Linköping Studies in Science and Technology, Dissertations. No 1969

Machine learning using approximate inference

Variational and sequential Monte Carlo methods

Christian Andersson Naesseth

LINKÖPING UNIVERSITY

Cover illustration: The collection of particle ancestral paths in the fully adapted sequential Monte Carlo algorithm, with stratified resampling, applied to a linear Gaussian state space model.

Linköping studies in science and technology. Dissertations.
No. 1969

Machine learning using approximate inference: Variational and sequential Monte Carlo methods

Christian Andersson Naesseth

christian.a.naesseth@liu.se
www.control.isy.liu.se
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 978-91-7685-161-6 ISSN 0345-7524

Printed by LiU-Tryck, Linköping, Sweden 2018

To my family

Abstract

Automatic decision making and pattern recognition under uncertainty are difficult tasks that are ubiquitous in our everyday life. The systems we design, and technology we develop, requires us to coherently represent and work with uncertainty in data. Probabilistic models and probabilistic inference gives us a powerful framework for solving this problem. Using this framework, while enticing, results in difficult-to-compute integrals and probabilities when conditioning on the observed data. This means we have a need for *approximate inference*, methods that solves the problem approximately using a systematic approach. In this thesis we develop new methods for efficient approximate inference in probabilistic models.

There are generally two approaches to approximate inference, variational methods and Monte Carlo methods. In Monte Carlo methods we use a large number of random samples to approximate the integral of interest. With variational methods, on the other hand, we turn the integration problem into that of an optimization problem. We develop algorithms of both types and bridge the gap between them.

First, we present a self-contained tutorial to the popular sequential Monte Carlo (SMC) class of methods. Next, we propose new algorithms and applications based on SMC for approximate inference in probabilistic graphical models. We derive nested sequential Monte Carlo, a new algorithm particularly well suited for inference in a large class of high-dimensional probabilistic models. Then, inspired by similar ideas we derive interacting particle Markov chain Monte Carlo to make use of parallelization to speed up approximate inference for universal probabilistic programming languages. After that, we show how we can make use of the rejection sampling process when generating gamma distributed random variables to speed up variational inference. Finally, we bridge the gap between SMC and variational methods by developing variational sequential Monte Carlo, a new flexible family of variational approximations.

Populärvetenskaplig sammanfattning

Fungerar den nya medicinen som vi har utvecklat? Vilket spel, film eller bok ska vi rekommendera härnäst? Bör jag investera nu eller bör jag vänta? Dessa är några exempel på frågor som man kan svara på med hjälp av *maskininlärning* (eng. machine learning). Maskininlärning handlar om metoder för att få en dator att automatiskt lära sig något från insamlad data. Data kan vara lite allt möjligt som man kan spara på en dator, alltifrån aktuell aktiekurs till vem man känner på Facebook. Datorn lär sig sen oftast en matematisk modell som beskriver datan. Med denna matematiska modell kan man: i) studera underliggande strukturer och mekanismer (t.ex. "hur påverkar tryck och temperatur vädret?"), ii) förutsäga hur framtida data kan se ut (t.ex. "kommer det regna imorgon?"), och iii) ta beslut (t.ex. "det är inte troligt att det regnar imorgon, så vi behöver inte ta med ett paraply").

Matematiska modeller används överallt inom teknologin och vetenskapens alla grenar. I många fall byggs modeller baserat på data som är insamlad under osäkra förhållanden eller med en i grunden slumpmässig variation. Dessa förhållanden lämpar sig väl för att jobba med sannolikheter, ett matematiskt koncept som blivit centralt i modern statistik och maskininlärning. Att lära sig och använda matematiska modeller baserat på sannolikhetsteori kallas för *statistisk inferens*. I många tillämpningar är datan vi samlat in för storskalig och modellen vi byggt för komplicerad för att exakt uträkningar ska vara möjliga. Detta betyder att vi måste använda och införa systematiska approximationer, vi utför *approximativ inferens*.

