t(x_t) := \log \gamma_t(x_t \mid x_{1:t-1})$, supressing the dependence on $x_{1:t-1}$, then

$$
\begin{aligned}
&l_t(x_t) \approx l_t(\bar{x}) + \nabla l_t(\bar{x})^\top (x_t - \bar{x}) + \frac{1}{2}(x_t - \bar{x})^\top \nabla^2 l_t(\bar{x})(x_t - \bar{x}) \\
&= \text{const} + \\
&\frac{1}{2}\left(x_t - \bar{x} + \left(\nabla^2 l_t(\bar{x})\right)^{-1} \nabla l_t(\bar{x})\right)^\top \nabla^2 l_t(\bar{x}) \left(x_t - \bar{x} + \left(\nabla^2 l_t(\bar{x})\right)^{-1} \nabla l_t(\bar{x})\right),
\end{aligned}
$$

where $\nabla l_t$ and $\nabla^2 l_t$ are the gradient and the Hessian of the log-PDF with respect to $x_t$, respectively. A natural approximation to the optimal proposal is then

$$
q_t(x_t \mid x_{1:t-1}) = \mathcal{N}\left(x_t \mid \bar{x} - \left(\nabla^2 l_t(\bar{x})\right)^{-1} \nabla l_t(\bar{x}), -\nabla^2 l_t(\bar{x})^{-1}\right). \tag{36}
$$

The mode of the distribution can be a good choice for the linearization point $\bar{x}$ if the distribution is unimodal. With this choice the mean simplifies to just $\bar{x}$. However, the mode is usually unknown and will depend on the value of $x_{1:t-1}$. This means that we are required to run a separate optimization for each particle $x^i_{1:t-1}$ and iteration $t$ to find the mode and Hessian. This can outweigh the benefits of the improved proposal distribution.

**Extended and Unscented Kalman Filter Approximations** The extended Kalman filter (EKF) and the unscented Kalman filter (UKF) [Anderson and Moore, 1979, Julier and Uhlmann, 1997] are by now standard solutions to non-linear and non-Gaussian filtering problems. We can leverage these ideas to derive another class of Gaussian approximations to the optimal proposal distribution [Doucet et al., 2000, Van Der Merwe et al., 2001].

The two methods depend on a structural equation representation of our probabilistic model,

$$
x_t = a(x_{1:t-1}, v_t), \tag{37a}
$$

$$
y_t = c(x_{1:t}, e_t), \tag{37b}
$$

where $v_t$ and $e_t$ are random variables. The representation implies a joint distribution on $x_t$ and $y_t$ conditional on $x_{1:t-1}$, i.e. $p(x_t, y_t \mid x_{1:t-1})$. The locally optimal proposal distribution corresponding to this representation is given by $q^\star_t(x_t \mid x_{1:t-1}) = p(x_t \mid x_{1:t-1}, y_t)$.

The EKF and UKF approximations rely on a Gaussian approximation to the joint conditional distribution of $x_t$ and $y_t$,

$$p(x_t, y_t \mid x_{1:t-1}) \approx \widehat{p}(x_t, y_t \mid x_{1:t-1}) = \mathcal{N}\left(\begin{pmatrix} x_t \\ y_t \end{pmatrix} \middle| \widehat{\mu}, \widehat{\Sigma}\right), \tag{38}$$

where we have suppressed the dependence on $x_{1:t-1}$ for clarity. We block the mean $\widehat{\mu}$ and covariance $\widehat{\Sigma}$ as follows

$$\widehat{\mu} = \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}, \quad \widehat{\Sigma} = \begin{pmatrix} \widehat{\Sigma}_{xx} & \widehat{\Sigma}_{xy} \\ \widehat{\Sigma}_{yx} & \widehat{\Sigma}_{yy} \end{pmatrix}. \tag{39}$$

Under the assumptions of the approximation in Eq. (38), the distribution of $x_t \mid x_{1:t-1}, y_t$ is tractable:

$$\widehat{p}(x_t \mid x_{1:t-1}, y_t) = \mathcal{N}(x_t \mid \mu_t, \Sigma_t), \tag{40}$$

with

$$\mu_t = \widehat{\mu}_x + \widehat{\Sigma}_{xy}\widehat{\Sigma}_{yy}^{-1}\left(y_t - \widehat{\mu}_y\right), \tag{41a}$$

$$\Sigma_t = \widehat{\Sigma}_{xx} - \widehat{\Sigma}_{xy}\widehat{\Sigma}_{yy}^{-1}\widehat{\Sigma}_{yx}. \tag{41b}$$

The key difference between the EKF- and UKF-based approximations is how to compute the estimates $\widehat{\mu}$ and $\widehat{\Sigma}$. We leave the details of these procedures to Appendix 3.A, and focus here on the intuition behind them.

The EKF uses a first-order Taylor approximation to the non-linear functions $a, c$ to derive an approximation to the full posterior distribution based on exact (analytical) updates of the approximate model. Following this line of thinking we can similarly linearize $a, c$ locally. Then under a Gaussian assumption on the noise $v_t, e_t$, we compute the distribution $\widehat{p}(x_t, y_t \mid x_{1:t-1})$ exactly for the linearized model. The EKF-based approximations $\widehat{\mu}$ and $\widehat{\Sigma}$ can be found in Eq. (81) in the appendix.

The UKF on the other hand uses so-called *sigma points* to reach the Gaussian approximation $\widehat{p}(x_t, y_t \mid x_{1:t-1})$. The key idea is to choose a set of points, pass them through the nonlinear functions $a$ and $c$, and then estimate the mean and variance of the transformed point-set. Unlike the EKF-based approximation, the UKF-based approximation does not require that the noises $v_t$ and $e_t$ are Gaussian distributed. However, we do require that the mean and variance for these random variables are available. The UKF-based approximations $\widehat{\mu}$ and $\widehat{\Sigma}$ can be found in Eq. (83) in the appendix.

**Algorithm 4:** Analytic Gaussian Proposal Approximations

Proposal: $q_t(x_t \mid x_{1:t-1}) = \mathcal{N}(x_t \mid \mu_t, \Sigma_t)$

- The Laplace approximation around the point $\bar{x}$ is

$$\mu_t = \bar{x} - \left(\nabla^2 l_t(\bar{x})\right)^{-1} \nabla l_t(\bar{x}),$$
$$\Sigma_t = -\nabla^2 l_t(\bar{x})^{-1},$$

  where $l_t(x_t) = \log \widetilde{\gamma}_t(x_{1:t})$ and derivatives are w.r.t. $x_t$.

- The EKF- and UKF-based approximations are

$$\mu_t = \widehat{\mu}_x + \widehat{\Sigma}_{xy}\widehat{\Sigma}_{yy}^{-1}\left(y_t - \widehat{\mu}_y\right),$$
$$\Sigma_t = \widehat{\Sigma}_{xx} - \widehat{\Sigma}_{xy}\widehat{\Sigma}_{yy}^{-1}\widehat{\Sigma}_{yx},$$

  where the variables are defined in Eq. (81) (EKF) and in Eq. (83) (UKF), respectively.

**Analytic Gaussian Approximations** We refer to the Laplace-, EKF- and UKF-based approximations as *analytic Gaussian* approximations to the locally optimal proposal distribution. We summarize these three approaches in Algorithm 4. Which one works best depends heavily on the model being studied. In Example 8 we study a simple example when $x$ and $y$ are one-dimensional random variables.

**Example 8** (Gaussian Proposal Approximations)**.** *We illustrate the analytic Gaussian approximations from Algorithm 4 based on a simple scalar example*

$$x = v, \quad v \sim \mathcal{N}(0, 1), \tag{42a}$$
$$y = \frac{(x+1)(x-1)(x-3)}{6} + e, \quad e \sim \mathcal{N}(0, 0.5). \tag{42b}$$

*The prior on the latent variable $x$ is a standard normal, but the measurement model $c$ is a polynomial in $x$ with additive Gaussian noise. In Fig. 6 we show the true (bimodal) posterior $p(x \mid y)$, and the corresponding approximations based on the Laplace,* EKF*, and* UKF *methods. Neither of these methods can capture the bimodality of the true normalized distribution, since they are all based on the basic simplifying assumption that the posterior is Gaussian. However, as previously*

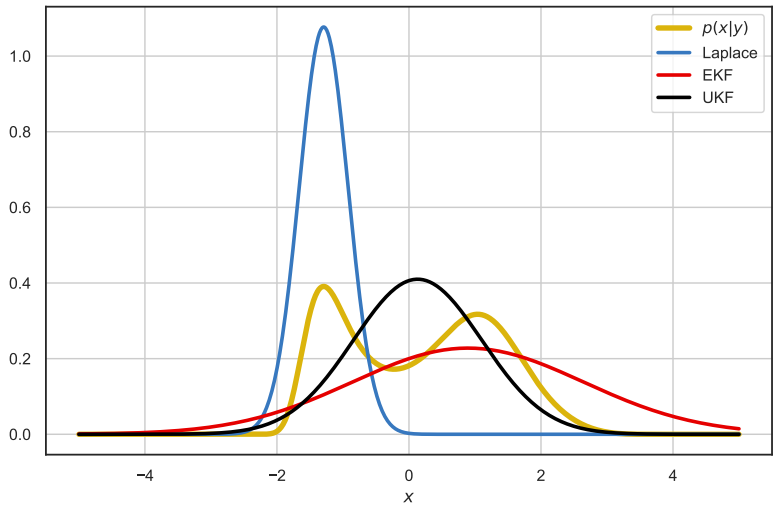



**Figure 6:** Analytic Gaussian approximations of $p(x \mid y)$ for the model in Eq. (42).

*discussed a good proposal will cover the bulk of probability mass of the target. This means it is likely that the* EKF *and* UKF *proposals will outperform the Laplace proposal in this case.*

**Exact Approximations** In *exact approximations* we use another level of MC. Instead of drawing exact samples from $q_t^\star$, we approximate draws from it using a nested MC algorithm [Naesseth et al., 2015, 2016]. This nested algorithm is chosen such that it does not alter the asymptotic exactness of the outer SMC method. We have discussed how SMC and IS can be used as distribution approximators. Nested MC methods takes this one step further, using a separate SMC (or IS) approximation $\widehat{q}_t^\star$ for each sample from $q_t^\star$ we would like. This can lead to substantial benefits for e.g. high-dimensional latent variables $x_t$. One of the key differences between the analytic Gaussian approximations and the exact approximations is the fixed form distribution assumption on the analytic approximation. The exact approximation is not limited to a standard parametric distribution, and we can obtain arbitrarily accurate approximations with enough compute. The trade-off is the additional effort needed to generate each sample $x_t^i$, which requires us to run independent SMC algorithms for each particle $i$ at each iteration $t$. While this might seem wasteful, it can in fact improve accuracy compared to SMC methods with standard proposals [Naesseth

**Algorithm 5:** Nested IS Approximation

**input :** Unnormalized target distributions $\widetilde{\gamma}_t$, $\widetilde{\gamma}_{t-1}$, resampled particle $x^i_{1:t-1}$, nested proposal $r_t$, number of samples $M$.
**output:** Sample and weight $x^i_t, \widetilde{w}^i_t$.
Sample $\bar{x}^{j,i}_t \sim r_t(x_t \mid x^i_{1:t-1}),\ j = 1, \ldots, M$.
Set $\widetilde{v}^{j,i}_t = \frac{\widetilde{\gamma}_t\left((x^i_{1:t-1}, \bar{x}^{j,i}_t)\right)}{\widetilde{\gamma}_{t-1}(x^i_{1:t-1}) r_t(\bar{x}^{j,i}_t \mid x^i_{1:t-1})}$.
Sample $x^i_t \sim \sum_{j=1}^M \frac{\widetilde{v}^{j,i}_t}{\sum_l \widetilde{v}^{l,i}_t} \delta_{\bar{x}^{j,i}_t}(x_t)$.
Set $\widetilde{w}^i_t = \frac{1}{M} \sum_{j=1}^M \widetilde{v}^{j,i}_t$.

et al., 2015], even for equal compute.

We are interested in approximating the locally optimal proposal distribution, i.e. $q^\star_t(x_t \mid x_{1:t-1}) \propto \widetilde{\gamma}_t(x_{1:t})$, separately for each particle $i$ using an MC method. A first approach is to use a nested IS sampler with a proposal $r_t(x_t \mid x_{1:t-1})$ targeting $q^\star_t(x_t \mid x_{1:t-1})$. We construct an approximation of the optimal proposal independently for each particle $x^i_{1:t-1}$ as follows

$$\widehat{q}^\star_t(x_t \mid x^i_{1:t-1}) = \sum_{j=1}^M \frac{\widetilde{v}^{j,i}_t}{\sum_l \widetilde{v}^{l,i}_t} \delta_{\bar{x}^{j,i}_t}(x_t), \quad \bar{x}^{j,i}_t \sim r_t(x_t \mid x^i_{1:t-1}), \tag{43}$$

where the weights $\widetilde{v}_t$ are given by

$$\widetilde{v}^{j,i}_t = \frac{\widetilde{\gamma}_t\left((x^i_{1:t-1}, \bar{x}^{j,i}_t)\right)}{\widetilde{\gamma}_{t-1}(x^i_{1:t-1}) r_t(\bar{x}^{j,i}_t \mid x^i_{1:t-1})}.$$

We replace $q_t(x_t \mid x_{1:t-1})$ in Eq. (29) with samples from the distribution defined in Eq. (43). The corresponding weight update, under the assumption that we have resampled $x^i_{1:t-1}$, is given by

$$\widetilde{w}^i_t = \frac{1}{M} \sum_{j=1}^M \widetilde{v}^{j,i}_t. \tag{44}$$

We summarize the nested IS approach to approximate the optimal proposal in Algorithm 5. The algorithm generates a single sample, and must be repeated for each $i = 1, \ldots, N$.

By what we know from the theory of SMC, see Section 2.3, we would expect that $\widehat{q}_t^\star \xrightarrow{\mathrm{d}} q_t^\star$ if we let $M \to \infty$. This is true, and as shown by Naesseth et al. [2016] the asymptotic variance in the limit of $M \to \infty$ is equal to that of using the locally optimal proposal distribution. The main implication is that we can obtain arbitrarily accurate optimal proposal approximations at the cost of increasing the number of samples $M$ for the nested MC method. We can also combine the analytic Gaussian approximations above with exact approximations by choosing $r_t$ as either the Laplace, EKF, UKF, or any other analytic approximation.

The computational complexity of the nested IS method is increased by a factor of $M$ compared to standard SMC. A natural question is therefore whether or not we are better of running standard SMC with $r_t$ as a proposal and using $NM$ particles. However, the memory requirement is lower for nested SMC and there are opportunities for speeding up through parallelization. There are even variants of nested MC that can outperform on the same computational budget [Naesseth et al., 2015], i.e. when compared to standard SMC with $NM$ particles without taking parallelization into account. We return to this class of methods in Section 4.3.2.

### 3.1.3 Learning a Proposal Distribution

Rather than relying on approximating the locally optimal proposal distribution, we can instead directly learn a good proposal distribution that can take into account *global* information of the target distribution. By parameterizing a suitable class of distributions and choosing a cost function to optimize, we can use standard optimization tools (such as stochastic gradient descent) to adapt our proposal to the problem we are trying to solve. When combined with the SMC approximation, we refer to these types of approaches as *adaptive* or *variational* SMC methods.

We will in this section focus on proposal distributions, $q_t$, parameterized by $\lambda$, we denote this as $q_t(x_t \mid x_{1:t-1}\,;\,\lambda)$. A common choice is to use a conditional Gaussian distribution

$$q_t(x_t \mid x_{1:t-1}\,;\,\lambda) = \mathcal{N}\left(x_t \mid \mu_\lambda(x_{1:t-1}), \sigma_\lambda^2(x_{1:t-1})\right),$$

where $\mu_\lambda(\cdot), \sigma_\lambda^2(\cdot)$ are neural networks parameterized by $\lambda$. The proposal is also a function of the data, either explicitly as a part of the input to the functions or implicitly through the cost function for the parameters. These types

of conditional distributions have recently been used with success in everything from image and speech generation [Gulrajani et al., 2017, Maddison et al., 2017] to causal inference [Louizos et al., 2017, Krishnan et al., 2017].

Below we discuss two classes of methods to learn the parameters $\lambda$, adaptive and variational methods, respectively.

**Adaptive Sequential Monte Carlo** Adaptive methods are characterized by choosing a cost function that tries to fit directly the proposal distribution $q_t(x_{1:T})$ to the target distribution $\gamma_t(x_{1:T})$. This can either be done locally at each iteration for $q_t(x_t \mid x_{1:t-1}; \lambda)$ as in e.g. Cornebise et al. [2008], or globally for $q_T(x_{1:T}; \lambda)$ as in e.g. Gu et al. [2015], Paige and Wood [2016]. Because we are mainly interested in the approximation of the final target distribution $\gamma_T$, we will here focus our attention to methods that take a global approach to learning proposals.

Since we generally need the proposal distribution to cover areas of high probability under the target distribution, we need the proposal to be more diffuse than the target. This will ensure that the weights we assign will have finite variance and that our approximation gets more accurate. This means that using the cost function $\mathrm{KL}(q_T(x_{1:T}; \lambda) \| \gamma_T(x_{1:T}))$, like in standard variational inference [e.g. Blei et al., 2017], would not result in a good proposal distribution. This cost function will encourage a proposal that is more concentrated than the target, and not less. Instead we focus our attention on the inclusive KL divergence $\mathrm{KL}(\gamma_T(x_{1:T}) \| q_T(x_{1:T}; \lambda))$ like in e.g. Gu et al. [2015] or Paige and Wood [2016]. The inclusive KL divergence encourages a $q_T$ that covers the high-probability regions of $\gamma_T$.

We focus our exposition on adaptive methods to the case when $\gamma_T(x_{1:T})$ is the posterior distribution $p(x_{1:T} \mid y_{1:T})$. The cost function to minimize is the inclusive KL divergence

$$\begin{aligned}\mathrm{KL}(p(x_{1:T} \mid y_{1:T}) \| q_T(x_{1:T}; \lambda)) &= \int p(x_{1:T} \mid y_{1:T}) \log \frac{p(x_{1:T} \mid y_{1:T})}{q_T(x_{1:T}; \lambda)} \mathrm{d}x_{1:T} \\ &= -\int p(x_{1:T} \mid y_{1:T}) \log q_T(x_{1:T}; \lambda) \mathrm{d}x_{1:T} + \text{const}, \end{aligned} \tag{45}$$

where "const" includes all terms constant with respect to the proposal distribution. If we could compute the gradients of Eq. (45) with respect to $\lambda$ we could employ a standard gradient descent method to optimize it. Unfortunately the

gradients, given by

$$\mathrm{g}_{\mathrm{KL}} = \nabla_\lambda \mathrm{KL}(p(x_{1:T} \mid y_{1:T}) \| q_T(x_{1:T}\,;\,\lambda)) = \\ -\mathbb{E}_{p(x_{1:T} \mid y_{1:T})}\left[\sum_{t=1}^{T} \nabla_\lambda \log q_t(x_t \mid x_{1:t-1}\,;\,\lambda)\right], \tag{46}$$

requires us to compute expectations with respect to the posterior distribution. Computing these types of expectations is the problem we try to solve with SMC in the first place! Below we will detail two approaches to solve this problem. First, we consider a stochastic gradient method that uses SMC to estimate the gradients. Then, we consider an alternative solution that uses *inference amortization* combined with stochastic gradient methods.

In the first approach, we use our current best guess for the parameters $\lambda$ and then the SMC procedure itself to approximate the gradient in Eq. (46). Given this estimate of the gradient we can do an update step based on standard stochastic gradient descent methods. Suppose we have the iterate $\lambda^{n-1}$, we estimate the gradient in Eq. (46) using Algorithm 3 with proposals $q_t(x_t \mid x_{1:t-1}\,;\,\lambda^{n-1})$, and get

$$\mathrm{g}_{\mathrm{KL}}^n \approx \widehat{\mathrm{g}}_{\mathrm{KL}}^n := \sum_{i=1}^{N} w_T^i \nabla_\lambda \log q_T(x_{1:T}^i\,;\,\lambda)\Big|_{\lambda=\lambda^{n-1}}. \tag{47}$$

With stochastic gradient descent we update our iterate by

$$\lambda^n = \lambda^{n-1} - \alpha_n \widehat{\mathrm{g}}_{\mathrm{KL}}^n, \tag{48}$$

where $\alpha_n$ are a set of positive step-sizes, typically chosen such that $\sum_n \alpha_n = \infty$ and $\sum_n \alpha_n^2 < \infty$. Note that unlike standard stochastic optimization methods, the gradient estimate is *biased* and convergence to a local minima of the cost function is not guaranteed. However, this approach has still shown to deliver useful proposal adaptation in practice [Gu et al., 2015].

The other approach relies on amortizing inference [Gershman and Goodman, 2014, Paige and Wood, 2016]. One view of inference amortization is as a procedure that does not just learn a single optimal $\lambda^\star$ for the observed dataset $y_{1:T}$, but rather learns a *mapping* $\lambda^\star(\cdot)$ from the data space to the parameter space. This mapping is optimized to make sure $q_T(x_{1:T}\,;\,\lambda^\star(y_{1:T}))$ is a good approximation to $p(x_{1:T} \mid y_{1:T})$ for any $y_{1:T}$ that is likely under the probabilistic model $p(y_{1:T})$. Note that the way we parameterize $q_T$ may differ from the above approach even though we use the same notation for the parameters $\lambda$.