I denna avhandling studerar vi och utvecklar flera olika metoder för approximativ inferens med användning inom maskininlärning. Vi presenterar en introduktion till en populär klass av metoder som är speciellt användbara om ens matematiska modell beskriver data som varierar i tiden, till exempel vädret. Vi presenterar nya strategier som på flera sätt underlättar och effektiviserar den automatiska lärande-proccesen så att vi enklare kan hantera storskalig data och bättre, mer avancerade, modeller.

Acknowledgments

These five+ years have been a truly enjoyable experience. I have had the pleasure of getting to know, and collaborate with, many wonderful people.

First of all, I would like to thank my amazing supervisors Fredrik Lindsten and Thomas Schön. Thank you both for your positive attitudes, support, and the energy you put into helping me during my graduate studies. Fredrik, with your depth of knowledge, attention to detail, and creativity, it has been a pleasure to have you as my supervisor. And Thomas, thanks for helping with the big picture, inspiration and coaching.

A good working environment is important. I would like to thank former head of division Svante Gunnarsson, current head of division Martin Enqvist, and Ninna Stensgård for their helpfulness and diligence in this respect. Thanks also go out to all my great current and former colleagues! Especially (in no particular order) Jonas Linder, Sina Khoshfetrat Pakazad, Hanna Nyqvist, Zoran Sjanic, Daniel Petersson, André Carvalho Bittencourt, Patrik Axelsson, Manon Kok, Martin Skoglund, Tohid Ardeshiri, Niklas Wahlström, Johan Dahlin, Clas Veibäck, Martin Lindfors, and everyone else; thank you all for everything from the entertaining fika-discussions and BBQs, to the enjoyable board game evenings. Thanks also to CADICS, a Linnaeus Center funded by the Swedish Research Council, for providing the funds for my position.

During my graduate studies I have had the privilege to be able to go for several research visits. Thank you Frank Wood for inviting me to visit your amazing research group at Oxford University. During this visit, I had a great time working with and getting to know Tom Rainforth (thanks also for letting me use your spare room!), Brooks Paige, and Jan-Willem van de Meent. Thanks David Blei for letting me spend time as a visiting student researcher in your awesome group at Columbia University. While there I had the pleasure of collaborating with and getting to know Francisco Ruiz, Scott Linderman, Rajesh Ranganath, and Alp Kucukelbir. Thanks also to Francisco and Maja for saving the world together from certain doom! Another big thanks to Sebastian Nowozin for inviting me to Microsoft Research Cambridge for the summer. Working together with you, Sebastian Tschiatschek, and Jan Stuehmer was a great experience.

Thank you to my proofreaders Martin Lindfors, Andreas Svensson, Carl Andersson, Jalil Taghia, Fredrik Lindsten, and Thomas Schön. While I have tried my best to keep the thesis as error-free as possible, all remaining errors are entirely my own.

A big thanks go out to my family of course. Thanks to my parents, Elizabeth Naesseth and Ingemar Andersson, and siblings, Malin Fridh, Daniélle Andersson Bredenberg and Andree Andersson Bredenberg, for your support and encouragement; I love you all. I especially want to thank my mom, for always being there for me and for all the love you have always given me.

Last but not least, thank you to my wonderful wife Ziwei Naesseth Hu, for existing, for your support, and for being awsome :) Love you more than I can express!

Linköping, November 2018
Christian Andersson Naesseth

Contents

II Publications

Part I

Background

1

Introduction

Data and machine learning are by now an integral part of our everyday lives. Everytime we conduct a web search, machine learning ensures personalized and progressively better search results. Watching movies or listening to music? Machine learning is there to provide you with automatic recommendations on what to look at or listen to next. Need that sentence translated from English to Swedish? No problem! Use deep learning-powered machine translation. Machine learning is having an impact on everything from healthcare to video games. Wherever we have access to complex data, there is the potential that machine learning can be useful.

We are creating data at an unprecedented speed, quintillions of bytes *every day*. This is way more than any person could ever process in a lifetime. Machine learning instead uses computers and computer programs to help us make sense of the data. The increasing scale and complexity requires ever more efficient methods to process the data effectively.