In this setting instead of the KL divergence in Eq. (45), we consider minimizing

$$\begin{aligned}&\mathrm{KL}\left(p(x_{1:T}, y_{1:T}) \| p(y_{1:T}) q_T(x_{1:T}\,;\, \lambda(y_{1:T}))\right)\\ &= -\mathbb{E}_{p(x_{1:T}, y_{1:T})}\left[\log q_T(x_{1:T}\,;\, \lambda(y_{1:T}))\right] + \text{const.}\end{aligned} \tag{49}$$

If we let $\lambda_\eta(\cdot)$ be a parametric function with parameters $\eta$, we can compute stochastic gradients of Eq. (49) w.r.t. $\eta$ with no need to resort to SMC. A common choice is to let $\lambda_\eta$ be defined by a neural network where $\eta$ are the weights and biases. We define the gradient

$$\mathrm{g}_{\mathrm{KL}} = -\mathbb{E}_{p(x_{1:T}, y_{1:T})}\left[\nabla_\eta \log q_T(x_{1:T}\,;\, \lambda_\eta(y_{1:T}))\right], \tag{50}$$

and estimate it, for the current iterate $\eta^n$, using MC

$$\begin{aligned}&\mathrm{g}_{\mathrm{KL}}^n \approx \widehat{\mathrm{g}}_{\mathrm{KL}}^n :=\\ &-\nabla_\eta \log q_T(\bar{x}_{1:T}\,;\, \lambda_\eta(\bar{y}_{1:T}))\Big|_{\eta=\eta^{n-1}}, \quad (\bar{x}_{1:T}, \bar{y}_{1:T}) \sim p(x_{1:T}, y_{1:T}).\end{aligned} \tag{51}$$

Previous methods we have considered focus on proposals that try to emulate the posterior or the locally optimal proposal, both conditionally on the observed data $y_{1:T}$. However, the amortized inference approach, in this setting, learns proposals based on simulated data from the model $p(x_{1:T}, y_{1:T})$. This is performed offline before using the learned proposal and the real dataset for inference.

The amortized inference method follows the same procedure as above in Eq. (48) when updating $\eta^n$, replacing the gradient with the expression from Eq. (51)

$$\eta^n = \eta^{n-1} - \alpha_n \widehat{\mathrm{g}}_{\mathrm{KL}}^n. \tag{52}$$

Unlike the above approach that uses SMC to estimate the gradients, the amortized inference approach results in an unbiased approximation of the gradient. This means that using the update Eq. (52) will ensure convergence to a local minima of its cost function Eq. (49) by standard stochastic approximation results [Robbins and Monro, 1951]. On the other hand, this approach requires us to learn a proposal that works well for any dataset that could be generated by our model. This puts extra stress on the model to be accurate for the actual observed dataset to be able to learn a good proposal. For more thorough discussion of these topics, see Paige and Wood [2016].

We summarize these two approaches to optimize the proposal in Algorithm 6.

**Algorithm 6:** Adaptive SMC

**Stochastic Gradient** $q_t(x_t \mid x_{1:t-1}; \lambda)$

$$\lambda^n = \lambda^{n-1} - \alpha_n \widehat{g}^n_{\mathrm{KL}},$$

where $\widehat{g}^n_{\mathrm{KL}}$ is given in Eq. (47).

**Amortized Inference** $q_t(x_t \mid x_{1:t-1}; \lambda_\eta(y_{1:T}))$

$$\eta^n = \eta^{n-1} - \alpha_n \widehat{g}^n_{\mathrm{KL}},$$

where $\widehat{g}^n_{\mathrm{KL}}$ is given in Eq. (51).

(The stepsizes satisfy $\alpha_n > 0$, $\sum_n \alpha_n = \infty$, and $\sum_n \alpha_n^2 < \infty$.)

**Variational Sequential Monte Carlo** The key idea in variational sequential Monte Carlo (VSMC) [Naesseth et al., 2018, Le et al., 2018, Maddison et al., 2017] is to use the parametric distribution for our proposals, $q_t(x_t \mid x_{1:t-1}; \lambda)$, and optimize the fit in KL divergence from the expected SMC approximation $\mathbb{E}[\widehat{\gamma}_T]$ to the target distribution $\gamma_T$. The dependence on $\lambda$ enters implicitly in the expected SMC approximation $\mathbb{E}[\widehat{\gamma}_T]$ through the proposed samples, weights, and resampling step. In contrast to mimicking the locally optimal proposal distribution, this cost function takes into account the complete SMC approximation and defines a coherent global objective function for it. Compared to the previously described adaptive SMC methods, VSMC optimizes the fit of the final SMC distribution approximation to the true target, rather than the fit between the proposal to the target distribution. This means that we explicitly take into account the resampling steps that are a key part of the SMC algorithm.

Studying the marginal distribution of a single sample from $\widehat{\gamma}_T$, i.e. $\mathbb{E}[\widehat{\gamma}_T]$, it is possible to show that [Naesseth et al., 2018] the KL divergence from this distribution to the target distribution $\gamma_T$ can be bounded from above

$$\mathrm{KL}\left(\mathbb{E}[\widehat{\gamma}_T(x_{1:T}; \lambda)] \,\|\, \gamma_T(x_{1:T})\right) \leq -\mathbb{E}\left[\log \frac{\widehat{Z}_T}{Z_T}\right], \tag{53}$$

where the log-normalization constant estimate, cf. Eq. (31), is

$$\log \widehat{Z}_T = \sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_t\left(x^i_{1:t}; \lambda\right)\right). \tag{54}$$

The expectation is taken with respect to all random variables generated by the SMC algorithm. Because $\log Z_T$ does not depend on the parameters $\lambda$, minimizing Eq. (53) is equivalent to maximizing

$$\mathbb{E}\left[\log \widehat{Z}_T\right] = \mathbb{E}\left[\sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_t\left(x_{1:t}^i\,;\,\lambda\right)\right)\right]. \tag{55}$$

The KL divergence $\mathrm{KL}(\mathbb{E}[\widehat{\gamma}_T] \| \gamma_T)$ is non-negative, which means that Eq. (55) is a lower bound to the log-normalization constant $\log Z_T$. This is why Eq. (55) is typically known as an *evidence lower bound* in the variational inference literature. By maximizing the cost function in Eq. (55), with respect to $\lambda$, we can find a proposal that fits the complete SMC distribution to the target distribution. The main issue is that evaluating and computing the gradient of this cost function is intractable, so we need to resort to approximations.

We assume that the proposal distributions $q_t$ are *reparameterizable* [Kingma and Welling, 2014, Rezende et al., 2014, Ruiz et al., 2016, Naesseth et al., 2017]. This means we can replace simulating $x_t \,|\, x_{1:t-1} \sim q_t(\cdot \,|\, x_{1:t-1})$ by

$$x_t = h_t(x_{1:t-1}, \varepsilon_t\,;\,\lambda), \quad \varepsilon_t \sim p(\varepsilon), \tag{56}$$

where the distribution of the random variable $\varepsilon_t$ is independent of the parameters $\lambda$. If we further assume that $h_t$ is differentiable we can use

$$\nabla_\lambda \mathbb{E}\left[\log \widehat{Z}_T\right] \approx \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} w_t^i \nabla_\lambda \log \widetilde{w}_t\left(x_{1:t}^i\,;\,\lambda\right)\right] =: \mathrm{g}_{\mathrm{VSMC}}, \tag{57}$$

where $\nabla_\lambda \log \widetilde{w}_t(\cdot)$ can be computed using e.g. automatic differentiation, replacing $x_{1:t}^i$ with its defintion through Eq. (56). The approximation follows from ignoring the gradient part that results from the resampling step, which has been shown to work well in practice [Naesseth et al., 2018, Le et al., 2018, Maddison et al., 2017].

Just like for the adaptive SMC methods, Eq. (57) suggests a stochastic gradient method to optimize $\lambda$. Given an iterate $\lambda^{n-1}$, we compute the gradient estimate by running SMC with proposals $q_t(x_t \,|\, x_{1:t-1}\,;\,\lambda^{n-1})$ and evaluate

$$\mathrm{g}_{\mathrm{VSMC}} \approx \widehat{\mathrm{g}}^n_{\mathrm{VSMC}} = \sum_{t=1}^{T}\sum_{i=1}^{N} w_t^i \nabla_\lambda \log \widetilde{w}_t\left(x_{1:t}^i\,;\,\lambda\right)\Big|_{\lambda=\lambda^{n-1}}. \tag{58}$$

**Algorithm 7:** Variational SMC

*Until convergence:*
Run Algorithm 3 with proposals $q_t(x_t \mid x_{1:t-1}; \lambda^{n-1})$.
Update parameters $\lambda^n = \lambda^{n-1} + \alpha_n \widehat{g}^n_{\text{VSMC}}$, where $\widehat{g}^n_{\text{VSMC}}$ is given in Eq. (58).
(The stepsizes satisfy $\alpha_n > 0$, $\sum_n \alpha_n = \infty$, and $\sum_n \alpha_n^2 < \infty$.)

With stochastic gradient ascent we update our iterate by

$$\lambda^n = \lambda^{n-1} + \alpha_n \widehat{g}^n_{\text{VSMC}}, \tag{59}$$

where $\alpha_n$ are positive step-sizes, chosen such that $\sum_n \alpha_n = \infty$ and $\sum_n \alpha_n^2 < \infty$.

We summarize the VSMC algorithm to adapt the proposals in Algorithm 7.

## 3.2 Adapting the Target Distribution

A less well-known design aspect of SMC algorithms is the target distribution itself. If we are mainly interested in the final distribution $\gamma_T$, we are free to choose the intermediate target distributions $\gamma_t$ to maximize the accuracy of our estimate $\widehat{\gamma}_T$. By making use of information such as future observations, we can change the target distribution to take this into account. This leads to so-called *auxiliary* or *twisted* SMC methods [Guarniero et al., 2017, Heng et al., 2017].

### 3.2.1 Twisting the Target

Even if we could simulate exactly from the locally optimal proposal distribution Eq. (35), we still would not be getting exact samples from our target distribution. The reason for this is that when sampling and weighting our particles at iteration $t$ we do not take into account potential future iterations. These future iterations can add new information regarding earlier latent variables. For instance, in an LVM (cf. Eq. (9)) a natural choice of target distributions is

$$\widetilde{\gamma}_t(x_{1:t}) = p(x_1) g_1(y_1 \mid x_1) \prod_{k=2}^{t} f_k(x_k \mid x_{1:k-1}) g_k(y_k \mid x_{1:k}).$$

However, with this choice we do not take future observations $y_{t+1}, \ldots, y_T$ into account at iteration $t$. The SMC approximation for earlier iterations has finite

support (represented by the particles $x^i_{1:t-1}$) and so the only way we can incorporate this is by reweighting and resampling. These two operations will typically impoverish our approximation for the full states $x_{1:t}$, a phenomena related to the path degeneracy discussed in Section 2. The key idea in twisting (or *tilting*) the target distribution is to change our intermediate target distributions to take into account information from future iterations already at the current step. These types of approaches also have a connection to so-called lookahead strategies [Lin et al., 2013] and block sampling [Doucet et al., 2006] for SMC.

The optimal target distribution, under the assumption that we are using the locally optimal proposal, is to let the target at each iteration be the marginal distribution of the final iteration's target, i.e.

$$\gamma^\star_t(x_{1:t}) = \gamma_T(x_{1:t}). \tag{60}$$

With this choice all samples from $x_{1:T} \sim \widehat{\gamma}_T$ are perfect samples from $\gamma_T$. We can easily see this if we write out the optimal proposal for this sequence of targets

$$q^\star_t(x_t \mid x_{1:t-1}) = \gamma^\star_t(x_t \mid x_{1:t-1}) = \gamma_T(x_t \mid x_{1:t-1}) = p(x_t \mid x_{1:t-1}, y_{1:T}),$$

where the final equality corresponds to the case when $\gamma_T$ is the posterior distribution $p(x_{1:T} \mid y_{1:T})$ for an LVM. This means that the locally optimal proposal distribution is no longer only locally optimal, it is in fact optimal in a global sense.

In the following discussion we will assume that the final unnormalized target distribution $\widetilde{\gamma}_T(x_{1:T})$ can be split into a product of factors:

$$\widetilde{\gamma}_T(x_{1:T}) = \prod_{t=1}^{T} \widetilde{f}_t(x_{1:t}). \tag{61}$$

With this form of the joint distribution of data and latent variables, we choose the following structure for our unnormalized target distributions:

$$\widetilde{\gamma}_t(x_{1:t}) = \widetilde{\gamma}_{t-1}(x_{1:t-1})\widetilde{f}_t(x_{1:t})\frac{\psi_t(x_{1:t})}{\psi_{t-1}(x_{1:t-1})}, \tag{62}$$

where $\psi_t \geq 0$ are our so-called *twisting potentials*, with $\psi_0 \equiv \psi_T \equiv 1$. It is easy to check that this recursive definition is consistent with our assumed

factorization of the final target $\widetilde{\gamma}_T(x_{1:T})$ in Eq. (61). The target distribution structure postulated in Eq. (62) can be directly applied in our basic SMC method in Algorithm 3 without any changes.

To deduce the optimal twisting potentials $\psi_t^\star$ we can make use of the property that the optimal targets must fulfill

$$\widetilde{\gamma}_t^\star(x_{1:t}) = \int \widetilde{\gamma}_{t+1}^\star(x_{1:t+1})\,\mathrm{d}x_{t+1}, \quad t = 1, \ldots, T-1. \tag{63}$$

This follows from Eq. (60), each target $\gamma_t^\star$ should be the marginal distribution of $x_{1:t}$ under the final target distribution $\gamma_T$. By replacing $\widetilde{\gamma}_t^\star$ in Eq. (63) with the definition from Eq. (62) we get

$$\psi_t^\star(x_{1:t}) = \int \widetilde{f}_{t+1}(x_{1:t+1})\psi_{t+1}^\star(x_{1:t+1})\,\mathrm{d}x_{t+1}. \tag{64}$$

While Eq. (64) is typically not available analytically for practical applications, we will see in Section 3.2.2 that it can serve as a guideline for designing tractable twisting potentials.

Below we give a few concrete examples on what the optimal potentials might look like for both non-Markovian LVMs and conditionally independent models with tempering.

**Non-Markovian Latent Variable Model** The non-Markovian LVM can be described by a transition PDF $f_t$ and an observation PDF $g_t$ and follows the structure in Eq. (61) by defining

$$\widetilde{f}_t(x_{1:t}) = f_t(x_t \mid x_{1:t-1})g_t(y_t \mid x_{1:t}, y_{1:t-1}). \tag{65}$$

For this model we can rewrite Eq. (64) as follows

$$\begin{aligned}\psi_t^\star(x_{1:t}) &= \mathbb{E}_{f_{t+1}(x_{t+1} \mid x_{1:t})}\left[g_{t+1}(y_{t+1} \mid x_{1:t+1}, y_{1:t})\psi_{t+1}^\star(x_{1:t+1})\right] \\ &= \mathbb{E}_{\prod_{k=t+1}^T f_k(x_k \mid x_{1:k-1})}\left[\prod_{l=t+1}^{T} g_l(y_l \mid x_{1:l}, y_{1:l-1})\right] = p(y_{t+1:T} \mid x_{1:t}),\end{aligned} \tag{66}$$

where the final expression is the predictive likelihood of $y_{t+1:T}$ given the latent variables $x_{1:t}$. In Example 9 we derive the optimal twisting potentials for our Gaussian non-Markovian LVM.

**Example 9** (Analytical Twisting Potential)**.** *Our running example is a non-Markovian* LVM *with an analytical optimal twisting potential $\psi_t^\star$. This is because the joint distribution is Gaussian, and thus the integrals we need to compute to construct it (cf. Eq.* (65)*) are tractable. We can rewrite the equation for the observations $y_t$ for each $t$ only as a function of $x_{1:l}$, $e_t$ and $v_{l+1:t}$, denoted by $y_{t\,|\,l}$, as follows*

$$y_{t\,|\,l} = \left( \sum_{k=l+1}^{t} \beta^{t-k} \phi^{k-l} \right) x_l + \sum_{k=1}^{l} \beta^{t-k} x_k + e_t + \sum_{k=l+1}^{t} \gamma_k v_k, \tag{67}$$

*where $\gamma_k$ is given by*

$$\gamma_k = \sum_{m=k}^{t} \beta^{t-m} \phi^{m-k}.$$

*Because we know that $\psi_t^\star(x_{1:t}) = p(y_{t+1:T} \,|\, x_{1:t})$ we get the optimal value by considering the distribution of $(y_{t+1\,|\,t}, \ldots, y_{T\,|\,t})^\top$. Since $e_t$ and $v_t$ are independent Gaussian, by Eq.* (67) *the predictive likelihood is a correlated multivariate Gaussian.*

**Conditionally Independent Models** Another class of probabilistic models we discussed in Section 1.2 was conditionally independent models. In the notation of Eq. (62) we can express this class of models as

$$\widetilde{f}_t(x_{1:t}) = s_t(x_{t-1} \,|\, x_t) \frac{p(x_t) g_t(x_t, \mathbf{y})}{p(x_{t-1}) g_{t-1}(x_{t-1}, \mathbf{y})}, \quad t = 2, \ldots, T, \tag{68}$$

where $p(x)$ is the prior distribution on the (static) latent variable, $s_t$ is a backward kernel, and $g_t$ is a *potential* depending on the observed data $\mathbf{y}$. We initialize by $\widetilde{f}_1(x_1) = p(x_1) g_1(x_1, \mathbf{y})$. The potential $g_t$ usually takes either of the two following forms:

$$g_t(x_t, \mathbf{y}) = \prod_{k=1}^{t} p(y_k \,|\, x_t), \quad \text{(data tempering)}$$

$$g_t(x_t, \mathbf{y}) = \left( \prod_{k=1}^{K} p(y_k \,|\, x_t) \right)^{\tau_t}, \quad \text{(likelihood tempering)}$$

where $0 = \tau_1 < \tau_2 < \ldots < \tau_T = 1$. The backward kernel $s_t$ is an artificially introduced distribution to enable the use of SMC methods, requiring approximating a space of increasing dimension. See Section 1.2 for more examples of

useful sequence models in annealing methods, or Del Moral et al. [2006] for a more thorough treatment of the subject.

When we replace $\widetilde{f}_t$ in Eq. (64) with its definition from Eq. (68) we get

$$\begin{aligned}\psi_t^\star(x_{1:t}) &= \int \frac{p(x_{t+1})g_{t+1}(x_{t+1},\mathbf{y})}{p(x_t)g_t(x_t,\mathbf{y})} s_{t+1}(x_t \mid x_{t+1})\psi_{t+1}^\star(x_{1:t+1})\,\mathrm{d}x_{t+1} \\ &= \frac{1}{p(x_t)g_t(x_t,\mathbf{y})} \int p(x_T)g_T(x_T,\mathbf{y}) \prod_{k=t+1}^{T} s_k(x_{k-1} \mid x_k)\,\mathrm{d}x_{t+1:T}, \end{aligned} \tag{69}$$

where we can see that because of the (reverse) Markov structure in the annealing model from Eq. (68), the optimal twisting potential only depends on the current value of $x_t$, i.e. $\psi_t^\star(x_{1:t}) = \psi_t^\star(x_t)$. This will hold true also for the SSM, the Markovian special case of Eq. (65) where $f_t(x_t \mid x_{1:t-1}) = f_t(x_t \mid x_{t-1})$ and $g_t(y_t \mid x_{1:t}, y_{1:t-1}) = g_t(y_t \mid x_t)$.

### 3.2.2 Designing the Twisting Potentials

The optimal twisting potentials are typically not tractable, requiring us to solve integration problems that might be as difficult to compute as finding the posterior itself. However, they can give insight into how to design and parameterize approximate twisting functions. First, we study a certain class of models that admit a *locally optimal* twisting potential, which gives rise to the so-called *fully adapted* SMC [Pitt and Shephard, 1999, Johansen and Doucet, 2008]. After discussing the locally optimal choice, we then move on to discuss a general approach to learning twisting potentials for SMC algorithms.

**Locally Optimal Twisting Potential** If we assume that the twisting potentials only satisfy the optimal twisting equation, Eq. (64), for one iteration we get a locally optimal twisting potential

$$\psi_t(x_{1:t}) = \int \widetilde{f}_{t+1}(x_{1:t+1})\,\mathrm{d}x_{t+1}. \tag{70}$$

The target distribution based on these potentials will allow us to adapt for information from one step into the future.

Furthermore, we combine the locally optimal twisting potentials with the

proposals based only on factors up until iteration $t$, i.e.

$$q_t(x_t \mid x_{1:t-1}) \propto \prod_{k=1}^{t} \widetilde{f}_k(x_{1:k}) \quad \Longrightarrow \quad q_t(x_t \mid x_{1:t-1}) = \frac{\widetilde{f}_t(x_{1:t})}{\psi_{t-1}(x_{1:t-1})}. \tag{71}$$

With these choices of twisting functions and proposals, based on the twisted target structure Eq. (62), we obtain the fully adapted SMC [Pitt and Shephard, 1999, Johansen and Doucet, 2008] with corresponding weights

$$\widetilde{w}_t(x_{1:t}) = \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_t(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})} = \psi_t(x_{1:t}). \tag{72}$$

Note that this perspective on fully adaptive SMC approximates the "predictive target"

$$\widetilde{\gamma}_t(x_{1:t}) = \underbrace{\int \widetilde{f}_{t+1}(x_{1:t+1}) \, \mathrm{d}x_{t+1}}_{\psi_t(x_{1:t})} \cdot \prod_{k=1}^{t} \widetilde{f}_k(x_{1:k}),$$

including information from iteration $t+1$ already at iteration $t$. In some situations we might be interested in the "filtering target" $\prod_{k=1}^{t} \widetilde{f}_k(x_{1:k})$. If so, we can make use of the approach described above to generate the particles $x_{1:t}^i$. Then to estimate the filtering target we simply average the particles $x_{1:t}^i$, ignoring the weights in Eq. (72), accounting for the discrepancy between the filtering and predictive targets.

**Example 10** (Locally Optimal Twisting for Non-Markovian LVM)**.** *Recall the non-Markovian latent variable model with factors given by (see Eq.* (65)*)* $\widetilde{f}_t(x_{1:t}) = f_t(x_t \mid x_{1:t-1}) g_t(y_t \mid x_{1:t}, y_{1:t-1}) = p(y_t, x_t \mid x_{1:t-1}, y_{1:t-1})$*. For this model the locally optimal twisting potential Eq.* (70) *is given by*

$$\psi_t(x_{1:t}) = \mathbb{E}_{f_{t+1}(x_{t+1} \mid x_{1:t})}[g_{t+1}(y_{t+1} \mid x_{1:t+1}, y_{1:t})] = p(y_{t+1} \mid x_{1:t}).$$

*The proposal in Eq.* (71) *becomes*

$$q_t(x_t \mid x_{1:t-1}) = \frac{p(y_t, x_t \mid x_{1:t-1}, y_{1:t-1})}{p(y_t \mid x_{1:t-1})} = p(x_t \mid x_{1:t-1}, y_{1:t})$$

*with weights according to Eq.* (72) *given by* $\widetilde{w}_t(x_{1:t}) = p(y_{t+1} \mid x_{1:t})$*. This can be recognized as a one-step look-ahead strategy, incorporating the observation* $y_{t+1}$ *already at iteration* $t$*. This is indeed how the fully adaptive* SMC *is often presented for* SSMs *and non-Markovian* LVMs*.*

*Remark.* The proposal used above is not the locally optimal proposal distribution for our (predictive) target distribution. The locally optimal proposal (see Eq. (35)) for the target Eq. (62) is

$$q_t(x_t \mid x_{1:t-1}) \propto \widetilde{f}_t(x_{1:t})\psi_t(x_{1:t}),$$

which would result in weights

$$\widetilde{w}_t(x_{1:t}) = \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_t(x_{1:t-1})q_t(x_t \mid x_{1:t-1})} = \frac{\int \widetilde{f}_t(x_{1:t})\psi_t(x_{1:t})\,\mathrm{d}x_t}{\psi_{t-1}(x_{1:t-1})}. \tag{73}$$

Just like in the non-twisted case the weights for the locally optimal proposal are independent of the actual samples we generate at iteration $t$ of the SMC algorithm. However, even if the potentials in Eq. (70) are available, the integration in Eq. (73) is not necessarily tractable.

Except for a few special cases the locally optimal twisting potential is not available, therefore we resort to approximating it. We can extend the ideas from Section 3.1 and either make an approximation to the locally optimal potential, or we can learn a full twisting potential.

**Learning Twisting Potentials** By parameterizing a class of positive functions and a cost function, we can effectively learn useful twisting potentials for the application at hand. From Eq. (64) we know that the optimal twisting potentials must satisfy a recursive relationship. We can make use of this property as we define the cost function and the twisting potentials.

We explain the approach by Guarniero et al. [2017], but generalize slightly to allow for non-Markovian LVMs. This means that our model is defined by $\widetilde{f}_t = f_t(x_t \mid x_{1:t-1})g_t(y_t \mid x_{1:t})$, cf. Eq. (65). Furthermore, we define the twisting potentials $\psi_t(x_{1:t})$ implicitly as expectations with respect to $f_{t+1}$ of positive functions $\bar{\psi}_{t+1}(x_{1:t+1}\,;\,\rho_{t+1})$, i.e.

$$\psi_t(x_{1:t}\,;\,\rho_{t+1}) := \mathbb{E}_{f_t(x_{t+1} \mid x_{1:t})}\big[\bar{\psi}_{t+1}(x_{1:t+1}\,;\,\rho_{t+1})\big], \quad t = 1, \ldots, T-1, \tag{74}$$

We further assume that this expectation can be computed analytically. With this setup we focus on learning the $\bar{\psi}_t$'s instead. Using the optimality condition from Eq. (64) with twisting potentials defined by Eq. (74), we get the recursive approximation criteria for the $\bar{\psi}_t$'s

$$\bar{\psi}_T(x_{1:T}\,;\,\rho_T) \approx g_T(y_T \mid x_{1:T}), \tag{75a}$$

$$\bar{\psi}_t(x_{1:t}\,;\,\rho_t) \approx g_t(y_t \mid x_{1:t})\mathbb{E}_{f_t(x_{t+1} \mid x_{1:t})}\big[\bar{\psi}_{t+1}(x_{1:t+1}\,;\,\rho_{t+1})\big]. \tag{75b}$$

Because we assumed that $\mathbb{E}_{f_t(x_t \mid x_{1:t-1})}\left[\bar{\psi}_t(x_{1:t}\,;\rho_t)\right]$ is available analytically, and Eq. (75) is telling us to view it as an approximation of $p(y_{t:T} \mid x_{1:t-1})$, it makes sense to use as the proposal

$$q_t(x_t \mid x_{1:t-1}\,;\rho_t) = \frac{f_t(x_t \mid x_{1:t-1})\bar{\psi}_t(x_{1:t}\,;\rho_t)}{\mathbb{E}_{f_t(x_t \mid x_{1:t-1})}\left[\bar{\psi}_t(x_{1:t}\,;\rho_t)\right]}. \tag{76}$$

With this choice of proposal we tie together the parameters for the proposal and the potentials. The corresponding weights in the SMC algorithm are

$$\widetilde{w}_t(x_{1:t}\,;\rho_{t:t+1}) = \frac{g_t(y_t \mid x_{1:t})\psi_t(x_{1:t}\,;\rho_{t+1})}{\bar{\psi}_t(x_{1:t}\,;\rho_t)}. \tag{77}$$

What remains is how to use Eq. (75) to actually learn a useful set of potentials. To do this we follow the approach by Guarniero et al. [2017] which uses an iterative refinement process: initialize parameters, run SMC with the current parameters, update parameters based on the current SMC approximation $\widehat{\gamma}_t$, repeat. The update step solves for $\rho_t^n$ by recursively minimizing

$$\rho_T^n = \arg\min_{\rho_T} \sum_{i=1}^{N} w_T^i \left(\bar{\psi}_T(x_{1:T}^i\,;\rho_T) - g_T(y_T \mid x_{1:T}^i)\right)^2, \tag{78a}$$

$$\rho_t^n = \arg\min_{\rho_t} \sum_{i=1}^{N} w_t^i \left(\bar{\psi}_t(x_{1:t}^i\,;\rho_t) - g_t(y_t \mid x_{1:t}^i)\psi_t(x_{1:t}^i\,;\rho_{t+1}^n)\right)^2, \tag{78b}$$

where the last step is for $t = T-1,\ldots,1$. With the updated set of parameters, we re-run SMC and repeat the updates in Eq. (78) if a stopping criteria has not been met. Guarniero et al. [2017] proposes a stopping criteria based on the normalization constant estimate $\widehat{Z}_T$.

We summarize the iterated auxiliary SMC method explained above in Algorithm 8, and give an example setup for a stochastic RNN in Example 11.

**Example 11** (Stochastic Recurrent Neural Network)**.** *A stochastic* RNN *is a non-Markovian* LVM *where the parameters of the transition and observation models are defined by* RNN*s. A common example is using the conditional Gaussian distribution to define the transition* PDF

$$f_t(x_t \mid x_{1:t-1}) = \mathcal{N}\left(x_t \mid \mu_t(x_{1:t-1}), \Sigma_t(x_{1:t-1})\right),$$

**Algorithm 8:** Iterated auxiliary SMC

Initialize $\rho_1^0, \ldots, \rho_T^0$.
**while** *(stopping criteria not met)* **do**
  Run Algorithm 3 with current parameters $\rho_1^{n-1}, \ldots, \rho_T^{n-1}$. The proposal and weights are given by Eq. (76) and Eq. (77), respectively.
  Update the parameters to $\rho_1^n, \ldots, \rho_T^n$ by solving Eq. (78).
**end**

*where the functions $\mu_t(\cdot), \Sigma_t(\cdot)$ are defined by* RNN*s. This model together with a Gaussian-like definition for the potentials satisfies the criteria for using Algorithm 8. We define $\bar{\psi}_t$ as follows*

$$\bar{\psi}_t(x_{1:t}\,;\,\rho_t) = \exp\left(-\frac{1}{2}x_t^\top \Lambda_t(x_{1:t-1})x_t + \iota_t(x_{1:t-1})^\top x_t + c_t(x_{1:t-1})\right),$$

*where the functions $\Lambda_t(\cdot), \iota_t(\cdot), c_t(\cdot)$ depend on the parameters $\rho_t$—for notational brevity we have not made this dependence explicit. The proposal (cf. Eq.* (76)*) is given by*

$$\begin{aligned} q_t(x_t \,|\, x_{1:t-1}\,;\,\rho_t) &= \mathcal{N}\left(x_t \,|\, \widehat{\mu}_t, \widehat{\Sigma}_t\right), \\ \widehat{\Sigma}_t &= \left(\Sigma_t(x_{1:t-1})^{-1} + \Lambda_t(x_{1:t-1})\right)^{-1}, \\ \widehat{\mu}_t &= \widehat{\Sigma}_t\left(\Sigma_t(x_{1:t-1})^{-1}\mu_t(x_{1:t-1}) + \iota_t(x_{1:t-1})\right), \end{aligned}$$

*and the twisting potential*

$$\psi_{t-1}(x_{1:t-1}\,;\,\rho_t) = \mathbb{E}_{f_t(x_t \,|\, x_{1:t-1})}\left[\bar{\psi}_t(x_{1:t}\,;\,\rho_t)\right] = \sqrt{\frac{\det\left(\widehat{\Sigma}_t\right)}{\det\left(\Sigma_t(x_{1:t-1})\right)}}$$
$$\cdot \exp\left(\frac{1}{2}\widehat{\mu}_t^\top \widehat{\Sigma}_t^{-1}\widehat{\mu}_t - \frac{1}{2}\mu_t(x_{1:t-1})^\top \Sigma_t(x_{1:t-1})^{-1}\mu_t(x_{1:t-1}) + c_t(x_{1:t-1})\right).$$

Twisting the target distribution is an area of active research [Guarniero et al., 2017, Heng et al., 2017, Lindsten et al., 2018]. The iterated auxiliary SMC [Guarniero et al., 2017] makes strong assumptions on the model for easier optimization. End-to-end optimization of a priori independent twisting potentials $\psi_t$ and proposals $q_t$ might lead to even more accurate algorithms.

## 3.A Taylor and Unscented Transforms

In this appendix we detail the EKF- and UKF-based approximations to the distribution $p(x_t, y_t \mid x_{1:t-1})$. We refer to the two approaches as Taylor and Unscented transforms, respectively.

**Taylor Transform** If we assume that $v_t$ and $e_t$ are zero-mean and linearize $a$ around $v_t = 0$ and $c$ around $x_t = a(x_{1:t-1}, 0), e_t = 0$ we get

$$x_t \approx \bar{a} + J_v^a(\bar{a}) v_t, \tag{79a}$$
$$y_t \approx \bar{c} + J_{x_t}^c(\bar{c})(x_t - \bar{a}) + J_{e_t}^c(\bar{c}) e_t, \tag{79b}$$

where $\bar{a} = a(x_{1:t-1}, 0)$, $\bar{c} = c\,((x_{1:t-1}, \bar{a}), 0)$, and $J_{v_t}^a(\bar{a})$ denotes the Jacobian of the function $a$ with respect to $v_t$ evaluated at the point $\bar{a}$. We rewrite Eq. (79):

$$x_t \approx \bar{a} + J_v^a(\bar{a}) v_t, \tag{80a}$$
$$y_t \approx \bar{c} + J_{x_t}^c(\bar{c}) J_v^a(\bar{a}) v_t + J_{e_t}^c(\bar{c}) e_t. \tag{80b}$$

For the approximation in Eq. (80) $x_t, y_t \mid x_{1:t-1}$ is indeed Gaussian, and with $v_t \sim \mathcal{N}(0, Q), e_t \sim \mathcal{N}(0, R)$, we can identify the blocks of $\widehat{\mu}$ and $\widehat{\Sigma}$:

$$\widehat{\mu}_x = \bar{a}, \tag{81a}$$
$$\widehat{\mu}_y = \bar{c}, \tag{81b}$$
$$\widehat{\Sigma}_{xx} = J_v^a(\bar{a}) Q J_v^a(\bar{a})^\top, \tag{81c}$$
$$\widehat{\Sigma}_{xy} = \widehat{\Sigma}_{yx}^\top = J_v^a(\bar{a}) Q J_v^a(\bar{a})^\top J_{x_t}^c(\bar{c})^\top, \tag{81d}$$
$$\widehat{\Sigma}_{yy} = J_{x_t}^c(\bar{c}) J_v^a(\bar{a}) Q J_v^a(\bar{a})^\top J_{x_t}^c(\bar{c})^\top + J_{e_t}^c(\bar{c}) R J_{e_t}^c(\bar{c})^\top. \tag{81e}$$

Using e.g. automatic differentiation we can easily compute the Jacobians needed for the EKF-based proposal. However, we still have to be able to evaluate the densities defined by $a(x_{1:t-1}, v_t)$ and $c(x_{1:t}, e_t)$ for the weight updates $\widetilde{w}_t$. For notational convenience we have omitted the dependence on previous latent states in the expressions above. However, in general because of this dependence we will have to compute the proposals separately for each particle, which includes inverse matrix computations that can be costly if $x_t$ is high-dimensional.

**Unscented Transform** We rewrite Eq. (37) as a single joint random variable, and such that the functions $a, c$ only depend on $x_{1:t-1}, v_t$, and $e_t$,

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} a(x_{1:t-1}, v_t) \\ c\left((x_{1:t-1}, a(x_{1:t-1}, v_t)), e_t\right) \end{pmatrix} =: \tilde{a}(x_{1:t-1}, z_t), \tag{82}$$

where $z_t = (v_t, e_t)^\top$. By approximating the conditional distribution of $(x_t, y_t)^\top$ given $x_{1:t-1}$ as Gaussian, we can again compute the distribution of $x_t \mid x_{1:t-1}, y_t$ for the approximation. We choose sigma points based on the two first moments of $z_t$. The mean and variance of $z_t$ is

$$\mu_z = \mathbb{E}[z_t], \quad \Sigma_z = \mathrm{Var}(z_t) = \sum_{l=1}^{n_x+n_y} \sigma_l^2 \mathrm{u}_l \mathrm{u}_l^\top,$$

where the scalars $\sigma_l$ and vectors $\mathrm{u}_l$ correspond to the singular value decomposition of $\Sigma_z$. We then choose the $2n_x + 2n_y + 1$ sigma points for $z$ as follows,

$$\mathrm{z}^0 = \mu_z, \quad \mathrm{z}^{\pm l} = \mu_z \pm \sigma_l \sqrt{n_x + n_y + \lambda} \cdot \mathrm{u}_l,$$

where $\lambda = \alpha^2(n_x + n_y + \kappa) - n_x - n_y$ [Julier and Uhlmann, 1997]. We will discuss the choice of design parameters $\alpha$ and $\kappa$ below. The mean and variance of $(x_t, y_t)^\top$ is estimated by

$$\mathbb{E}\left[(x_t, y_t)^\top\right] \approx \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix} = \sum_{l=-(n_x+n_y)}^{n_x+n_y} \omega^l \tilde{a}(x_{1:t-1}, \mathrm{z}^l), \tag{83a}$$

$$\begin{aligned} &\mathrm{Var}\left((x_t, y_t)^\top\right) \approx \begin{pmatrix} \widehat{\Sigma}^{xx} & \widehat{\Sigma}^{xy} \\ \widehat{\Sigma}^{yx} & \widehat{\Sigma}^{yy} \end{pmatrix} = \\ &(1 - \alpha^2 + \beta)\left(\tilde{a}(x_{1:t-1}, \mathrm{z}^0) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)\left(\tilde{a}(x_{1:t-1}, \mathrm{z}^0) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)^\top + \\ &\sum_{l=-(n_x+n_y)}^{n_x+n_y} \omega^l \left(\tilde{a}(x_{1:t-1}, \mathrm{z}^l) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)\left(\tilde{a}(x_{1:t-1}, \mathrm{z}^l) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)^\top, \end{aligned} \tag{83b}$$

where the coefficients $\omega^l$ are

$$\omega^0 = \frac{\lambda}{n_x + n_y + \lambda}, \quad \omega^{\pm l} = \frac{1}{2(n_x + n_y + \lambda)},$$

and $\beta$ is another design parameter.

Like in the EKF-based approximation we still need access to the densities corresponding to Eq. (37). Unlike in the EKF case the, UKF-based approximation does not need access to derivatives. However, we need to choose the three design parameters: $\alpha, \beta, \kappa$. Both $\alpha$ and $\kappa$ control the spread of the sigma points, and $\beta$ is related to the distribution of $z_t$ [Julier, 2002]. Typical values of these parameters are $(\alpha, \kappa, \beta) = (10^{-3}, 0, 2)$.

# 4 Random but Proper Weights: Algorithms and Applications

In this chapter we will discuss nesting Monte Carlo (MC) algorithms, using several layers of MC to construct composite algorithms. Usually, this is done to approximate some intractable idealized algorithm. The intractability can stem from e.g. a likelihood or weight that we cannot evaluate in closed form, or a proposal distribution that we cannot sample exactly from. Either way, the key idea of these methods is to introduce additional auxiliary variables to tackle the intractability in the idealized method. The approximate method is constructed in such a way that it retains some of the favorable properties of the idealized MC method that it tries to emulate. We will in this section study particle-based pseudo marginal (PM) methods for several idealized MC methods, ranging from marginal Metropolis-Hastings (MH) [Andrieu et al., 2009, 2010] to distributed SMC [Vergé et al., 2015] and various nested SMC methods [Fearnhead et al., 2010, Chopin et al., 2013, Naesseth et al., 2015].

In the first section below we discuss the unbiasedness property of the SMC normalization constant estimate, a key property underpinning several nested MC methods. Then, in the next section we discuss particle Metropolis-Hastings (PMH). Next, we introduce the concept of *proper weights* and discuss various SMC methods relying on random (but proper) weights. Finally, we introduce a few perspectives on distributed SMC.

## 4.1 The Unbiasedness of the Normalization Constant Estimate

We briefly touched upon one of the key theoretical properties of SMC in Section 2.3, the unbiasedness of the normalization constant estimate $\widehat{Z}_t$. This will be important for several of the PM methods that we discuss in this chapter. We repeat the definition of $\widehat{Z}_t$ again for clarity:

$$\widehat{Z}_t = \prod_{k=1}^{t} \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_k^i. \tag{84}$$

In Proposition 2 we provide a result that formalizes the unbiasedness property. Furthermore, we provide a straightforward self-contained proof of this important property of the SMC algorithm.

**Proposition 2.** *The* SMC *normalization constant estimate* $\widehat{Z}_t$ *is non-negative and unbiased*

$$\mathbb{E}\left[\widehat{Z}_t\right] = Z_t, \quad \widehat{Z}_t \geq 0. \tag{85}$$

*Proof.* See Appendix 4.A. □

Often it will prove useful to think of the $\widehat{Z}_t$ simply as a function of the random seed used to generate all the random variables in the SMC algorithm. Below we formalize this point-of-view, which will be useful for e.g. PMH and nested sequential Monte Carlo (NSMC) in Section 4.2 and Section 4.3, respectively.

**Auxiliary Variables and Random Seeds** The complete set of random variables generated in the SMC algorithm consists of each of the particles $x_t^i$, for $i \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T\}$, and ancestor variables $a_t^i$ for $i \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T-1\}$. Both $x_t^i$ and $a_t^i$ are often generated by sampling base auxiliary random numbers such as Gaussian or uniforms, then the particles and ancestor variables are simply a function of these auxiliary variables. We will denote the collection of all base auxiliary random variables, also referred to as the *random seed*, by $\mathbf{u}$. Each individual particle $x_t^i$ and ancestor $a_t^i$ are then a transformation from the random seed. For example we might have $\mathbf{u} = u_{x,1:T}^{1:N} \cup u_{a,1:T-1}^{1:N}$, where each $u_{x,t}^i, u_{a,t}^i \sim \mathcal{U}(0,1)$. The function that connects $u_{x,t}^i$ to $x_t^i$ and $u_{a,t}^i$ to $a_t^i$ are then the inverse cumulative distribution functions (CDFs) of the proposal $q_t$ and resampling mechanism from Section 2.2.1, respectively. A common tool in PM is to think of the normalization constant estimate in Eq. (84) as a function of all the random variables $\mathbf{u}$, effectively the random seed of the SMC algorithm. Then we can write Eq. (84) as $\widehat{Z}_t(\mathbf{u})$, i.e.

$$\widehat{Z}_t(\mathbf{u}) := \prod_{k=1}^{t} \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_k^i, \tag{86}$$

where the weights $\widetilde{w}_k^i$ are functions of the random seed $\mathbf{u}$. We denote the distribution of the random seed $\mathbf{u}$ by $\tau(\mathbf{u})$. The main result in Proposition 2 can then be written in terms of the random seed $\mathbf{u}$: $\mathbb{E}_{\tau(\mathbf{u})}\left[\widehat{Z}_t(\mathbf{u})\right] = Z_t$.

**Example 12** ($\widehat{Z}_t(\mathbf{u})$ for Example 1)**.** *We revisit our running non-Markovian Gaussian example and consider a standard* SMC *estimator with the prior proposal and no twisting of the target. This means that for* $\mathbf{u} = \varepsilon_{1:T}^{1:N} \cup u_{1:T-1}^{1:N}$*, where* $\varepsilon_t^i \sim \mathcal{N}(0,1)$ *and* $u_t^i \sim \mathcal{U}(0,1)$*, we have*

$$\widehat{Z}_t(\mathbf{u}) = \prod_{k=1}^{t} \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_k^i = \prod_{k=1}^{t} \frac{1}{N} \sum_{i=1}^{N} g_k\left(y_k \,|\, x_{1:k}^i\right),$$

*where* $x_{1:k}^i = (x_{1:k-1}^{a_{k-1}^i}, x_k^i)$*, the ancestor variable* $a_{k-1}^i$ *is defined by the resampling mechanism Section 2.2.1 using* $u_{k-1}^i$*, and*

$$x_k^i = \phi x_{k-1}^{a_{k-1}^i} + \sqrt{q}\varepsilon_k^i.$$

*In Example Code 5 we provide a code snippet, definitions left out for clarity, that illustrates how* $\widehat{Z}_t(\mathbf{u})$ *can be computed. For numerical stability purposes we compute the log-normalization constant estimate,* $\log \widehat{Z}_t(\mathbf{u})$*, instead.*

```
def multinomial_resampling(w, u):
  bins = np.cumsum(w)
    return np.digitize(u,bins).astype(int)
...
for t in range(T):
  if t > 0:
    a=multinomial_resampling(w,u[:,t-1])
    mu = mu[a]
  x[:,t]=phi*x[a,t-1]+np.sqrt(q)*eps[:,t]
  mu = beta*mu + x[:,t]
  logw = norm.logpdf(y[t], mu, np.sqrt(r))
  w = np.exp(logw-logsumexp(logw))
  logZ += logsumexp(logw)-np.log(N)
```

**Example Code 5:** Computing $\log \widehat{Z}_t(\mathbf{u})$ for Example 1 with $\mathbf{u} = \varepsilon_{1:T}^{1:N} \cup u_{1:T-1}^{1:N}$.