With this motivation, the research in this thesis focuses on increasing the machine learning expert's toolbox and understanding of the tools available. We focus on stuyding the class of sequential Monte Carlo (SMC) methods for use in machine learning. We introduce the method in a tutorial article, study new applications within e.g. information theory and graphical models, develop methodological advances to SMC, and connect it to variational methods.

1.1 Data and machine learning

We begin by giving some general background to what we mean when we talk about *data* and *(machine) learning*.

1.1.1 Data

Data is arguably the most important ingredient in machine learning. We need to study data to learn patterns and make predictions. Basically anything that can be stored or recorded can be considered to be data. Whilst data can have near arbitrary form, in this thesis we ultimately only consider data that is well-represented by numbers. To be concrete we provide below a few of the examples that appear in some form or other in Part II:

Images: By using intensity for grayscale or RGB values for color, we can represent images using numbers. One example that we study is how machine learning can be used to generate images. (see Paper G)

Text: We can represent text using e.g. integer values, corresponding to a word's location in a dictionary. We study, amongst other things, how we can automatically categorize and sort documents from just their textual content. (see Papers C and G)

Brain activity: One common way to represent data from a biological neural network is to use spikes of brain activity, e.g. the number of neurons firing as a function of time. We study dynamical models for motor cortex neurons in a macaque monkey. (see Paper H)

Forex: The exchange rate between two currencies is easily represented by a numerical value. We study the *volatility*, or degree of variation, of the exchange rate between the US dollar and various other currencies. (see Paper H)

Precipitation: Average amount of precipitation in a given location during a given time period can also be represented using numerical values. We use observations of monthly average precipitation values collected over decades to detect extended periods of drought in North America and sub-Saharan Africa. (see Paper D)

To illustrate new methods for machine learning we also make liberal use throughout this thesis of artificially generated (simulated) data. Sometimes the data may mimick properties of real data, such as the soil carbon data from Paper E. Other times, the simulated data is purely artificial and only used to illustrate and profile the properties of the proposed algorithm, such as in Paper B.

1.1.2 Machine learning

Learning can be thought of as distilling information, or improving performance at a task, based on data. *Machine learning* simply means there is a machine (or computer program) that, with more or less automation, does the actual learning.

One of the most succesful coherent approaches to machine learning in the presence of uncertainty is probabilistic machine learning (Ghahramani, 2015). Random variables and probabilities are used to relate the data to a mathematical model. The model includes *latent variables*, variables that we do not observe directly but nevertheless are interested in knowing the values of. We make use of *inference* to deduce these values based on the model and data. Based on the results of inference, we can take rational *decisions* or actions. We illustrate with a simple example where our data consists of recorded results of a sequence of coin flips (heads/tails):

Model: The model is a probability distribution describing the relation between data and (unobserved) latent variables.

Example: The latent variable is the probability of a flip resulting in heads.

Inference: We deduce the value, or range of potential values, for the latent variables given our observed data.

Example: What is our best guess for the value, or range of likely values, of the probability of heads?

Decision: Based on the inference result and available choices, we take a rational decision on the best action.

Example: Is the coin fair enough, probability of heads is close to one half, or should we use another coin?

Inference typically leads to mathematical equations that we can not write down a closed-form solution to. This means that we have use for methods that solve the inference task approximately, so called *approximate inference* methods.

We return to these subjects in Chapter 2 and Chapter 3, where we expand and formalize the concepts briefly introduced above.

1.2 Contributions

The main contribution of this thesis is developing new methods for approximate inference and learning in latent variable models.

This section will elaborate on the contributions, organized by different research themes. Each theme is introduces and summarized in its own subsection. Fur-

thermore, the relevant papers to each theme is presented, as well as the author of this thesis' contributions to each particular paper.

1.2.1 Elements of sequential Monte Carlo

Sequential Monte Carlo methods are a powerful tool for approximate inference. The first contribution in Part II of this thesis is a self-contained tutorial article on SMC methods from a machine learning perspective.