## 4.2 Particle Metropolis-Hastings

PMH melds particle-based methods, such as IS and SMC, with Metropolis-Hastings, providing the practitioners with powerful approximate Bayesian inference for a large class of intractable likelihood problems. MH is a MCMC method [Robert

and Casella, 2004] for approximate simulation from a PDF. The typical example is the posterior distribution of unknown model parameters $\theta$ given the observed data $\mathbf{y}$,

$$p(\theta \mid \mathbf{y}) = \frac{p(\theta)p(\mathbf{y} \mid \theta)}{p(\mathbf{y})} = \frac{p(\theta)\int p(\mathbf{x}, \mathbf{y} \mid \theta)\,\mathrm{d}\mathbf{x}}{p(\mathbf{y})}, \tag{87}$$

where here $\mathbf{x}$ are latent (unknown) variables that we wish to integrate out. The likelihood $p(\mathbf{y} \mid \theta) = \int p(\mathbf{x}, \mathbf{y} \mid \theta)\,\mathrm{d}\mathbf{x}$ is often intractable. A common solution is to use data augmentation, sampling instead (approximately) from the posterior distribution of both $\theta$ and $\mathbf{x}$ given the data $\mathbf{y}$, i.e. $p(\theta, \mathbf{x} \mid \mathbf{y})$. Marginal MH, see Algorithm 9, on the other hand generates approximate samples directly from the marginal distribution $p(\theta \mid \mathbf{y})$, provided that the likelihood can be evaluated exactly. The key idea is to generate a Markov chain $\theta_1, \theta_2, \ldots$ with a stationary distribution equal to the target distribution of interest Eq. (87). The accept-reject step in Algorithm 9, accepting the proposed value $\theta'$ as the new iterate with probability $A$, ensures that the stationary distribution is equal to Eq. (87). The acceptance probability

$$A = \min\left(1, \frac{p(\mathbf{y} \mid \theta')\,p(\theta')\,q(\theta^{j-1} \mid \theta')}{p(\mathbf{y} \mid \theta^{j-1})\,p(\theta^{j-1})\,q(\theta' \mid \theta^{j-1})}\right)$$

is the minimum of one and the ratio of the posterior and the proposal evaluated at the proposed value, divided by the same ratio evaluated at the previous iterate. For a more thorough treatment of the MH algorithm see e.g. Robert and Casella [2004].

### 4.2.1 Particle Marginal Metropolis-Hastings

The pseudo marginal method replaces the exact likelihood with an estimate of it, usually constructed by IS or SMC like in Eq. (84). For a fixed value of the parameter $\theta$ we can use the tools from the previous chapters to construct a non-negative and unbiased approximation to $p(\mathbf{y} \mid \theta)$, we will use the notation from (86) and label this estimate $\widehat{Z}(\mathbf{u}, \theta)$. From Proposition 2 we have that

$$\mathbb{E}_{\tau(\mathbf{u})}\left[\widehat{Z}(\mathbf{u}, \theta)\right] = p(\mathbf{y} \mid \theta), \qquad \widehat{Z}(\mathbf{u}, \theta) \geq 0. \tag{88}$$

Simply replacing $p(\mathbf{y} \mid \theta)$ with this unbiased and non-negative approximation, $\widehat{Z}(\mathbf{u}, \theta)$, results in the particle marginal Metropolis-Hastings (PMMH) algorithm

**Algorithm 9:** Metropolis-Hastings

**input :** PDF $p(\mathbf{y}, \theta)$, proposal $q(\theta' \mid \theta)$, initial iterate $\theta^0$, number of iterations $J$.

**output:** Samples $\{\theta^j\}_{j=1}^J$ approximately distributed as $p(\theta \mid \mathbf{y})$.

**for** $j = 1$ **to** $J$ **do**
  Sample $\theta' \sim q(\theta' \mid \theta^{j-1})$
  Compute $A = \min\left(1, \frac{p(\mathbf{y}\mid\theta')p(\theta')q(\theta^{j-1}\mid\theta')}{p(\mathbf{y}\mid\theta^{j-1})p(\theta^{j-1})q(\theta'\mid\theta^{j-1})}\right)$
  Sample $u \sim \mathcal{U}(0,1)$
  **if** $u < A$ **then**
    Set $\theta^j = \theta'$
  **else**
    Set $\theta^j = \theta^{j-1}$
  **end**
**end**

in Algorithm 10. In a practical implementation we do not need to save the entire seed $\mathbf{u}$ for each iteration, it suffices to save the scalar value for our normalization constant estimate $\widehat{Z}(\theta, \mathbf{u})$. We illustrate the MH and PMMH algorithms applied to our running example in Example 13.

**Example 13** (Particle marginal Metropolis-Hastings for Example 1)**.** *We revisit our running non-Markovian Gaussian example. To keep it simple we focus on the* IS *special case, i.e. with* $T = 1$*, which means we are interested in inference for* $\theta = (q, r)$*. We assign a log-normal prior to the variance parameters* $q$ *and* $r$*, i.e.* $\log q \sim \mathcal{N}(0,1)$ *and* $\log r \sim \mathcal{N}(0,1)$*. The corresponding marginal likelihood is then given by* $p(\mathbf{y}\mid\theta) = \mathcal{N}(\mathbf{y}\mid 0, q+r)$*. We use a normal distribution random walk proposal with variance* $0.5$ *for the logarithm of the parameters. In Fig. 7 we can see the kernel density estimate based on the samples of a* $20\,000$ *iterations long run of the two respective algorithms. The two algorithms obtain the same marginal distribution as expected.*

The motivation for why this works is fairly straightforward. We can think of the PMMH algorithm as a standard MH on the extended space of $(\theta, \mathbf{u})$, i.e. adding the random seed as an auxiliary variable. The proposal on this extended space is

$$q(\theta' \mid \theta)\tau(\mathbf{u}'), \tag{89}$$

**Algorithm 10:** Particle marginal Metropolis-Hastings

**input :** Proposals $q(\theta' \mid \theta)$, $\tau(\mathbf{u})$, likelihood approximation $\widehat{Z}(\mathbf{u}, \theta)$, initial iterates $\theta^0, \mathbf{u}^0$, number of iterations $J$.

**output:** Samples $\{\theta^j\}_{j=1}^J$ approximately distributed as $p(\theta \mid \mathbf{y})$.

**for** $j = 1$ **to** $J$ **do**
  Sample $\theta' \sim q(\theta' \mid \theta^{j-1})$, $\mathbf{u}' \sim \tau(\mathbf{u}')$
  Compute $\widehat{Z}(\mathbf{u}', \theta')$
  Compute $A = \min\left(1, \frac{\widehat{Z}(\mathbf{u}', \theta') p(\theta') q(\theta^{j-1} \mid \theta')}{\widehat{Z}(\mathbf{u}^{j-1}, \theta^{j-1}) p(\theta^{j-1}) q(\theta' \mid \theta^{j-1})}\right)$
  Sample $u \sim \mathcal{U}(0, 1)$
  **if** $u < A$ **then**
    Set $\theta^j = \theta'$, $\mathbf{u}^j = \mathbf{u}'$
  **else**
    Set $\theta^j = \theta^{j-1}$, $\mathbf{u}^j = \mathbf{u}^{j-1}$
  **end**
**end**

and the distribution we are trying to draw samples from, i.e. the target distribution, is

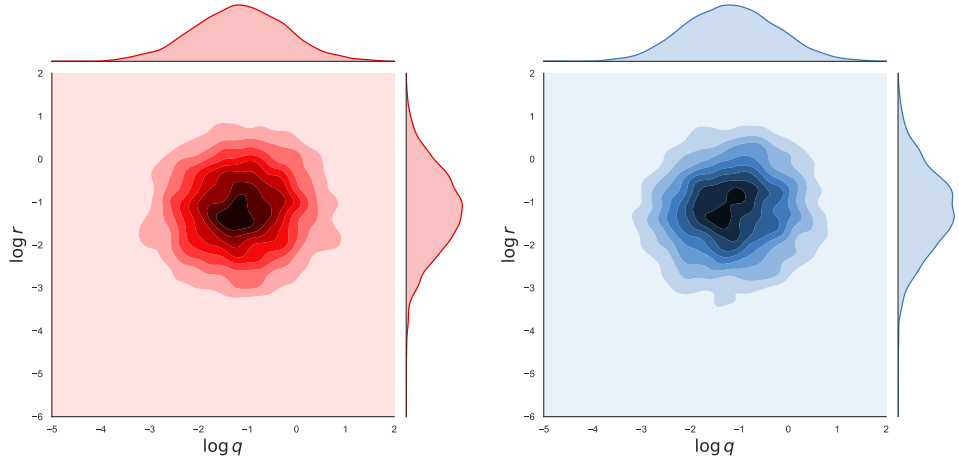
$$\frac{\widehat{Z}(\mathbf{u}, \theta) p(\theta) \tau(\mathbf{u})}{p(\mathbf{y})}, \tag{90}$$

which by the unbiasedness of $\widehat{Z}(\mathbf{u}, \theta)$ has the posterior distribution as a marginal

$$p(\theta \mid \mathbf{y}) = \int \frac{\widehat{Z}(\mathbf{u}, \theta) p(\theta) \tau(\mathbf{u})}{p(\mathbf{y})} \, \mathrm{d}\mathbf{u}.$$

This means that if we can generate samples approximately distributed as the distribution in Eq. (90), we get samples approximately distributed as $p(\theta \mid \mathbf{y})$ as a by-product by considering only the samples for $\theta$.

To show that PMMH is a standard MH on an extended space we first note that the proposal $q(\theta' \mid \theta)\tau(\mathbf{u}')$ is the distribution of the proposed value $(\theta', \mathbf{u}')$. Then studying the corresponding acceptance probability $A$ for the above choice




**(a)** MH **(b)** PMMH with $N = 10$

**Figure 7:** Kernel density estimates of the samples for the parameters $\log q$ and $\log r$ using 20 000 MCMC iterations.

of proposal and target distributions,

$$A = \min\left(1, \frac{\widehat{Z}(\mathbf{u}', \theta')p(\theta')\tau(\mathbf{u}')}{\widehat{Z}(\mathbf{u}^j, \theta^j)p(\theta^j)\tau(\mathbf{u}^j)} \frac{q(\theta^{j-1} \mid \theta')\tau(\mathbf{u}^j)}{q(\theta' \mid \theta^{j-1})\tau(\mathbf{u}')}\right)$$
$$= \min\left(1, \frac{\widehat{Z}(\mathbf{u}', \theta')p(\theta')}{\widehat{Z}(\mathbf{u}^j, \theta^j)p(\theta^j)} \frac{q(\theta^{j-1} \mid \theta')}{q(\theta' \mid \theta^{j-1})}\right),$$

we get exactly the expression from Algorithm 10. These two properties ensure that PMMH is an MH algorithm on the extended space with Eq. (90) as its stationary distribution.

### 4.2.2 Sampling the Latent Variables

If we are interested not just in the parameters but in the latent variables also, we can adapt the particle MH framework with a simple extension of the PMMH algorithm. We can use particle methods to generate approximate samples from the complete target with both parameters and latent variables, $p(\theta, \mathbf{x} \mid \mathbf{y})$. We simply use the final SMC approximation for $p(\mathbf{x} \mid \theta, \mathbf{y})$ that we obtain as a by-product of computing $\widehat{Z}(\mathbf{u}, \theta)$ as a proposal in the MH algorithm. The SMC

approximation to the conditional distribution $\mathbf{x} \mid \theta, \mathbf{y}$ is

$$p(\mathbf{x} \mid \theta, \mathbf{y}) \approx \widehat{p}(\mathbf{x} \mid \mathbf{u}) := \sum_{i=1}^{N} w_T^i \delta_{x_{1:T}^i}(\mathrm{d}x_{1:T}), \tag{91}$$

where we suppress the dependence on the parameters $\theta$ and data $\mathbf{y}$ for clarity. Just like $\widehat{Z}(\mathbf{u}, \theta)$ the conditional distribution estimator is also completely defined by the random seed $\mathbf{u}$. Using the above distribution as our proposal for $\mathbf{x}$ lets us derive the particle Metropolis-Hastings algorithm, see Algorithm 11.

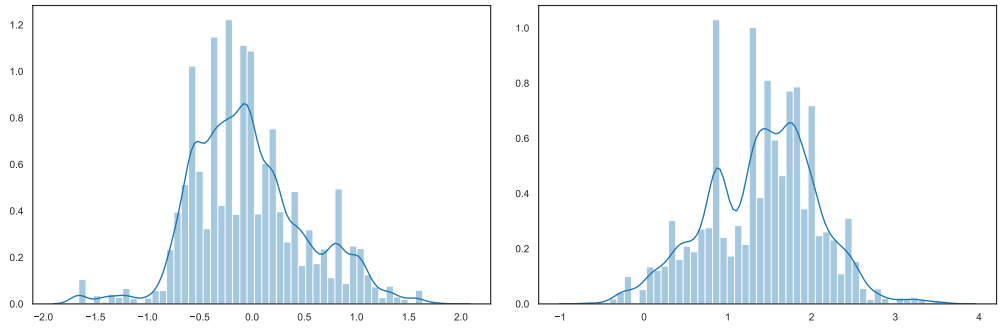
Clearly, this algorithm can be applied for sampling the latent variables also when the model parameters $\theta$ are all known. In this case we simply drop all $\theta$-dependent terms and variables in Algorithm 11. The proposal on the extended space is then

$$\tau(\mathbf{u})\widehat{p}(\mathbf{x} \mid \mathbf{u}), \tag{92}$$

which corresponds to first running the SMC algorithm (sample $\mathbf{u}' \sim \tau(\mathbf{u})$) and then sample one particle trajectory from the resulting SMC approximation (sample $\mathbf{x}' \sim \widehat{p}(\mathbf{x} \mid \mathbf{u})$. Since this is done independently from the current state of the Markov chain, it is referred to as an *independent proposal* and the resulting algorithm is denoted particle independent Metropolis-Hastings (PIMH). The acceptance probability for particle independent Metropolis-Hastings (PIMH) is simply the minimum of one and the ratio of normalization constant estimators, i.e.

$$A = \min\left(1, \frac{\widehat{Z}(\mathbf{u}')}{\widehat{Z}(\mathbf{u}^{j-1})}\right).$$

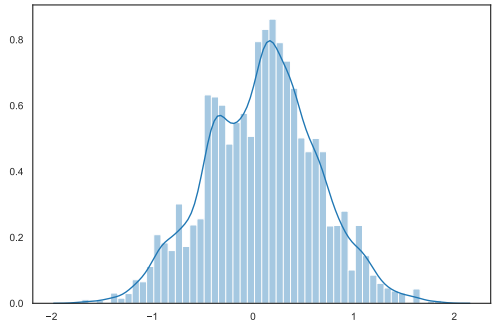
**Example 14** (Particle independent Metropolis-Hastings for Example 1)**.** *We revisit our running non-Markovian Gaussian example. We set* $T = 20$ *and run* PIMH *for* $N = 5$*,* $10$ *and* $20$*. The kernel density estimates of the marginal posterior distributions of* $x_1$ *and* $x_T$ *based on the* $10,000$ *samples generated can be seen in Fig. 8. We can see that, while in theory exact in the limit of infinite number of samples, the approximations for the lower number of particles* $N = 5$*,* $10$*, are distinctly non-Gaussian. It is only when we use a large enough number of particles, e.g.* $N = 20$*, that we obtain more satisfactory results for a finite number of* MCMC *iterations.*



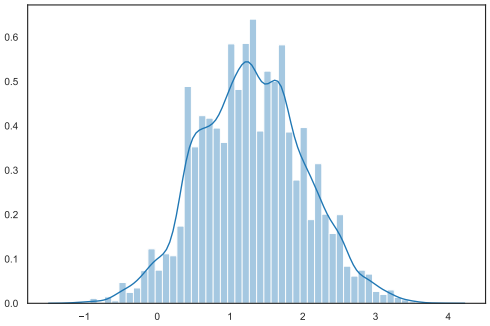
**(a)** $t = 1, N = 5$



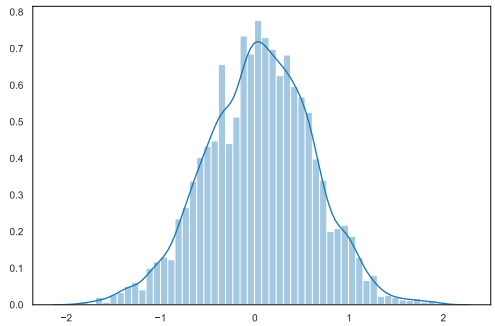
**(b)** $t = 20, N = 5$



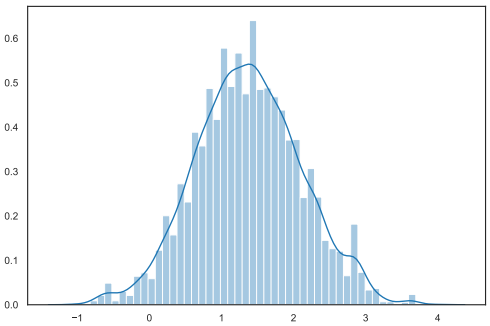
**(c)** $t = 1, N = 10$



**(d)** $t = 20, N = 20$



**(e)** $t = 1, N = 20$



**(f)** $t = 20, N = 20$

**Figure 8:** Illustration of the kernel density estimates of the PIMH marginal posterior approximation of $x_1$ and $x_T$ for$T = 20$ for $N = 5$, 10 and 20.

**Algorithm 11:** Particle Metropolis-Hastings

**input :** Proposals $q(\theta' \mid \theta)$, $\tau(\mathbf{u})$, $\widehat{p}(\mathbf{x} \mid \mathbf{u})$, likelihood approximation $\widehat{Z}(\mathbf{u}, \theta)$, initial iterates $\theta^0, \mathbf{u}^0, \mathbf{x}^0$, number of iterations $J$.

**output:** Samples $\{(\theta^j, \mathbf{x}^j)\}_{j=1}^J$ approximately distributed as $p(\theta, \mathbf{x} \mid \mathbf{y})$.

**for** $j = 1$ **to** $J$ **do**
  Sample $\theta' \sim q(\theta' \mid \theta^{j-1})$, $\mathbf{u}' \sim \tau(\mathbf{u}')$, $\mathbf{x}' \sim \widehat{p}(\mathbf{x}' \mid \mathbf{u}')$
  Compute $\widehat{Z}(\mathbf{u}', \theta')$
  Compute $A = \min\left(1, \frac{\widehat{Z}(\mathbf{u}', \theta') p(\theta') q(\theta^{j-1} \mid \theta')}{\widehat{Z}(\mathbf{u}^{j-1}, \theta^{j-1}) p(\theta^{j-1}) q(\theta' \mid \theta^{j-1})}\right)$
  Sample $u \sim \mathcal{U}(0, 1)$
  **if** $u < A$ **then**
    Set $\theta^j = \theta'$, $\mathbf{u}^j = \mathbf{u}'$, $\mathbf{x}^j = \mathbf{x}'$
  **else**
    Set $\theta^j = \theta^{j-1}$, $\mathbf{u}^j = \mathbf{u}^{j-1}$, $\mathbf{x}^j = \mathbf{x}^{j-1}$
  **end**
**end**

Analogously to PMMH, we can motivate PMH as a standard MH algorithm on an extended space. We focus on the the setting where the model parameters are known and the extended space is given by $(\mathbf{u}, \mathbf{x})$. The proposal on this extended space is

$$\tau(\mathbf{u}) \widehat{p}(\mathbf{x} \mid \mathbf{u}), \tag{93}$$

and the target distribution is

$$\frac{\widehat{Z}(\mathbf{u}) \tau(\mathbf{u}) \widehat{p}(\mathbf{x} \mid \mathbf{u})}{p(\mathbf{y})}. \tag{94}$$

By the unbiasedness property of SMC, i.e. Theorem 1, we have that the posterior distribution $p(\mathbf{x} \mid \mathbf{y})$ is a marginal of Eq. (94)

$$\frac{\int \widehat{Z}(\mathbf{u}) \tau(\mathbf{u}) \widehat{p}(\mathbf{x} \mid \mathbf{u}) \, \mathrm{d}\mathbf{u}}{p(\mathbf{y})} \overset{\mathrm{d}}{=} \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} = p(\mathbf{x} \mid \mathbf{y}),$$

where $\overset{\mathrm{d}}{=}$ denotes equal to in distribution. Studying the corresponding accep-

tance probability $A$ for the above proposal and target distribution we obtain

$$
\begin{aligned}
A &= \min\left(1, \frac{\widehat{Z}(\mathbf{u}')\,\tau(\mathbf{u}')\,\widehat{p}(\mathbf{x}' \mid \mathbf{u}')}{\widehat{Z}(\mathbf{u}^{j-1})\,\tau(\mathbf{u}^{j-1})\,\widehat{p}(\mathbf{x}^{j-1} \mid \mathbf{u}^{j-1})} \frac{\tau(\mathbf{u}^{j-1})\,\widehat{p}(\mathbf{x}^{j-1} \mid \mathbf{u}^{j-1})}{\tau(\mathbf{u}')\,\widehat{p}(\mathbf{x}' \mid \mathbf{u}')}\right) \\
&= \min\left(1, \frac{\widehat{Z}(\mathbf{u}')}{\widehat{Z}(\mathbf{u}^{j-1})}\right),
\end{aligned}
$$

which is identical to the expression from Algorithm 11 (when the model parameters are known).

### 4.2.3 Correlated Pseudo-Marginal Methods

One of the main limitations of the PM methods that we have covered so far is that they require that the normalization constant estimate $\widehat{Z}$ is sufficiently accurate to achieve good performance. This requirement means we have to focus much of the computational effort on the SMC algorithm by using a high number of particles $N$, rather than generating more samples from the posterior $p(\theta \mid \mathbf{y})$. By studying the PMMH algorithm we can see that the proposed parameter $\theta'$ at each iteration is *correlated* with the previous parameter $\theta^{j-1}$, encouraging local exploration and higher acceptance rates. However, this is not true for the random seed $\mathbf{u}$ that is generated independently at each iteration. Introducing correlation also for this random seed, i.e. replacing the independent proposal $\tau(\mathbf{u})$ with a correlated proposal $\tau(\mathbf{u}' \mid \mathbf{u})$, can allow us to effectively lower the computational cost of the SMC estimate through choosing a lower number of particles $N$. Since the dimension of $\mathbf{u}$ typically increases linearly with $N$, we need a proposal distribution which scales favorably to high dimensions. Examples include preconditioned Crank-Nicolson [Deligiannidis et al., 2018, Dahlin et al., 2015], elliptical slice sampling [Murray and Graham, 2016] and Hamiltonian Monte Carlo [Lindsten and Doucet, 2016].

## 4.3 Proper Weights and Sequential Monte Carlo

So far in this manuscript we have focused on SMC algorithms where the particles are sampled exactly from the proposal $q_t$ and where the weights $\widetilde{w}_t$ are deterministic functions of the particles. We can in fact relax both of these restrictions significantly by considering the weights and particles as joint random variables that satisfy a *proper weighting* condition. This will lead us to powerful

algorithms such as the random-weights particle filter [Fearnhead et al., 2010], SMC$^2$ [Chopin et al., 2013], and NSMC methods [Naesseth et al., 2015, 2016].

To formally justify these algorithms we can make use of the concept of proper weighting. This is a property on the expected value of the weights and samples when applied to functions.

**Definition 4.1** (Proper Weighting)**.** We say the (random) pair $(x, \widetilde{w})$ are *properly weighted* for the (unnormalized) distribution $\widetilde{\gamma}$ if $\widetilde{w} \geq 0$ a.s. and for all measurable functions $h$

$$\mathbb{E}[h(x)\widetilde{w}(x)] = \mathscr{C} \int h(x)\widetilde{\gamma}(x)\,\mathrm{d}x, \tag{95}$$

for some positive constant $\mathscr{C} > 0$ that is independent of $x$ and $\widetilde{w}$.

This is highly related to the unbiasedness property of the SMC method that we saw already in Theorem 1. However, while the basic SMC has deterministic weights when conditioned on the samples, here we allow also for random weights within the confines of Definition 4.1.

We will denote the distribution of the random pair $(\widetilde{w}, x)$ by $Q(\mathrm{d}\widetilde{w}, \mathrm{d}x)$. By simulating from $Q$ independently

$$(\widetilde{w}^i, x^i) \overset{\text{iid}}{\sim} Q(\mathrm{d}\widetilde{w}, \mathrm{d}x),\ i = 1, \ldots, N,$$

we can construct an estimate of $\mathbb{E}_\gamma[h(x)]$ that satisfies the law of large numbers,

$$\mathbb{E}_\gamma[h(x)] \approx \frac{\sum_{i=1}^N \widetilde{w}^i h(x^i)}{\sum_{j=1}^N \widetilde{w}^j} = \frac{\frac{1}{N}\sum_{i=1}^N \widetilde{w}^i h(x^i)}{\frac{1}{N}\sum_{j=1}^N \widetilde{w}^j}. \tag{96}$$

Proper weighting ensures that the numerator converges to $Z\mathscr{C}\mathbb{E}_\gamma[h(x)]$ and the denominator to $Z\mathscr{C}$, and thus the right hand side converges to the true expectation $\mathbb{E}_\gamma[h(x)]$.

We have already seen one example of a properly weighted IS method in Section 3.1, namely nested IS (Algorithm 5). Because standard IS is a special case of nested IS it follows that also standard IS leads to weight-sample pairs that are properly weighted. Below we will see how this can be used to motivate more powerful algorithms for approximate inference.

### 4.3.1 Random Weights Sequential Monte Carlo

Random weights SMC and IS focuses on solving the problem with intractable weights. Because we pick the proposal ourselves it is often easy to sample from, whereas the target distribution depends largely on the model we want to study and can lead to intractable importance weights. A concrete example of this is partially observed stochastic differential equations (SDEs) as studied by e.g. Beskos et al. [2006], Fearnhead et al. [2010]. The corresponding target distribution, and thus the weights, contains an exponentiated integral that is intractable. If we replace the intractable weights in the standard SMC framework, $\widetilde{w}(x)$, with (non-negative) unbiased approximations of them, $\widehat{\widetilde{w}}(x)$, we obtain random weights SMC. While we focus the exposition on the basic IS case, the result and method follows for SMC when we have unbiased estimates of the weights conditional on previous samples. It is straightforward to prove that the random pair $(\widehat{\widetilde{w}}(x), x)$ are properly weighted for $\widetilde{\gamma}$,

$$\mathbb{E}\left[\widehat{\widetilde{w}}(x)h(x)\right] = \mathbb{E}\left[\mathbb{E}\left[\widehat{\widetilde{w}}(x) \mid x\right] \cdot h(x)\right] = \mathbb{E}\left[\widetilde{w}(x)h(x)\right] = \int h(x)\widetilde{\gamma}(x)\,\mathrm{d}x. \tag{97}$$

A simplified version of the target distribution in an SDE has the following form of its intractable part

$$\exp\left(\Phi(x)\right), \tag{98}$$

where $\Phi(x)$ is an intractable functional, often an integral. We assume that we can generate unbiased approximations $\widehat{\Phi}(x)$ of the functional for a given value of $x$. Using the *Poisson estimator* [Wagner, 1987] we can obtain an unbiased estimate of the exponentiated functional in Eq. (98).

$$\begin{aligned}\exp\left(\Phi(x)\right) &= \sum_{k=0}^{\infty} \frac{1}{k!} \left(\Phi(x)\right)^k = e^{\lambda} \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{\Phi(x)}{\lambda}\right)^k \lambda^k e^{-\lambda} \\ &= e^{\lambda} \mathbb{E}\left[\prod_{j=1}^{\kappa} \frac{\Phi(x)}{\lambda}\right],\end{aligned} \tag{99}$$

where the expectation is with respect to $\kappa \sim \text{Poisson}(\lambda)$. A MC estimate of Eq. (99) leads to the Poisson estimator

$$e^{\lambda} \mathbb{E}\left[\prod_{j=1}^{\kappa} \frac{\Phi(x)}{\lambda}\right] \approx e^{\lambda} \prod_{j=1}^{\kappa} \frac{\widehat{\Phi}^j(x)}{\lambda}, \tag{100}$$

**Algorithm 12:** Random Weights Sequential Monte Carlo

**input :** Proposals $Q_t$ that generate sample pairs properly weighted for $\widetilde{\gamma}_t$, number of samples $N$.

**output:** Samples and weights $\left\{\left(x_{1:t}^i, w_t^i\right)\right\}_{i=1}^N$ approximating $\gamma_t$, for $t = 1, \ldots, T$.

**for** $t = 1$ **to** $T$ **do**
  **for** $i = 1$ **to** $N$ **do**
    Sample $(\widetilde{w}_t^i, x_{1:t}^i) \sim \widehat{\gamma}_{t-1}(x_{1:t-1}) Q_t(\cdot, \cdot \mid x_{1:t-1})$
  **end**
  Set $w_t^i = \frac{\widetilde{w}_t^i}{\sum_j \widetilde{w}_t^j}$, for $i = 1, \ldots, N$
**end**

where $\widehat{\Phi}^j(x)$ are iid unbiased estimators of $\Phi(x)$. While Eq. (100) is unbiased, it is not necessarily non-negative and might be unsuitable for weights estimation in an SMC algorithm. With proper modification to the estimate we can sometimes ensure that the estimate is non-negative [Fearnhead et al., 2010, Jacob and Thiery, 2015]. Replacing the intractable part in the weights with its unbiased estimate, like Eq. (100) in the SDE case, results in the random weights SMC algorithm Algorithm 12. The samples $x_t^i$ are generated by a standard user chosen proposal $q_t(x_t \mid x_{1:t-1})$ just like in standard SMC.

Rather than interpreting random weights SMC as a properly weighted SMC algorithm, we can instead motivate it as a standard SMC on an extended space.

### 4.3.2 Nested Sequential Monte Carlo

We already briefly touched on NSMC in Section 3.1, illustrating how a nested IS sampler can be used to approximate the locally optimal proposal distribution. Here we will instead consider the case when the nested sampler is itself SMC rather than IS. This usually allows the user to derive more efficient proposal approximations, pushing the boundaries of the type of problems that SMC methods can be effectively applied to.

NSMC is specifically targeting models where the latent state $x_t$ is of dimension $n_x > 1$ (typically in the order of 10's to 100's). The goal of NSMC is to approximate the locally optimal proposal distribution by running a separate internal (or nested) SMC algorithm over the components of $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,n_x})$

for each sample we need. At first this might seem wasteful of our computational resources, running a nested SMC sampler with $M$ internal particles for each particle in the outer algorithm results in $\mathcal{O}(NM)$ operations. However, for many models it results in much more accurate estimate than, for instance, a corresponding standard SMC algorithm with $NM$ particles and the prior proposal.

Unlike in the random weights SMC, NSMC changes not only the way we compute weights, but also the way we generate the samples $x_t^i$. The key idea is to construct a SMC approximation to the locally optimal proposal $q_t^\star(x_t \mid x_{1:t-1})$. For each particle $x_t^i$ we want to simulate we construct an SMC approximation to $q_t^\star$

$$\widehat{q}_t^\star(x_t \mid x_{1:t-1}^i) = \sum_{j=1}^{M} \frac{\widetilde{\nu}_{n_x}^{j,i}}{\sum_l \widetilde{\nu}_{n_x}^{l,i}} \delta_{\bar{x}_{1:n_x}^{j,i}}(x_t), \tag{101}$$

where $(\widetilde{\nu}_{n_x}^{j,i}, \bar{x}_{1:n_x}^{j,i})$ are properly weighted for $q_t^\star(x_t \mid x_{1:t-1}^i)$. We assume that Eq. (101) is constructed using a nested SMC sampler on the components of $x_t$; first targeting $x_{t,1}$, then $x_{t,1:2}$, etc, where the final target is $q_t^\star(x_{t,1:n_x} \mid x_{1:t-1}^i)$. With this approximation NSMC replaces the sampling step Eq. (29) by simulating from Eq. (101), and the weighting step Eq. (30) by

$$\widetilde{w}_t^i = \prod_{k=1}^{n_x} \frac{1}{M} \sum_{j=1}^{M} \widetilde{\nu}_k^{j,i}, \tag{102}$$

where $\widetilde{\nu}_k^{j,i}$ are the corresponding weights in the nested SMC sampler. We summarize NSMC in Algorithm 13 and give examples below of ways to construct $\widehat{q}_t^\star(x_t \mid x_{1:t-1}^i)$.

**Example 15** (Nested sequential Monte Carlo for Example 1)**.** *We revisit our running non-Markovian Gaussian example. Because $x_t$ is scalar for this example, we make a straightforward extension to the model to illustrate* NSMC*. We replicate the model from Example 1 $n_x$ times, i.e.*

$$f_t(x_t \mid x_{t-1}) = \prod_{k=1}^{n_x} f(x_{t,k} \mid x_{t-1,k}) = \prod_{k=1}^{n_x} \mathcal{N}(x_{t,k} \mid \phi x_{t-1,k}, q),$$
$$g_t(y_t \mid x_{1:t}) = \prod_{k=1}^{n_x} g(y_{t,k} \mid x_{1:t,k}) = \prod_{k=1}^{n_x} \mathcal{N}\left(y_{t,k} \mid \sum_{l=1}^{t} \beta^{t-l} x_{l,k}, r\right).$$

**Algorithm 13:** Nested sequential Monte Carlo (NSMC)

**input :** Unnormalized target distributions $\widetilde{\gamma}_t$, nested SMC sampler targeting $q_t^\star(x_t \mid x_{1:t-1})$, number of samples $N$ and nested samples $M$.

**output:** Samples and weights $\left\{\left(x_{1:t}^i, w_t^i\right)\right\}_{i=1}^N$ approximating $\gamma_t$, for $t = 1, \ldots, T$.

**for** $t = 1$ **to** $T$ **do**

  **for** $i = 1$ **to** $N$ **do**

    Sample $x_{1:t}^i \sim \widehat{\gamma}_{t-1}(x_{1:t-1})\widehat{q}_t^\star(x_t \mid x_{1:t-1})$ (see Eq. (101))

    Set $\widetilde{w}_t^i = \prod_{k=1}^{n_x} \frac{1}{M} \sum_{j=1}^M \widetilde{\nu}_k^{j,i}$ (see Eq. (102))

  **end**

  Set $w_t^i = \frac{\widetilde{w}_t^i}{\sum_j \widetilde{w}_t^j}$, for $i = 1, \ldots, N$

**end**

*To construct $\widehat{q}_t^\star(x_t \mid x_{1:t-1}^i)$ we run an internal* SMC *sampler with $M$ samples targeting*

$$\eta_l(x_{t,1:l} \mid x_{1:t-1}) \propto \widetilde{\eta}_l(x_{t,1:l} \mid x_{1:t-1}) := \prod_{k=1}^{l} f(x_{t,k} \mid x_{t-1,k}) g(y_{t,k} \mid x_{1:t,k}),$$

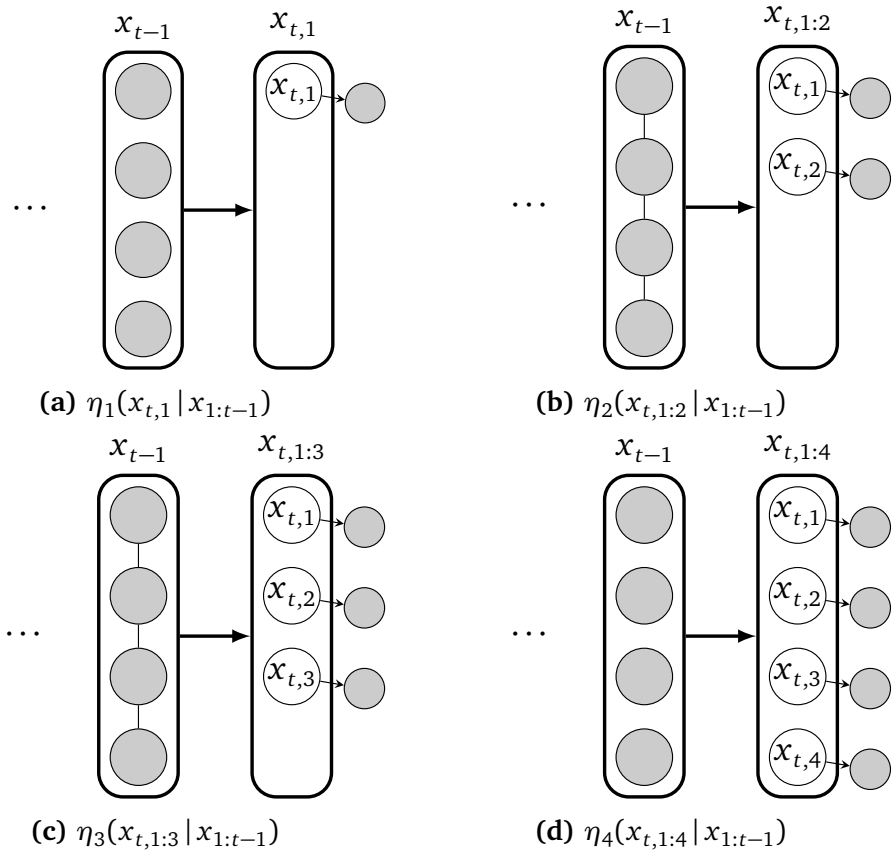
*for $l = 1, \ldots, n_x$. We denote the particles and weights for the nested* SMC *by $\bar{x}_{1:l}^{j,i}$ and $\widetilde{\nu}_l^{j,i}$, respectively. A straightforward proposal for the above target distribution is $\bar{x}_{1:l}^{j,i} \sim \widehat{\eta}_{l-1}(x_{t,1:l-1} \mid x_{1:t-1}^i) f(x_{t,l} \mid x_{t-1,l}^i)$, i.e. using the prior proposal. We have that $\widehat{\eta}_l$ is*

$$\widehat{\eta}_l(x_{t,1:l} \mid x_{1:t-1}^i) = \sum_{j=1}^M \frac{\widetilde{\nu}_l^{j,i}}{\sum_m \widetilde{\nu}_l^{m,i}} \delta_{\bar{x}_{1:l}^{j,i}}(x_{t,1:l}),$$

*where the weights are given by*

$$\widetilde{\nu}_l^{j,i} = g(y_{t,l} \mid (x_{1:t-1,k}^i, \bar{x}_l^{j,i})) = \mathcal{N}\left(y_{t,l} \mid \bar{x}_l^{j,i} + \sum_{m=1}^{t-1} \beta^{t-m} x_{m,l}^i, r\right).$$

*We obtain the approximation to the locally optimal proposal $\widehat{q}_t^\star(x_t \mid x_{1:t-1}^i) \equiv \widehat{\eta}_{n_x}(x_{t,1:n_x} \mid x_{1:t-1}^i)$. This defines the* NSMC *method in Algorithm 13 for the multivariate extension to our running example. We illustrate the nested target distributions in Fig. 9.*




**(a)** $\eta_1(x_{t,1} \mid x_{1:t-1})$ **(b)** $\eta_2(x_{t,1:2} \mid x_{1:t-1})$

**(c)** $\eta_3(x_{t,1:3} \mid x_{1:t-1})$ **(d)** $\eta_4(x_{t,1:4} \mid x_{1:t-1})$

**Figure 9:** Illustration of NSMC for the running example with $n_x = 4$. Note that the each variable $x_{t,l}$ depends on the complete sequence of variables from the previous steps $x_{1:t-1}$.

*Just like in the standard* SMC *sampler, there is room for improvement in the nested* SMC *by e.g. choosing a better internal proposal distributions and target distributions under the proper weighting constraint. It is possible to further improve performance by correlating samples through a combination of fully adapted* SMC *and a forward-filtering backward-sampling procedure as discussed by Naesseth et al. [2015, 2016].*

## 4.4 Distributed Sequential Monte Carlo

Developing computational methods that can take advantage of parallel and multi-core computing architectures is important for efficient approximate inference algorithms. We will discuss independent importance weighted SMC samplers as well as the island particle filter [Vergé et al., 2015], a method for distributing the computation and thus speeding up inference.

### 4.4.1 Importance Weighted Independent SMC Samplers

Assume that we have access to $M$ computing nodes. These could correspond to different machines, different cores, or even $M$ consecutive runs on the same machine. A (very) simple way of distributing computation across these nodes is to run separate independent SMC samplers on each node. Using the notation from above, let $\mathbf{u}^m \sim \tau(\mathbf{u})$ denote all the random variables generated during the run of the SMC sampler in node $m \in \{1, \ldots, M\}$. We then get $m$ independent approximations of the target $\gamma_T$ given by

$$\widehat{\gamma}_T(x_{1:T} \mid \mathbf{u}^m) = \sum_{i=1}^{N} w_T^{m,i} \delta_{x_{1:T}^{m,i}}(x_{1:T}),$$

where we have assumed that each node uses the same number of particles $N$ and the same SMC algorithm. Assume that we want to compute the expected value $\mathbb{E}_{\gamma_T}[h(x_{1:T})]$ of some test function $h$. Each node provides an estimate,

$$\widehat{h}(\mathbf{u}^m) := \sum_{i=1}^{N} w_T^{m,i} h(x_{1:T}^{m,i}), \tag{103}$$

and since these estimates are independent and identically distributed a natural approach is to take a plain average $\frac{1}{M}\sum_{m=1}^{M} \widehat{h}(\mathbf{u}^m)$ as the final estimate. However, since each SMC estimate is biased for finite $N$, the resulting average will also be biased regardless of the number of nodes $M$ used in the computation. A better approach can therefore be to weight the independent SMC samplers according to their respective normalizing constant estimates:

$$\widehat{h}_{\text{aggr}} := \sum_{m=1}^{M} \frac{\widehat{Z}_T(\mathbf{u}^m)}{\sum_n \widehat{Z}_T(\mathbf{u}^n)} \widehat{h}(\mathbf{u}^m). \tag{104}$$

This aggregated estimator is obtained by viewing the SMC sampler in each node as a proposal for an outer-level importance sampler. Specifically, the target distribution for this importance sampler is given by

$$\Pi(\mathbf{u}) := \frac{\widehat{Z}_T(\mathbf{u})\tau(\mathbf{u})}{Z_T},$$

and the proposal is given by $\tau(\mathbf{u})$, resulting in weights $\Pi(\mathbf{u})/\tau(\mathbf{u}) \propto \widehat{Z}_T(\mathbf{u})$. By computations analogous to those in Section 4.2 it holds that $\mathbb{E}_\Pi[\widehat{h}(\mathbf{u})] = \mathbb{E}_\gamma[h(x_{1:T})]$. Consequently, by a standard importance sampling argument it holds that Eq. (104) is a consistent estimate of $\mathbb{E}_\gamma[h(x_{1:T})]$ as $M$ becomes large, regardless of the number $N$ of particles used in each SMC sampler.

It is not expected that Eq. (104) will be a better estimate than what we would obtain from a single SMC sampler with a total of $NM$ particles. However, as pointed out above, the computation of the estimators Eq. (104) can be distributed, or even performed sequentially on the same machine in situations when memory constraints limit the number of particles than can be used.

### 4.4.2 The Island Particle Filter

In the preceding section we considered importance sampling of SMC estimators. It is possible to take this one step further and develop SMC sampling of SMC estimators. This is the idea behind the island particle filter by Vergé et al. [2015].

Similarly to above, the island particle filter consists of $M$ nodes, each running an SMC sampler with $N$ particles. However, these $M$ samplers are allowed to interact at each iteration according to the standard resampling, propagation, and weighting steps of SMC. We provide pseudo-code in Algorithm 14, where we assume that resampling at the outer level only occurs when the effective sample size (ESS) drops below some threshold (see Section 2.2.2). This type of adaptive resampling is recommended for a distributed implementation of the island particle filter. Indeed, resampling at the outer level at each iteration would result in very high communication costs, since it involves copying complete SMC states (particles and weights) from node to node. Furthermore, we can control the outer level ESS by increasing $N$, thereby ensuring that outer level resampling happens rarely when done adaptively.

The island particle filter fits into the standard SMC framework. To make this explicit we start by writing $\mathbf{u} = \mathbf{u}_{1:T}$ where $\mathbf{u}_t$ corresponds to all the random

**Algorithm 14:** The island particle filter

**input :** Unnormalized target distributions $\widetilde{\gamma}_t$, proposals $q_t$, number of nodes $M$ and number of samples per node $N$.

**output:** Samples and weights $\left\{\left(x_{1:t}^{m,i}, w_t^{m,i}\right)\right\}_{i=1}^{N}$ for $m = 1, \ldots, M$, and island-level weights $\{\Omega_t^m\}_{m=1}^{M}$, for $t = 1, \ldots, T$.

**for** $m = 1$ **to** $M$ **do**

  `// Initialize inner SMC samplers`

  Sample $x_1^{m,i} \sim q_1(x_1)$ and set $\widetilde{w}_1^{m,i} = \frac{\widetilde{\gamma}_1(x_1^{m,i})}{q_1(x_1^{m,i})}$ for $i = 1, \ldots, N$.

  Set $w_1^{m,i} = \frac{\widetilde{w}_1^{m,i}}{\sum_j \widetilde{w}_1^{m,j}}$ for $i = 1, \ldots, N$.

  `// Initialize outer SMC sampler`

  *Identify:* $\mathbf{u}_1^m = \{x_1^{m,i}, \widetilde{w}_1^{m,i}\}_{i=1}^{N}$.

  Set $\widetilde{\Omega}_1^m = \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_1^{m,i}$.

**end**

Set $\Omega_1^m = \frac{\widetilde{\Omega}_1^m}{\sum_n \widetilde{\Omega}_1^n}$ for $m = 1, \ldots, M$

**for** $t = 2$ **to** $T$ **do**

  Compute $\mathrm{ESS}_{t-1}^{\text{island}} = \left[\sum_{m=1}^{M} \left(\Omega_{t-1}^m\right)^2\right]^{-1}$.

  **if** $\mathrm{ESS}_{t-1}^{\text{island}} <$ threshold **then**

    `// Resample islands (overload notation)`

    $\mathbf{u}_{1:t-1}^{1:M} \sim \text{Resampling}(\mathbf{u}_{1:t-1}^{1:M}, \Omega_{t-1}^{1:M})$.

    $\Omega_{t-1}^m \leftarrow \frac{1}{M}$, $m = 1, \ldots, M$.

  **end**

  **for** $m = 1$ **to** $M$ **do**

    `// Update inner SMC samplers`

    Sample $x_{1:t}^{m,i} \sim \widehat{\gamma}_{t-1}(x_{1:t-1} \mid \mathbf{u}_{1:t-1}^m) q_t(x_t \mid x_{1:t-1})$, for $i = 1, \ldots, N$.

    Set $\widetilde{w}_t^{m,i} = \frac{\widetilde{\gamma}_t(x_{1:t}^{m,i})}{\widetilde{\gamma}_{t-1}(x_{1:t-1}^{m,i}) q_t(x_t^{m,i} \mid x_{1:t-1}^{m,i})}$, for $i = 1, \ldots, N$.

    Set $w_t^{m,i} = \frac{\widetilde{w}_t^{m,i}}{\sum_j \widetilde{w}_t^{m,j}}$, for $i = 1, \ldots, N$.

    `// Update outer SMC sampler`

    *Identify:* $\mathbf{u}_{1:t}^m = \{x_{1:t}^{m,i}, \widetilde{w}_t^{m,i}\}_{i=1}^{N}$.

    Set $\widetilde{\Omega}_t^m = \frac{\Omega_{t-1}^m}{1/M} \cdot \left[\frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_t^{m,i}\right]$.

  **end**

  Set $\Omega_t^m = \frac{\widetilde{\Omega}_t^m}{\sum_n \widetilde{\Omega}_t^n}$ for $m = 1, \ldots, M$.

**end**

variables generated *at iteration* $t$ of an (internal) SMC sampler. By making the (non restrictive) assumption that these auxiliary variables are drawn independently at each iteration we can write

$$\tau(\mathbf{u}) = \prod_{t=1}^{T} \tau_t(\mathbf{u}_t).$$

The island particle filter is then a standard SMC sampler with $M$ particles, targeting the sequence of distributions

$$\frac{\widehat{Z}_t(\mathbf{u}_{1:t})\left[\prod_{k=1}^{t} \tau_k(\mathbf{u}_k)\right]}{Z_t}$$

and using $\tau_t(\mathbf{u}_t)$ as a proposal distribution at iteration $t$.

If we denote the outer level (island level) unnormalized SMC weights by $\{\widetilde{\Omega}_t^m\}_{m=1}^M$ we get

$$\widetilde{\Omega}_t^m = \frac{\widehat{Z}_t(\mathbf{u}_{1:t}^m)}{\widehat{Z}_{t-1}(\mathbf{u}_{1:t}^m)} = \frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_t^{m,i}$$

at iterations when the islands are resampled, and

$$\widetilde{\Omega}_t^m = \frac{\Omega_{t-1}^m}{1/M} \cdot \left[\frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_t^{m,i}\right],$$

at iterations when the outer level resampling is omitted; see Eq. (34). For implementation purposes we can use the latter expression at all iterations if we overwrite $\Omega_{t-1}^m \leftarrow 1/M$ whenever the islands are resampled. In the above expressions, $\{\widetilde{w}_t^{m,i}\}_{i=1}^M$ denote the unnormalized (inner level) SMC weights at node $M$.

The derivation above is based on viewing the complete random seed as an auxiliary variable. In practice it is usually more convenient to simply keep track of the resulting particle trajectories and weights, which can be computed deterministically given $\mathbf{u}_{1:t}$. Thus, in Algorithm 14 we have, by slight abuse of notation, identified $\mathbf{u}_{1:t}$ with $\left\{\left(x_{1:t}^i, \widetilde{w}_t^i\right)\right\}_{i=1}^N$.

## 4.A Proof of Unbiasedness

For clarity we prove the unbiasedness of the normalizing constant estimate for the version of SMC presented in Algorithm 3, which involves multinomial resampling at each iteration of the algorithm. We note, however, that the property holds more generally, for instance when using low-variance and adaptive resampling. The proof presented here can be straightforwardly extended to such cases.

The distribution of all random variables generated by the SMC method in Algorithm 3 is given by

$$\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right) = \prod_{i=1}^{N} q_1(x_1^i) \cdot \prod_{t=2}^{T} \prod_{i=1}^{N} \left[ w_{t-1}^{a_{t-1}^i} q_t\left(x_t^i \mid x_{1:t-1}^{a_{t-1}^i}\right)\right], \tag{105}$$

where the particles are $x_{1:T}^{1:N} = \bigcup_{t=1}^{T} \left\{x_t^i\right\}_{i=1}^{N}$, and the ancestor variables from the resampling step are $a_{1:T-1}^{1:N} = \bigcup_{t=1}^{T-1} \left\{a_t^i\right\}_{i=1}^{N}$. Assuming that $\widetilde{w}_t^i \geq 0$ for all $i$ and $t$, we have that $\widehat{Z}_T \geq 0$. We are now going to prove that

$$\mathbb{E}_{\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right)}\left[\frac{\widehat{Z}_T}{Z_T}\right] = 1, \tag{106}$$

from which the result follows. Note that the results holds for any value of $T$. To prove the result we introduce another set of variables $b_T^i = i$ and $b_t^i = a_t^{b_{t+1}^i}$, for $t = 1, \ldots, T-1$. The notation describes the ancestor index at iteration $t$ for the final particle $x_{1:T}^i$. This means that we can write $x_{1:T}^i = x_{1:T}^{b_{1:T}^i} = (x_1^{b_1^i}, \ldots, x_T^{b_T^i})$.

Using this notation we rewrite the integrand in Eq. (106),

$$
\begin{aligned}
&\frac{\widehat{Z}_T}{Z_T}\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right) \\
&= \frac{1}{Z_T}\prod_{t=1}^{T}\left[\frac{1}{N}\sum_{i=1}^{N}\widetilde{w}_t^i\right]\cdot\prod_{i=1}^{N}q_1(x_1^i)\cdot\prod_{t=2}^{T}\prod_{i=1}^{N}\left[\frac{\widetilde{w}_{t-1}^{a_{t-1}^i}}{\sum_{j=1}^{N}\widetilde{w}_{t-1}^j}q_t\left(x_t^i \mid x_{1:t-1}^{a_{t-1}^i}\right)\right] \\
&= \frac{1}{N^T Z_T}\sum_{i=1}^{N}\Bigg[\widetilde{w}_1^{b_1^i}q_1(x_1^{b_1^i})\prod_{t=2}^{T}\widetilde{w}_t^{b_t^i}q_t(x_t^{b_t^i} \mid x_{1:t-1}^{b_{t-1}^i}) \\
&\qquad\qquad \cdot\prod_{j\neq b_1^i}q_1(x_1^j)\cdot\prod_{t=2}^{T}\prod_{j\neq b_{t-1}^i}\left[w_{t-1}^{a_{t-1}^j}q_t\left(x_t^j \mid x_{1:t-1}^{a_{t-1}^j}\right)\right]\Bigg] \\
&= \frac{1}{N^T}\sum_{i=1}^{N}\left[\frac{\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}^i})}{Z_T}\cdot\prod_{j\neq b_1^i}q_1(x_1^j)\cdot\prod_{t=2}^{T}\prod_{j\neq b_{t-1}^i}\left[w_{t-1}^{a_{t-1}^j}q_t\left(x_t^j \mid x_{1:t-1}^{a_{t-1}^j}\right)\right]\right],
\end{aligned}
$$

where the first equality is the definition, the second equality separates the dependence of the particle $x_{1:T}^i = x_{1:T}^{b_{1:T}^i}$ from the rest, and the third equality simplifies the product between weights and proposals for particle $x_{1:T}^i = x_{1:T}^{b_{1:T}^i}$.

Now, using Eq. (106) and the above rewrite of the integrand we obtain

$$
\begin{aligned}
&\frac{1}{N^T}\sum_{a_{1:T-1}^{1:N}}\int\sum_{i=1}^{N}\left[\frac{\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}^i})}{Z_T}\cdot\prod_{j\neq b_1^i}q_1(x_1^j)\cdot\prod_{t=2}^{T}\prod_{j\neq b_{t-1}^i}\left[w_{t-1}^{a_{t-1}^j}q_t\left(x_t^j \mid x_{1:t-1}^{a_{t-1}^j}\right)\right]\right]\mathrm{d}x_{1:T}^{1:N} \\
&= \frac{1}{N^{T-1}}\sum_{a_{1:T-1}^{1:N}}\int\frac{\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}^1})}{Z_T}\cdot\prod_{j\neq b_1^1}q_1(x_1^j)\cdot\prod_{t=2}^{T}\prod_{j\neq b_{t-1}^1}\left[w_{t-1}^{a_{t-1}^j}q_t\left(x_t^j \mid x_{1:t-1}^{a_{t-1}^j}\right)\right]\mathrm{d}x_{1:T}^{1:N} \\
&= \frac{1}{N^{T-1}}\sum_{b_{1:T-1}^1}\int\frac{\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}^1})}{Z_T}\,\mathrm{d}x_{1:T}^1 = \int\frac{\widetilde{\gamma}_T(x_{1:T})}{Z_T}\,\mathrm{d}x_{1:T} = 1,
\end{aligned}
$$

where in the first equality we note that the sum over $i$ generates $N$ equal values, and thus we can arbitrarily choose one of these (in this case $i = 1$) to evaluate and multiply the result by $N$. In the second equality we marginalize the variables not involved in the particle $x_{1:T}^{b_{1:T}^1}$. The two final equalities follow

because we are averaging $N^{T-1}$ values that are all equal to 1. This concludes the proof.

# 5 Conditional SMC: Algorithms and Applications

Conditional sequential Monte Carlo (CSMC) is a recent algorithm combining SMC and MCMC, originally developed as a means to approximate and mimic idealized Gibbs samplers for local latent variable models. The intractability of the exact Gibbs sampler stems from the challenge in simulating from the exact conditional of latent variables given the data and the parameters. The key idea in CSMC is to simulate from this conditional distribution approximately using a slightly modified SMC approximation, retaining a single particle at each iteration. By iterating this procedure we obtain a MCMC method that has the desired target distribution as its stationary distribution. We will see how this can be used not only for simulating from a target distribution, but also how we can use it to generate unbiased inverse normalization constant estimates, speed up inference for e.g. probabilistic programming, and evaluating other approximate inference methods.

In the first section below we introduce the CSMC algorithm and illustrate it with an example. Then, we discuss the unbiasedness property of the CSMC inverse normalization constant estimate, a key property underpinning the rest of the section. Using CSMC as a tool we move on by deriving the distribution of the expected SMC empirical distribution approximation. Next, we introduce the IPMCMC algorithm, leveraging CSMC and SMC to derive a method that can take advantage of distributed and multi-core computational architectures. Finally, we discuss ways in which SMC and CSMC can be used for evaluating approximate inference.

## 5.1 Conditional Sequential Monte Carlo

One way to look at CSMC is to view it as Markov kernel with the target distribution $\gamma_T$ as its stationary distribution. Each iteration takes the previous sample as an input, a reference trajectory, and outputs an updated sample with no need for an accept–reject step. If we keep iterating, applying CSMC over and over, we obtain the *iterated* CSMC algorithm. The iterated CSMC sampler is a type of Gibbs sampler on an extended space of all random variables generated at each step [Andrieu et al., 2010]. We will sometimes refer to the previous sample $x'_{1:T}$, i.e. the reference trajectory, as the *retained particle*.

The CSMC algorithm differs from standard SMC described in Section 2.2 only in that a single particle is set deterministically to the retained particle from the previous iteration. This means that we condition on the event that a

specific particle survives and is equal to the retained particle $x'_{1:T}$, hence the name *conditional* SMC. We obtain the CSMC algorithm by a simple modification of the SMC proposal and weight update equations. At each iteration, for the first $N-1$ particles, $i = 1, \ldots, N-1$, we follow the standard updates according to Eqs. (29) and (30). The proposal is

$$x^i_{1:t} \sim \widehat{\gamma}_{t-1}(x_{1:t}) q_t(x_t \mid x_{1:t-1}), \tag{107}$$

and the weights

$$\widetilde{w}^i_t = \frac{\widetilde{\gamma}_t\left(x^i_{1:t}\right)}{\widetilde{\gamma}_{t-1}\left(x^i_{1:t-1}\right) q_t(x^i_t \mid x^i_{1:t-1})}. \tag{108}$$

The last particle $i = N$ we set deterministically equal to the retained particle $x^N_{1:t} = x'_{1:t}$. Its weight is computed according to Eq. (108), that is we use the same weight expression for all $N$ particles. The empirical distribution approximation of $\gamma_T$ is, analogously to SMC, equal to

$$\widehat{\gamma}_T(x_{1:T}) = \sum_{i=1}^{N} \frac{\widetilde{w}^i_T}{\sum_j \widetilde{w}^j_T} \delta_{x^i_{1:T}}. \tag{109}$$

The updated retained particle is obtained by drawing a single sample from the empirical distribution approximation in Eq. (109), i.e. $x'_{1:T} \sim \widehat{\gamma}_T(x_{1:T})$.

We summarize the CSMC algorithm in Algorithm 15. Iterated CSMC is obtained by letting the retained particle at the next iteration be the output of the previous step. This algorithm constructs a Markov chain, consisting of the retained particles at each iteration, with the target distribution $\gamma_T$ as its stationary distribution. For a thorough treatment of the theory of iterated CSMC and particle Gibbs (PG), i.e. the extension to sampling also model parameters, see Andrieu et al. [2010].

The path degeneracy issue discussed in Section 2.2 is particularly prominent in iterated CSMC. Because we enforce that the retained particle survives to the very end, we simultaneously force that all the particles collapse for early steps $t$ to $x'_{1:t}$. This means that the output from Algorithm 15, with high probability, is equal to the input for early values of $x'_{1:T}$. The samples we get are highly correlated and any estimator will require a significant number of samples to reasonably approximate expectation with respect to the target distribution $\gamma_T$. By modifying the way we update particle $N$, the retained particle,

**Algorithm 15:** Conditional sequential Monte Carlo (CSMC)

**input :** Unnormalized target distributions $\widetilde{\gamma}_t$, proposals $q_t$, retained particle $x'_{1:T}$, number of samples $N$.
**output:** Updated $x'_{1:T}$ approximately distributed as $\gamma_T$.

**for** $t = 1$ **to** $T$ **do**
  **for** $i = 1$ **to** $N-1$ **do**
    Sample $x^i_{1:t} \sim \widehat{\gamma}_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})$ (see Eq. (107))
    Set $\widetilde{w}^i_t = \frac{\widetilde{\gamma}_t(x^i_{1:t})}{\widetilde{\gamma}_{t-1}(x^i_{1:t-1}) q_t(x^i_t \mid x^i_{1:t-1})}$ (see Eq. (108))
  **end**
  Set $x^N_{1:t} = x'_{1:t}$
  Set $\widetilde{w}^N_t = \frac{\widetilde{\gamma}_t(x^N_{1:t})}{\widetilde{\gamma}_{t-1}(x^N_{1:t-1}) q_t(x^N_t \mid x^N_{1:t-1})}$
  Set $w^i_t = \frac{\widetilde{w}^i_t}{\sum_j \widetilde{w}^j_t}$, for $i = 1, \ldots, N$
**end**
Sample $x'_{1:T} \sim \widehat{\gamma}_T(x_{1:T})$
**return** $x'_{1:T}$

we can alleviate this problem. One way to do this is using *ancestor sampling* [Lindsten et al., 2014]. With ancestor sampling, instead of setting $x^N_{1:t} = x'_{1:t}$ deterministically as in Algorithm 15, we sample at each time step

$$x'_{1:t} \sim \sum_{i=1}^{N} \frac{\widetilde{\nu}^i_t}{\sum_j \widetilde{\nu}^j_t} \delta_{x^i_{1:t-1}}(x_{1:t-1}) \cdot \delta_{x'_t}(x_t), \tag{110}$$

where the auxiliary weights $\widetilde{\nu}^i_t$ are given by

$$\widetilde{\nu}^i_t = w^i_{t-1} \frac{\widetilde{\gamma}_T\left((x^i_{1:t-1}, x'_{t:T})\right)}{\widetilde{\gamma}_{t-1}\left(x^i_{1:t-1}\right)}. \tag{111}$$

For an in-depth treatment of the path degeneracy issue and its other potential solutions we refer to Lindsten and Schön [2013].

**Example 16** (Conditional sequential Monte Carlo for Example 1)**.** *We revisit our running non-Markovian Gaussian example. We let* $T = 20$ *and study the performance of iterated* CSMC *without (blue) and with ancestor sampling (red) in*

*Fig. 10. We set $N = 5$, 10 and study the kernel density estimates based on* $10,000$ MCMC *samples of the marginal distributions $\gamma_T(x_1)$ and $\gamma_T(x_T)$, respectively.*

*We can see the striking effect that path degeneracy has the iterated* CSMC *approximation in Fig. 10a; effectively we approximate $\gamma_T(x_1)$ using a single sample when $N = 5$, whereas the approximation for $\gamma_T(x_T)$ is more diverse. When using ancestor sampling on the other hand, the approximation is already accurate for $N = 5$. This accuracy comes at a cost however, the ancestor sampling step requires that we evaluate Eq.* (111) *at each iteration. This extra step has a computational cost that in general may be proportional to $T^2$, whereas the cost of standard* CSMC *is often only on the order of $T$. For some target distributions, such as the* SSM *or our running example, there is only a small constant computational overhead associated with evaluating Eq.* (111)*. However, whether the benefit of ancestor sampling will outweigh its computational cost will have to be evaluated on a case by case basis.*

## 5.2 The Unbiasedness of the Inverse Normalization Constant Estimate

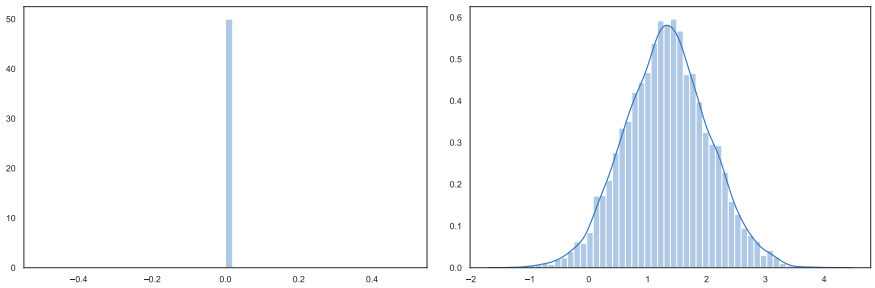
Unlike in the standard SMC algorithm, the normalization constant estimate for CSMC based on the product of the average weights is biased. We restate the normalization constant estimate below:

$$\widehat{Z}_T = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} \widetilde{w}_t^i. \tag{112}$$

Because we condition on a reference trajectory, this normalization constant estimate has different properties than does the SMC estimate. However, it is still consistent, converging almost surely to $Z_T$ as the number of particles tend to infinity.

While the above estimator is biased, interestingly the CSMC inverse normalization constant estimate $1/\widehat{Z}_T$ *is unbiased*. We will see that this is particularly relevant for the inference evaluation methods that we will study in Section 5.5. The unbiasedness holds under the assumption that the reference trajectory is distributed according to the target distribution $x'_{1:T} \sim \gamma_T(x_{1:T})$. This assumption relies on the Markov chain having already converged to its stationary distribution. However, in some applications it is possible to make use of CSMC with the problem set up in such a way that this is always true.

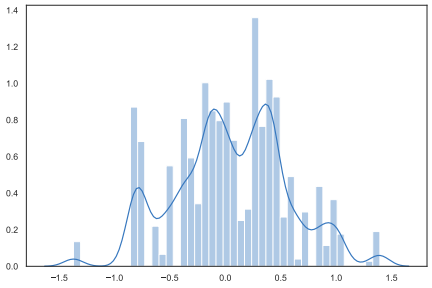
We formalize and prove the unbiasedness result in Proposition 3.



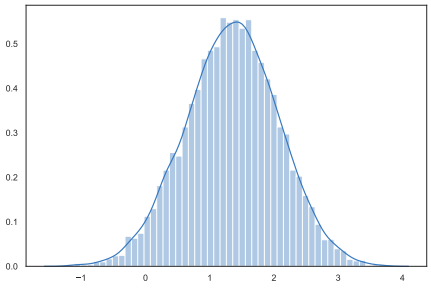
**(a)** $t = 1, N = 5$



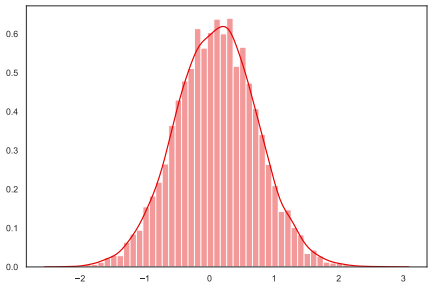
**(b)** $t = 20, N = 5$



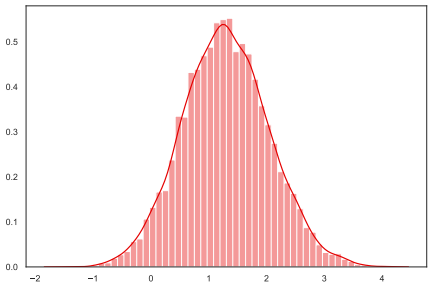
**(c)** $t = 1, N = 10$



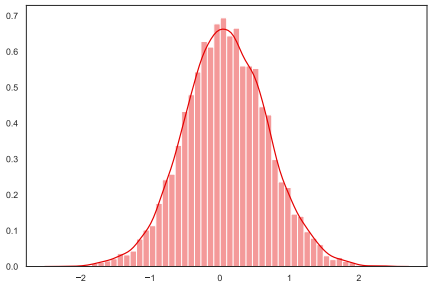
**(d)** $t = 20, N = 10$



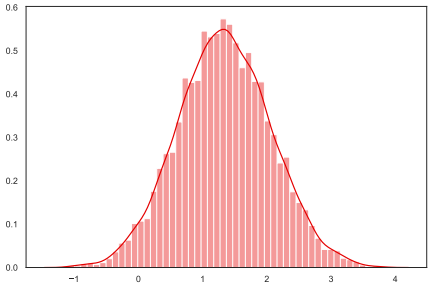
**(e)** $t = 1, N = 5$



**(f)** $t = 20, N = 5$



**(g)** $t = 1, N = 10$



**(h)** $t = 20, N = 10$

**Figure 10:** Illustration of the kernel density estimates based on $10,000$ MCMC samples of the marginal distributions $\gamma_T(x_1)$ and $\gamma_T(x_T)$ from iterated CSMC (top two rows, blue) and CSMC with ancestor sampling (bottow two rows, red).

**Proposition 3.** *Assuming that the retained particle $x'_{1:T} \sim \gamma_T(x_{1:T})$, then the* CSMC *inverse normalization constant estimate $\frac{1}{\widehat{Z}_T}$ is non-negative and unbiased*

$$\mathbb{E}\left[\frac{1}{\widehat{Z}_T}\right] = \frac{1}{Z_T}, \quad \frac{1}{\widehat{Z}_t} \geq 0. \tag{113}$$

*Proof.* See Appendix 5.A. □

## 5.3 The Expected SMC Empirical Distribution

Another method for representing the expected sequential Monte Carlo empirical distribution $\mathbb{E}[\widehat{\gamma}_T(x_{1:T})]$ is using CSMC. This can be useful for some applications of SMC, such as in variational SMC or in inference evaluation which we will see later in this chapter. The marginal distribution of a single sample from the SMC empirical distribution $\widehat{\gamma}_T$ is

$$\mathbb{E}[\widehat{\gamma}_T(x_{1:T})] = \mathbb{E}_{\mathrm{Q_{SMC}}}\left[\sum_{i=1}^{N} w_T^i \delta_{x_{1:T}^i}(x_{1:T})\right], \tag{114}$$

where $\mathrm{Q_{SMC}}$ (see Eq. (105) for its definition) is the distribution of all particles and ancestor variables generated by a single run of the SMC algorithm.

We formalize the alternate representation of the expected SMC empirical distribution in Proposition 4.

**Proposition 4.** *The expected* SMC *empirical distribution satisfies*

$$\mathbb{E}_{Q_{SMC}}[\widehat{\gamma}_T(x_{1:T})] = \widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{Q_{CSMC}}\left[\widehat{Z}_T^{-1} \,|\, x_{1:T}\right], \tag{115}$$

*where $Q_{CSMC}$ is the distribution of all random variables generated by a single run of the* CSMC *algorithm with $x_{1:T}$ as its retained particle.*

*Proof.* See Appendix 5.B. □

## 5.4 Parallelization

In Section 4.4 we briefly discussed a few approaches for leveraging multi-core and distributed computational architectures. Here we will see how we can combine SMC and CSMC to derive IPMCMC [Rainforth et al., 2016], an algorithm

that takes advantage of parallelization to speed up inference for amongst other things probabilistic programming.

The IPMCMC method, like the iterated CSMC, can be thought of as a type of Gibbs sampler on an extended space that draws approximate samples from the target distribution $\gamma_T$. However unlike iterated CSMC, the IPMCMC algorithm generates not only one sample at each iteration but several. The method is based on a pool of several interacting standard and conditional sequential Monte Carlo samplers. These pool of samplers are run as parallel processes and will be referred to as nodes below. Each node either runs an iteration of CSMC like in Algorithm 15, or an iteration of SMC like in Algorithm 3. After each run of this pool, we update the node indices running CSMC by applying successive Gibbs updates. This means that the nodes running CSMC can change from iteration to iteration. This approach can let the user trade off exploration (SMC) and exploitation (CSMC) to achieve a faster and more accurate approximate inference method.

To simplify notation we will let $\mathbf{x}$ denote the full trajectory $x_{1:T}$. We assume that we have a total of $M$ nodes, indexed by $j = 1, \ldots, M$, of which $P$ run CSMC and the remaining run standard SMC. The conditional nodes, i.e. the nodes running CSMC at any given time, are denoted by a set of indices $c_{1:P} \in \{1, \ldots, M\}^P$ where each $c_j$ is distinct. The conditional node $c_j$'s retained particle is referred to by $\mathbf{x}'_j$. After each run all nodes produce a normalization constant estimate $\widehat{Z}_T^j$, i.e. the product of the average unnormalized weights in that node. IPMCMC proceeds by successively updating the conditional node indices and retained particles by Gibbs steps. For $j = 1, \ldots, P$ we sample

$$\mathbb{P}(c_j = i) = \frac{\widehat{Z}_T^i \cdot \mathbb{1}\left(i \notin c_{\neg j}\right)}{\sum_{k=1}^{P} \widehat{Z}_T^k \cdot \mathbb{1}\left(k \notin c_{\neg j}\right)}, \tag{116}$$

where $\mathbb{1}(\cdot)$ is the identity function evaluating to one if its argument is true and zero otherwise. Furthermore, we have used the node index shorthand $c_{\neg j} := \{c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_P\}$. After selecting new nodes to become conditional nodes through Eq. (116) we select their new corresponding retained particle by simulating

$$\mathbf{x}'_j \sim \widehat{\gamma}_T^{c_j}(\mathbf{x}), \quad j = 1, \ldots, P, \tag{117}$$

where $\widehat{\gamma}_T^{c_j}$ is the standard target distribution approximation from node $c_j$.

We summarize one iteration of IPMCMC in Algorithm 16. Iterating the update steps, letting the retained particles at the previous step be the input for

**Algorithm 16:** Interacting particle Markov chain Monte Carlo

**input :** Number of nodes $M$, conditional nodes $P$, retained particles $\mathbf{x}'_{1:P}$, conditional node indices $c_{1:P}$.

**output:** Updated $\mathbf{x}'_{1:P}$, each approximately distributed as $\gamma_T$, and updated node indices $c_{1:P}$.

Nodes $1 : M \backslash c_{1:P}$ run Algorithm 3 (SMC)
Nodes $c_{1:P}$ run Algorithm 15 (CSMC)
**for** $j = 1$ **to** $P$ **do**
  Select a new conditional node by sampling $c_j$ (see Eq. (116))
  Set a new retained particle $\mathbf{x}'_j$ by simulating from $\widehat{\gamma}_T^{c_j}$ (see Eq. (117))
**end**
**return** $\mathbf{x}'_{1:P}$ *and* $c_{1:P}$

the subsequent step, generates a Markov chain on an extended space where each retained particle is approximately distributed according to the target distribution.

IPMCMC is suited to distributed and multi-core architectures. The main computational cost is running the SMC and CSMC iterations, which can be done in parallel by the nodes. At each iteration, only a single normalization constant estimate needs be communicated between nodes and the calculation of the updates of $c_{1:P}$ is negligible. Retained particles do not need to be communicated and may be stored locally in each node until needed. Furthermore, IPMCMC may be amenable to asynchronous adaptation under certain assumptions [Rainforth et al., 2016]. The algorithm is also suited for use in probabilistic programming where we are often only able to simulate from the prior and evaluate the likelihood. Interacting particle Markov chain Monte Carlo can allow for tackling more challenging inference problems, i.e. more complex probabilistic programs, compared to standard MCMC methods.

## 5.5 Evaluating Inference

Approximate Bayesian inference is central to a large part of probabilistic machine learning and its application. It is a coherent approach to treat uncertainty in both the data and the model with applications to everything from robotics to health care. A notoriously difficult problem is to measure the accuracy of our posterior approximation compared to the true posterior distribution. Infer-

ence checking, or inference evaluation, refers to the process of comparing our approximate inference result to what we would have obtained using the true posterior. In practice whether an approximation is accurate or not may differ between different applications, we might be interested in different aspects of the posterior. Some examples are tail probabilities in rare event estimation or posterior mean for minimum mean square error estimation. In this section we will focus on comparing directly the posterior approximation to the exact posterior distribution in terms of symmetrized KL divergence, also known as the Jeffreys divergence, following Grosse et al. [2016], Cusumano-Towner and Mansinghka [2017].

The posterior distribution that we want to approximate is denoted by $p(\mathbf{x}\,|\,\mathbf{y})$, and the approximate posterior will be denoted by $q(\mathbf{x})$. The approximation $q$ could for example be the expected SMC empirical distribution approximation, similar to VSMC, or it could be an optimized variational approximation, or any of the other examples detailed in Grosse et al. [2016], Cusumano-Towner and Mansinghka [2017]. In either case, the symmetrized KL divergence from $q$ to $p$ is given by

$$\mathrm{D_J}\left(q(\mathbf{x})\,\|\,p(\mathbf{x}\,|\,\mathbf{y})\right) = \mathrm{KL}\left(q(\mathbf{x})\,\|\,p(\mathbf{x}\,|\,\mathbf{y})\right) + \mathrm{KL}\left(p(\mathbf{x}\,|\,\mathbf{y})\,\|\,q(\mathbf{x})\right), \tag{118}$$

where the standard Kullback-Leibler divergence from $q(\mathbf{x})$ to $p(\mathbf{x})$ is defined by $\mathrm{KL}\left(q(\mathbf{x})\,\|\,p(\mathbf{x})\right) := \mathbb{E}_{q(\mathbf{x})}\left[\log q(\mathbf{x}) - \log p(\mathbf{x})\right]$. We can rewrite Eq. (118) to eliminate the need for computing the marginal likelihood $p(\mathbf{y})$ as follows

$$\begin{aligned}\mathrm{D_J}\left(q(\mathbf{x})\,\|\,p(\mathbf{x}\,|\,\mathbf{y})\right) &= \mathbb{E}_{q(\mathbf{x})}\left[\log \frac{q(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right] + \mathbb{E}_{p(\mathbf{x}\,|\,\mathbf{y})}\left[\log \frac{p(\mathbf{x}\,|\,\mathbf{y})}{q(\mathbf{x})}\right] \\ &= \mathbb{E}_{q(\mathbf{x})}\left[\log q(\mathbf{x}) - \log p(\mathbf{x},\mathbf{y})\right] + \mathbb{E}_{p(\mathbf{x}\,|\,\mathbf{y})}\left[\log p(\mathbf{x},\mathbf{y}) - \log q(\mathbf{x})\right]. \end{aligned} \tag{119}$$

To be able to estimate Eq. (119) using a standard MC estimator we require that we can sample $q(\mathbf{x})$ and $p(\mathbf{x}\,|\,\mathbf{y})$, respectively. Furthermore, we require that we are able to evaluate the PDFs $q(\mathbf{x})$ and $p(\mathbf{x},\mathbf{y})$ pointwise. These requirements can be restrictive for the types of problems we encounter in practice, and Grosse et al. [2016], Cusumano-Towner and Mansinghka [2017] make various relaxations and assumptions to make the problem tractable. Below we will detail the two approaches to inference checking known as bounding divergences with reverse annealing (BREAD) [Grosse et al., 2016] and auxiliary inference divergence estimator (AIDE) [Cusumano-Towner and Mansinghka, 2017].

**Bounding Divergences with Reverse Annealing** The BREAD approach focuses on bounding the symmetrized KL for a posterior approximation based on annealed importance sampling (AIS) [Neal, 2001] for *simulated data*. By using simulated data we ensure that we can generate an exact sample from the posterior distribution $\mathbf{x} \sim p(\mathbf{x}|\mathbf{y})$. We can think of AIS as a single sample, $N = 1$, SMC method that uses as its proposal a Markov kernel with stationary distribution $\gamma_t(x_t)$. The target distribution is typically chosen according to a data or likelihood tempering scheme as discussed in Section 1.2. Per definition the posterior distribution is obtained as the marginal at the final iteration, i.e. $p(\mathbf{x}|\mathbf{y}) = \gamma_T(x_T)$ where $\mathbf{x} \equiv x_T$. The posterior distribution approximation is then defined by $q(\mathbf{x}) := \mathbb{E}[\widehat{\gamma}_T(x_T)]$ and the expectation is with respect to all the proposal distributions.

Because of this set up the intractability in estimating, or bounding, the Jeffreys divergence in Eq. (119) comes from having to evaluate the PDF $q(\mathbf{x})$ pointwise. We can bound Eq. (119) from above by noting that marginalizing out variables cannot increase the KL divergence

$$\begin{aligned}\mathrm{D_J}(q(\mathbf{x}) \,\|\, p(\mathbf{x}|\mathbf{y})) &\leq \mathrm{D_J}(\mathbb{E}[\widehat{\gamma}_T(x_{1:T})] \,\|\, \gamma_T(x_{1:T})) \\ &= \mathrm{D_J}\left(\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \,|\, x_{1:T}\right] \,\|\, \gamma_T(x_{1:T})\right), \end{aligned} \tag{120}$$

where the equality follows from Proposition 4. We can further bound Eq. (120) using Jensen's inequality and simplifying, leading to

$$\begin{aligned}&\mathrm{D_J}\left(\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \,|\, x_{1:T}\right] \,\|\, \gamma_T(x_{1:T})\right) \\ &\leq \mathbb{E}_{\mathrm{Q_{SMC}}}\left[-\log \widehat{Z}_T\right] + \mathbb{E}_{\gamma_T}\left[\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\log \widehat{Z}_T \,|\, x_{1:T}\right]\right]. \end{aligned} \tag{121}$$

The inequality in Eq. (121) holds generally even if we would use a standard SMC algorithm rather than AIS as our posterior approximation. The details are provided in Section 5.C. However, when AIS is used we can further simplify to obtain the same bound as Grosse et al. [2016]

$$\begin{aligned}&\mathbb{E}_{\mathrm{Q_{SMC}}}\left[-\log \widehat{Z}_T\right] + \mathbb{E}_{\gamma_T}\left[\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\log \widehat{Z}_T \,|\, x_{1:T}\right]\right] \\ &= \mathbb{E}_{q(x_{1:T})}\left[-\sum_{t=1}^{T} \log \widetilde{w}_t\right] + \mathbb{E}_{p(x_T|\mathbf{y})s(x_{1:T-1}|x_T)}\left[\sum_{t=1}^{T} \log \widetilde{w}_t\right], \end{aligned} \tag{122}$$

where $q(x_{1:T}) = q_1(x_1)\prod_{t=1}^{T} q_t(x_t \,|\, x_{t-1})$ are the proposals (forward kernels), $s(x_{1:T-1} \,|\, x_T) = \prod_{t=1}^{T-1} s_t(x_t \,|\, x_{t+1})$ are the corresponding backward kernels, and

the weights are in the likelihood tempering variant given by

$$\widetilde{w}_t = p(\mathbf{y} \mid x_{t-1})^{\tau_t - \tau_{t-1}},$$

with $\widetilde{w}_1 = 1$. This means it is possible to obtain a stochastic upper bound, i.e. a MC estimate of the upper bound in Eq. (122), by running a standard (forward) AIS evaluating the negative log-weights, and running a *reverse* AIS evaluating the log-weights; the left and right expressions in Eq. (122), respectively.

**Auxiliary Inference Divergence Estimator** The AIDE approach, on the other hand, treats a more general class of posterior approximations such as variational inference, SMC, or MCMC. However, rather than assuming simulated data, Cusumano-Towner and Mansinghka [2017] propose to study the Jeffreys divergence from the approximation to that of a gold standard inference approximation. Typically this gold standard approximation would consist of a long run of an MCMC method or SMC with a large number of particles that we know approximates the posterior well.

In this setting not only is the PDF for the approximation $q(\mathbf{x})$ difficult to evaluate, so is the PDF for our gold standard algorithm. Rather than introducing new notation for the gold standard algorithm we will keep the posterior distribution notation $p(\mathbf{x} \mid \mathbf{y})$. The approach that the AIDE takes is to use *meta-inference* methods, based on SMC or CSMC, to obtain approximations of the PDFs and their reciprocals. A meta-inference algorithm takes as input a sample from the distribution of interest and outputs an unbiased estimate of either its PDF or the reciprocal. In fact it suffices that the meta-inference algorithms provide estimates that are unbiased up to a constant of proportionality because any constant is canceled by the symmetry in the Jeffrey's divergence.

We note that the approach suggested by Cusumano-Towner and Mansinghka [2017] is essentially equivalent to the bound derived in Eqs. (120) to (122). However, instead of the exact target distribution $\gamma_T(x_{1:T})$ we have another SMC approximation to represent the gold standard algorithm. To differentiate between the two SMC approximations we will use superscript $q$ for the approximation and $p$ for the gold standard. With this the corresponding Jeffrey's divergence is

$$\begin{aligned}
&\mathrm{D_J}(q(\mathbf{x}) \,\|\, p(\mathbf{x} \mid \mathbf{y})) = \mathrm{D_J}\left(\mathbb{E}[\widehat{\gamma}_T^q(x_{1:T})] \,\|\, \mathbb{E}[\widehat{\gamma}_T^p(x_{1:T})]\right) \\
&= \mathrm{D_J}\left(\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q}^q_{\mathrm{CSMC}}}[1/\widehat{Z}_T^q \mid x_{1:T}] \,\|\, \widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q}^p_{\mathrm{CSMC}}}[1/\widehat{Z}_T^p \mid x_{1:T}]\right), \qquad (123)
\end{aligned}$$

where $\mathrm{Q}^q_{\mathrm{CSMC}}$, $\widehat{Z}^q_T$ and $\mathrm{Q}^p_{\mathrm{CSMC}}$, $\widehat{Z}^p_T$ are the CSMC distributions and normalization constant estimates for the approximation and gold standard algorithm, respectively. The first equality follows from the definition of $q(\mathbf{x})$ and $p(\mathbf{x}|\mathbf{y})$. The second equality follows from Proposition 4 for the two distributions, respectively. The divergence in Eq. (123) can be bounded, analogously to the BREAD case, as follows

$$\begin{aligned}
&\mathrm{D}_{\mathrm{J}}\left(\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q}^q_{\mathrm{CSMC}}}\left[1/\widehat{Z}^q_T \,|\, x_{1:T}\right] \| \widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q}^p_{\mathrm{CSMC}}}\left[1/\widehat{Z}^p_T \,|\, x_{1:T}\right]\right) \\
&\leq \mathbb{E}_{q(\mathbf{x})}\left[\mathbb{E}_{\mathrm{Q}^p_{\mathrm{CSMC}}}\left[\log \widehat{Z}^p_T \,|\, x_{1:T}\right]\right] - \mathbb{E}_{\mathrm{Q}^q_{\mathrm{SMC}}}\left[\log \widehat{Z}^q_T\right] + \\
&+ \mathbb{E}_{p(\mathbf{x}|\mathbf{y})}\left[\mathbb{E}_{\mathrm{Q}^q_{\mathrm{CSMC}}}\left[\log \widehat{Z}^q_T \,|\, x_{1:T}\right]\right] - \mathbb{E}_{\mathrm{Q}^p_{\mathrm{SMC}}}\left[\log \widehat{Z}^p_T\right],
\end{aligned} \tag{124}$$

where $q(\mathbf{x})$ and $p(\mathbf{x}|\mathbf{y})$ are the two corresponding expected SMC empirical distributions.

The bound above differs slightly from that derived by Cusumano-Towner and Mansinghka [2017]. The AIDE procedure uses a form of importance weighted version of Eq. (124), in their notation it is equivalent to the bound based on a single run of the meta-inference for the approximation ($M_t = 1$) and gold standard ($M_g = 1$). For more details we refer the interested reader to that paper.

## 5.A Proof of Unbiasedness

When introducing the CSMC method we consistently set the $N$:th particle to the reference trajectory. In practice it does not matter which particle is deterministically set to the reference trajectory. It will prove simplifying to let the particle that is conditioned on, i.e. the reference trajectory, have its own index $b_{1:T} = (b_1, \ldots, b_T) \in \{1, \ldots, N\}^T$. For clarity we focus on using multinomial resampling at each iteration of the algorithm. The distribution of all random variables generated by the CSMC method in Algorithm 15 is then given by

$$\begin{aligned} &\mathrm{Q}_{\mathrm{CSMC}}\left(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}\right) \\ &= \prod_{i \neq b_1} q_1(x_1^i) \cdot \prod_{t=2}^{T} \prod_{i \neq b_t} \left[ w_{t-1}^{a_{t-1}^i} q_t\left(x_t^i \mid x_{1:t-1}^{a_{t-1}^i}\right)\right], \end{aligned} \tag{125}$$

where the particles are $x_{1:T}^{\neg b_{1:T}} = \bigcup_{t=1}^{T} \left\{x_t^i\right\}_{i=1}^{N} \backslash \{x_t^{b_t}\}_{t=1}^{T}$, and the ancestor variables from the resampling step are $a_{1:T-1}^{\neg b_{1:T-1}} = \bigcup_{t=1}^{T-1} \left\{a_t^i\right\}_{i=1}^{N} \backslash \{b_t\}_{t=1}^{T-1}$. Assuming that $\widetilde{w}_t^i \geq 0$ for all $i$ and $t$, we have that $\widehat{Z}_T \geq 0$ and thus ${}^1/_{\widehat{Z}_T} \geq 0$. We are now going to prove that when $\left(x_{1:T}^{b_{1:T}}, b_{1:T}\right) \sim \gamma_T(x_{1:T}) \cdot N^{-T}$

$$\mathbb{E}_{b_{1:T}}\left[\mathbb{E}_{\gamma_T \cdot \mathrm{Q}_{\mathrm{CSMC}}}\left[\frac{Z_T}{\widehat{Z}_T}\right]\right] = 1, \tag{126}$$

from which the result follows. We rewrite the integrand in Eq. (126),

$$\begin{aligned} &\frac{Z_T}{\widehat{Z}_T} \frac{\gamma_T(x_{1:T}^{b_{1:T}})}{N^T} \mathrm{Q}_{\mathrm{CSMC}}\left(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}\right) \\ &= \frac{1}{\widehat{Z}_T} \frac{\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}})}{N^T} \frac{\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right)}{q_1(x_1^{b_1}) \prod_{t=2}^{T} w_{t-1}^{b_{t-1}} q_t\left(x_t^{b_t} \mid x_{1:t-1}^{b_{t-1}}\right)} \\ &= \frac{\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}})}{\widehat{Z}_T} \cdot \frac{\prod_{t=1}^{T} \frac{1}{N} \sum_i \widetilde{w}_t^i}{q_1(x_1^{b_1}) \prod_{t=2}^{T} \widetilde{w}_{t-1}^{b_{t-1}} q_t\left(x_t^{b_t} \mid x_{1:t-1}^{b_{t-1}}\right)} \cdot \frac{\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right)}{\sum_i \widetilde{w}_T^i} \\ &= \frac{\widetilde{w}_T^{b_T}}{\sum_i \widetilde{w}_T^i} \mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right), \end{aligned}$$

where the first line is the integrand. In the first equality we have divided and multiplied by $q_1 \prod_{t=2}^{T} w_{t-1}^{b_{t-1}} q_t$ to be able to identify the difference between

$\mathrm{Q}_{\mathrm{SMC}}$ and $\mathrm{Q}_{\mathrm{CSMC}}$. Note that we have replaced $b_t = a_t^{b_{t+1}}$ in the notation for $\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right)$ to identify the SMC distribution. For the second equality we have rewritten the normalized weights in the denominator, together with $N^{-T}$ and dividing and multiplying by $\sum_i \widetilde{w}_T^i$, to identify the normalization constant estimator. For the final equality we see that the normalization constant estimators cancel, and the product in the denominator simplifies together with $\widetilde{\gamma}_T$ to become $\widetilde{w}_T^{b_T}$.

Using this simplified expression for the integrand in Eq. (126) we obtain

$$\begin{aligned}
&\sum_{b_T} \sum_{a_{1:T-1}^{1:N}} \int \frac{\widetilde{w}_T^{b_T}}{\sum_i \widetilde{w}_T^i} \mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right) \mathrm{d}x_{1:T}^{1:N} \\
&= \sum_{a_{1:T-1}^{1:N}} \int \mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right) \mathrm{d}x_{1:T}^{1:N} = 1,
\end{aligned}$$

where again we first replace the original sum over $b_{1:T-1}$ with the ancestor variable $a$ notation. The first equality marginalizes $b_T$ and the final equality follows because $\mathrm{Q}_{\mathrm{SMC}}$ is a distribution on $x_{1:T}^{1:N}, a_{1:T-1}^{1:N}$. This concludes the proof.

## 5.B Proof of SMC Distribution Representation

We will prove the following alternate representation of the expected SMC empirical distribution

$$\mathbb{E}_{\mathrm{Q}_{\mathrm{SMC}}}\left[\widehat{\gamma}_T(x_{1:T})\right] = \widetilde{\gamma}_T(x_{1:T}) \mathbb{E}_{\mathrm{Q}_{\mathrm{CSMC}}}\left[\widehat{Z}_T^{-1} \,|\, x_{1:T}\right], \tag{127}$$

were we for clarity again focus on using multinomial resampling at each iteration of the algorithm. Note that the left hand side is a probability measure and the right hand side a probability distribution function. To be precise the equality means that the corresponding probability measure of the right hand side (usually adding either the Lebesgue our counting measure) is equal almost everywhere to the left hand side.

To prove the result we need to show that the expectation of any bounded and continuous function $h(x_{1:T})$ coincides for the left and right hand sides of Eq. (127). We have that the expectation of $h(x_{1:T})$ with respect to the right

hand side of Eq. (127) is

$$
\begin{aligned}
&\int h(x_{1:T})\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q}_{\mathrm{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right]\mathrm{d}x_{1:T} \\
&= \sum_{b_{1:T}} \int \frac{1}{N^T} h(x_{1:T}^{b_{1:T}})\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}})\mathbb{E}_{\mathrm{Q}_{\mathrm{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}^{b_{1:T}}\right]\mathrm{d}x_{1:T}^{b_{1:T}} \\
&= \sum_{b_{1:T}} \sum_{a_{1:T-1}^{\neg b_{1:T-1}}} \int \left[ \frac{h(x_{1:T}^{b_{1:T}})\widetilde{\gamma}_T(x_{1:T}^{b_{1:T}})}{N^T \prod_{t=1}^T \frac{1}{N}\sum_{i=1}^N \widetilde{w}_t^i} \cdot \right. \\
&\qquad \left. \cdot \mathrm{Q}_{\mathrm{CSMC}}\left(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}\right) \right] \mathrm{d}x_{1:T}^{b_{1:T}}\, \mathrm{d}x_{1:T}^{\neg b_{1:T}} \\
&= \sum_{b_T} \sum_{a_{1:T-1}^{1:N}} \int h(x_{1:T}^{b_T}) \frac{\widetilde{w}_T^{b_T}}{\sum_{i=1}^N \widetilde{w}_T^i} \mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right)\mathrm{d}x_{1:T}^{1:N} \\
&= \int h(x_{1:T}) \cdot \\
&\qquad \cdot \left[ \int \sum_{a_{1:T-1}^{1:N}} \sum_{b_T} \frac{\widetilde{w}_T^{b_T}}{\sum_{i=1}^N \widetilde{w}_T^i} \delta_{x_{1:T}^{b_T}}(x_{1:T}) \mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right)\mathrm{d}x_{1:T}^{1:N} \right] \mathrm{d}x_{1:T} \\
&= \int h(x_{1:T})\mathbb{E}_{\mathrm{Q}_{\mathrm{SMC}}}\left[\widehat{\gamma}_T(x_{1:T})\right]\mathrm{d}x_{1:T},
\end{aligned}
$$

where the first equality is the definition. In the second equality we have expanded the terms and can identify the expression derived in Section 5.A above. In the third equality we have replaced it with the corresponding $\mathrm{Q}_{\mathrm{SMC}}$ distribution, again using $b_t = a_t^{b_{t+1}}$ in the notation for $\mathrm{Q}_{\mathrm{SMC}}$. In the penultimate equality we re-arrange a few terms and replace the sum over $b_T$ with an integration with respect to a point mass distribution. In the final equality we identify the expectation with respect to $\mathbb{E}_{\mathrm{Q}_{\mathrm{SMC}}}[\widehat{\gamma}_T(x_{1:T})]$. This concludes the proof.

## 5.C Jeffreys Divergence Bound

We detail the derivation of the bound in Eq. (121) below under the assumption of multinomial resampling at each iteration of the algorithm,

$$
\begin{aligned}
&\mathrm{D_J}\left(\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right] \| \gamma_T(x_{1:T})\right) \\
&= \int \widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right] \log \frac{\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right]}{\gamma_T(x_{1:T})} \mathrm{d}x_{1:T} \\
&\quad + \int \gamma_T(x_{1:T}) \log \frac{\gamma_T(x_{1:T})}{\widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right]} \mathrm{d}x_{1:T} \\
&= \int \widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right] \log \mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right] \mathrm{d}x_{1:T} \\
&\quad - \int \gamma_T(x_{1:T}) \log \mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \mid x_{1:T}\right] \mathrm{d}x_{1:T} \\
&\leq \int \widetilde{\gamma}_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\widehat{Z}_T^{-1} \log \widehat{Z}_T^{-1} \mid x_{1:T}\right] \mathrm{d}x_{1:T} \\
&\quad - \int \gamma_T(x_{1:T})\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\log \widehat{Z}_T^{-1} \mid x_{1:T}\right] \mathrm{d}x_{1:T} \\
&= \mathbb{E}_{\mathrm{Q_{SMC}}}\left[-\log \widehat{Z}_T\right] + \mathbb{E}_{\gamma_T}\left[\mathbb{E}_{\mathrm{Q_{CSMC}}}\left[\log \widehat{Z}_T \mid x_{1:T}\right]\right],
\end{aligned}
$$

where the first equality is the definition. The second equality is a simplification of the previous expression where the terms cancelling, $\widetilde{\gamma}_T, \gamma_T, Z_T$, have been removed. Next, the inequality follows from the (conditional) Jensen's inequality for the convex functions $x \log x$ and $-\log x$, respectively. The final equality follows from Proposition 4, noting that the first term is equivalent to an expectation with respect to the full SMC distribution $\mathrm{Q_{SMC}}$.

The Jeffrey's divergence bound between two generic expected SMC empirical distributions in Eq. (123) is a straightforward extension to the above derivation.

# 6 Discussion

We have described sequential Monte Carlo, methods that use random samples to approximate the posterior. The goal is to approximate the distribution of the latent (unobserved) variables given the data. The idea is to draw samples from a (simpler) proposal distribution, then correct for the discrepancy between the proposal and the posterior using weights. SMC is generally applicable and can be considered to be an alternative, or complement, to MCMC methods.

First, we discussed the roots of SMC in (sequential) importance sampling and sampling/importance resampling. Next, we introduced the SMC algorithm in its full generality. The algorithm was illustrated with synthetic data from a non-Markovian latent variable model. Then, we discussed practical issues and theoretical results. The SMC estimate satisfies favorable theoretical properties; under appropriate assumptions it is both consistent and has a central limit theorem. Then, we discussed and explained how to learn a good proposal and target distribution. As a motivating example, we described how to go about learning a proposal- and target distribution for a stochastic recurrent neural network. Next, we discussed *proper weighting* and how to use SMC to construct nested estimators based on several layers of MC approximations to approximate idealized MC algorithms. We provided a straightforward proof to the unbiasedness property of the SMC normalization constant estimate. We studied PMH algorithms, methods that construct high-dimensional proposals for MH based on SMC samplers. We studied random and nested SMC methods that deal with intractability in the weights and proposal distributions with applications to stochastic differential equations and optimal proposal approximation. We illustrated how to leverage proper weighting and SMC to design algorithms that makes use of distributed and multi-core computing architectures. Finally, we introduced conditional sequential Monte Carlo algorithms and applications of these. We provided proof that the CSMC sampler obtains unbiased estimates of the *inverse* normalization constant. We showed how the IPMCMC method combines SMC and CSMC to design a powerful MCMC method that can take advantage of parallel computing. As a motivating example we show how SMC and CSMC can be used to evaluate inference in the context of approximate Bayesian inference.

There are several avenues open for further work and research in SMC methods and their application. We provide a few examples below. The proposal distribution is crucial for performance, and finding efficient ways to learn useful proposals is an area of continued research. The twisting potential can drasti-

cally improve accuracy. However, jointly learning twisting potentials and proposal distributions for complex models is a challenging and largely unsolved problem.

A rather obvious—but still very interesting and important—direction consists in adapting the SMC algorithm to parallel and distributed computing environments. This is indeed an active area of research, see e.g. Lee and Whiteley [2016], Lee et al. [2010]. However, we have reason to believe that there is still a lot that can be done in this direction. Just to give a few concrete examples we mention the fact that we can assemble SMC algorithm in new ways that we directly designed for parallel and distributed computing. As a concrete example we mention the smoother developed by Jacob et al. [2019] which is based on the clever debiasing scheme of Glynn and Rhee [2014]. Perhaps the divide and conquer SMC algorithm [Lindsten et al., 2017] can be useful in this effort. Its successful implementation within a probabilistic programming environment with support for parallel computation might give rise to an interesting scalable use of SMC.

An important observation is that SMC is increasingly being *used as a component* in various larger (often iterative) algorithms. Just to give a few examples of this we have Bayesian learning, where the particle filter is used within a Markov chain Monte Carlo algorithm [Andrieu et al., 2010, Lindsten et al., 2014], iterated filtering [Ionides et al., 2015] and maximum likelihood using expectation maximization [Schön et al., 2011]. The idea of using SMC within itself has occurred in several forms, including SMC$^2$ by Chopin et al. [2013] and nested SMC by Naesseth et al. [2015]. There are still most likely many interesting algorithms to uncover by following this idea in new directions.

There has recently been a very interesting line of research when it comes to merging deep learning architectures—such as recurrent neural networks, temporal convolutional networks and gated recurrent units—with stochastic latent models—such as the state space model—for sequence modelling, see e.g. [Bai et al., 2018, Aksan and Hilliges, 2019, Osendorfer and Bayer, 2014, Fraccaro et al., 2016]. An important idea here is to combine the expressiveness of the deep neural networks with the stochastic latent variable model's ability to represent uncertainty. These models are often trained by optimizing the ELBO objective. It has recently been shown that we can exploit the fact that the SMC estimator of the likelihood is unbiased to construct tighter lower bounds of this objective [Le et al., 2018, Maddison et al., 2017, Naesseth et al., 2018]. This can most likely open up for better training of combinations of deep learning architectures and stochastic latent models.

Given that we have now understood that the SMC methods are much more generally applicable than most of us first thought we foresee an interesting and creative future for these algorithms. We hope that this paper can serve as a catalyst for further research on these methods and in particular on their application in machine learning.

## Acknowledgments

This research was financially supported by the Swedish Foundation for Strategic Research (SSF) via the projects *ASSEMBLE* (contract number: RIT15-0012) and *Probabilistic Modeling and Inference for Machine Learning* (contract number: ICA16-0015), and by the Swedish Research Council via the projects *NewLEADS - New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079) and *Learning of Large-Scale Probabilistic Dynamical Models* (contract number: 2016-04278).

## References

E. Aksan and O. Hilliges. STCN: stochastic temporal convolutional networks. In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

BDO Anderson and JB Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

Christophe Andrieu, Gareth O Roberts, et al. The pseudo-marginal approach for efficient monte carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

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.

M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

Alexandros Beskos, Omiros Papaspiliopoulos, Gareth O Roberts, and Paul Fearnhead. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion). *Journal of the*

*Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):333–382, 2006.

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

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer-Verlag New York, 2005.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

Nicolas Chopin. Central limit theorem for sequential monte carlo methods and its application to bayesian inference. *The Annals of Statistics*, 32:2385–2411, 2004.

Nicolas Chopin, Pierre E Jacob, and Omiros Papaspiliopoulos. SMC$^2$: an efficient algorithm for sequential analysis of state-space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.

Julien Cornebise, Éric Moulines, and Jimmy Olsson. Adaptive methods for sequential importance sampling with application to state space models. *Statistics and Computing*, 18(4):461–480, 2008.

Marco F. Cusumano-Towner and Vikash K. Mansinghka. AIDE: An algorithm for measuring the accuracy of probabilistic inference algorithms. In *Advances in Neural Information Processing Systems*, pages 3004–3014. 2017.

Johan Dahlin, Fredrik Lindsten, Joel Kronander, and Thomas B Schön. Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. *arXiv:1511.05483*, 2015.

Pierre Del Moral. *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Springer Verlag, 2004.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

George Deligiannidis, Arnaud Doucet, and Michael K Pitt. The correlated pseudo-marginal method. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(5):839–870, 2018.

Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 64–69. IEEE, 2005.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704): 3, 2009.

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

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

P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14:11–21, 2004.

Paul Fearnhead and Hans R Künsch. Particle filters and data assimilation. *Annual Review of Statistics and Its Application*, 5:421–449, 2018.

Paul Fearnhead, Omiros Papaspiliopoulos, Gareth O Roberts, and Andrew Stuart. Random-weight particle filtering of continuous time processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):497–512, 2010.

M. Fraccaro, S. K. Sø nderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Mathieu Gerber, Nicolas Chopin, and Nick Whiteley. Negative association, ordering and convergence of resampling methods. *arXiv:1707.01845*, 2017.

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36, 2014.

P. W. Glynn and C.-H. Rhee. Exact estimation for markov chain equilibrium expectations. *Journal of Applied Probability*, 51A:377–389, 2014.

Andrew D Gordon, Thomas A Henzinger, Aditya V Nori, and Sriram K Rajamani. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*, pages 167–181. ACM, 2014.

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

Roger B Grosse, Zoubin Ghahramani, and Ryan P Adams. Sandwiching the marginal likelihood using bidirectional monte carlo. *arXiv:1511.02543*, 2015.

Roger B Grosse, Siddharth Ancha, and Daniel M Roy. Measuring the reliability of MCMC inference with bidirectional Monte Carlo. In *Advances in Neural Information Processing Systems 29*, pages 2451–2459. Curran Associates, Inc., 2016.

Shixiang Gu, Zoubin Ghahramani, and Richard E Turner. Neural adaptive sequential monte carlo. In *Advances in Neural Information Processing Systems*, pages 2629–2637, 2015.

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

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. PixelVAE: A latent variable model for natural images. In *International Conference on Representation Learning*, 2017.

Johannes Edmund Handschin and David Q Mayne. Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9(5):547–559, 1969.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Jeremy Heng, Adrian N Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential Monte Carlo. *arXiv:1708.08396*, 2017.

N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, editors. *Bayesian Nonparametrics*. Cambridge University Press, 2010.

Jeroen D Hol, Thomas B Schon, and Fredrik Gustafsson. On resampling algorithms for particle filters. In *IEEE Nonlinear Statistical Signal Processing Workshop*, pages 79–82. IEEE, 2006.

Jonathan H Huggins and Daniel M Roy. Sequential monte carlo as approximate sampling: bounds, adaptive resampling via $\infty$-ess, and an application to particle gibbs. *arXiv:1503.00966*, 2015.

E. L. Ionides, D. Nguyen, Y. Atchadé, S. Stoev, and A. A. King. Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. *Proceedings of the National Academy of Sciences*, 112(3):719–724, 2015.

P. E. Jacob, F. Lindsten, and T. B. Schön. Smoothing with couplings of conditional particle filters. *Journal of the American Statistical Association (JASA)*, 2019.

Pierre E. Jacob and Alexandre H. Thiery. On nonnegative unbiased estimators. *The Annals of Statistics*, 43(2):769–784, 04 2015.

Pierre E Jacob, Lawrence M Murray, and Sylvain Rubenthaler. Path storage in the particle filter. *Statistics and Computing*, 25(2):487–496, 2015.

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

Simon J Julier. The scaled unscented transformation. In *Proceedings of the 2002 American Control Conference*, volume 6, pages 4555–4559. IEEE, 2002.

Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics, 1997.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

G. Kitagawa. A Monte Carlo filtering and smoothing method for non-Gaussian nonlinear state space models. In *Proceedings of the 2nd US-Japan joint Seminar on Statistical Time Series Analysis*, pages 110–131, 1993.

Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *AAAI*, pages 2101–2109, 2017.

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

A. Lee and N. Whiteley. Forest resampling for distributed sequential Monte Carlo. *Journal of statistical analysis and data mining*, 9(4):230–248, 2016.

A. Lee, C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 10(4): 769–789, 2010.

Ming Lin, Rong Chen, Jun S Liu, et al. Lookahead strategies for sequential monte carlo. *Statistical Science*, 28(1):69–94, 2013.

F. Lindsten and A. Doucet. Pseudo-marginal Hamiltonian Monte Carlo. arXiv.org, arXiv:1607.02516, 2016.

F. Lindsten, A. Johansen, C. A. Naesseth, B. Kirkpatrick, T. B. Schön, J. Aston, and A. Bouchard-Côté. Divide-and-conquer with sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

Fredrik Lindsten and Thomas B Schön. Backward simulation methods for monte carlo statistical inference. *Foundations and Trends® in Machine Learning*, 6(1):1–143, 2013.

Fredrik Lindsten, Michael I Jordan, and Thomas B Schön. Particle gibbs with ancestor sampling. *The Journal of Machine Learning Research*, 15(1):2145–2184, 2014.

Fredrik Lindsten, Jouni Helske, and Matti Vihola. Graphical model inference: Sequential Monte Carlo meets deterministic approximations. In *Advances in Neural Information Processing Systems 31*, pages 8201–8211. Curran Associates, Inc., 2018.

Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems*, pages 6446–6456. Curran Associates, Inc., 2017.

C. J. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Whye Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, 2017.

Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

I. Murray and M. M. Graham. Pseudo-marginal slice sampling. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Cadiz, Spain, 2016.

Christian A Naesseth, Fredrik Lindsten, and Thomas B. Schön. Sequential Monte Carlo for Graphical Models. In *Advances in Neural Information Processing Systems 27*, pages 1862–1870. Curran Associates, Inc., Montreal, Canada, 2014.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Nested sequential monte carlo methods. In *International Conference on Machine Learning (ICML)*, pages 1292–1301, 2015.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. High-dimensional filtering using nested sequential monte carlo. *arXiv:1612.09162*, 2016.

Christian A. Naesseth, Francisco Ruiz, Scott W. Linderman, and David M. Blei. Reparameterization gradients through acceptance-rejection sampling algorithms. In *Artificial Intelligence and Statistics (AISTATS)*, pages 489–498, 2017.

Christian A. Naesseth, Scott Linderman, Rajesh Ranganath, and David M. Blei. Variational sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pages 968–977. PMLR, 2018.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11 (2):125–139, 2001.

C. Osendorfer and J. Bayer. Learning stochastic recurrent networks. Technical report, arXiv:1411.7610, 2014.

Brooks Paige and Frank Wood. Inference networks for sequential monte carlo in graphical models. In *International Conference on Machine Learning*, pages 3040–3049, 2016.

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

Tom Rainforth, Christian Naesseth, Fredrik Lindsten, Brooks Paige, Jan-Willem Van de Meent, Arnaud Doucet, and Frank Wood. Interacting particle markov chain monte carlo. In *International Conference on Machine Learning*, pages 2616–2625, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 09 1951.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2004.

Donald B. Rubin. Comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The sir algorithm. *Journal of the American Statistical Association*, 82(398):543–546, 1987.

Francisco Ruiz, Michalis Titsias, and David Blei. The generalized reparameterization gradient. In *Advances in neural information processing systems*, pages 460–468, 2016.

T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, January 2011.

Leland Stewart and Perry McCarty, Jr. Use of Bayesian belief networks to fuse continuous and discrete information for target recognition, tracking, and situation assessment. In *Proc. SPIE*, volume 1699, pages 177–185, 1992.

Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric A Wan. The unscented particle filter. In *Advances in neural information processing systems (NIPS)*, pages 584–590, 2001.

Christelle Vergé, Cyrille Dubarry, Pierre Del Moral, and Eric Moulines. On parallel implementation of sequential Monte Carlo methods: the island particle model. *Statistics and Computing*, 25(2):243–260, 2015.

Wolfgang Wagner. Unbiased monte carlo evaluation of certain functional integrals. *Journal of Computational Physics*, 71(1):21–33, 1987.

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

Frank Wood, Jan Willem Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pages 1024–1032, 2014.