The tutorial has a distinct focus from previously published tutorials on the topic. It focuses on the SMC algorithm's ability to approximate the final smoothing or marginal distribution of the latent variable model. These types of methods have recently seen use within machine learning to everything from improving variational inference to reinforcement learning.

This tutorial complements Part I, the background for this thesis.

Relevant publications

> *Paper A:*
> Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Elements of sequential Monte Carlo. *Foundations and Trends in Machine Learning*, 2018b. (proposal accepted, manuscript in preparation).

The idea originated from the author of this thesis, and was subsequently refined in discussion with the co-authors. The majority of the article is written by the author of this thesis. All examples and code supplied have been implemented and written by the author of this thesis.

1.2.2 Sequential Monte Carlo for graphical models

A probabilistic graphical model (PGM) is a type of model where the conditional independencies of the joint probability distribution of the variables are described by edges in a graph. The graph structure allows for easier and stronger control on the type of prior information that the user can express. The main limitation of the PGM is that exact inference is typically intractable and approximate inference is difficult.

The relevant publications in Part II explores new applications for SMC inference to compute the capacity of two-dimensional information channels. The capacity of the channel can be expressed as the normalization constant of an undirected PGM. An example where this is potentially useful is in emerging technology such as optical data storage. Furthermore, a new way of applying SMC methods specifically suitable for generic PGMs is developed and studied. Potential applications

include sampling from undirected PGMs, a notoriously difficult problem, generating efficient unbiased estimates of normalization constants, and evaluating model fit.

Relevant publications

> *Paper B:*
> Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *IEEE Information Theory Workshop (ITW)*, pages 431–435, 2014a.

The idea originated from the author of this thesis, and was subsequently refined in discussion with the co-authors. The majority of the article is written by the author of this thesis. All empirical studies have been carried out by the author of this thesis.

> *Paper C:*
> Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1862–1870, 2014b.

The idea originated from Fredrik Lindsten and Thomas B. Schön. The majority of the article is written by the author of this thesis. All empirical results, apart from the algorithm implementation for the latent Dirichlet allocation example, are by the author of this thesis.

> Fredrik Lindsten, Adam M. Johansen, Christian A. Naesseth, Brent Kirkpatrick, Thomas B. Schön, John Aston, and Alexandre Bouchard-Côté. Divide-and-conquer with sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

The author of this thesis has made minor contributions to the empirical results and writing of this paper. Because the author of this thesis has only made minor contributions to this paper it is not included in Part II.

1.2.3 Nested sequential Monte Carlo

The key design choice in any SMC algorithm is without a doubt the proposal distribution. The locally optimal proposal distribution is the best local proposal distribution that uses data only in a causal fashion, i.e. only up until the current iteration. Unfortunately this distribution is typically unavailable and approximations must be made for a practical SMC algorithm.

The first two relevant papers within this theme in Part II introduce so called *exact approximations* to any proposal distribution, focusing on the locally optimal proposal. The key idea is to construct a *nested* Monte Carlo (MC) method that approximates the proposal distribution, and then choose weights and samples in

such a way such that the resulting estimate is still consistent. There is a significant methodological overlap in these two papers. However, they complement each other with distinct derivations, focus, and results.

The last relevant paper uses similar ideas as in nested MC methods, but now instead tackle the problem of parallelization of inference for generic probabilistic programming. A new algorithm, interacting particle Markov chain Monte Carlo, particularly suited for taking advantage of multi-core architecture is developed and studied.

Relevant publications

> *Paper D:*
> Christian Naesseth, Fredrik Lindsten, and Thomas Schön. Nested sequential Monte Carlo methods. In *International Conference on Machine Learning (ICML)*, pages 1292–1301, 2015a.

The idea originated from discussions between the co-authors of the paper. The majority of the article is written by the author of this thesis. The proof of Theorem 2 is by Fredrik Lindsten. All empirical results are by the author of this thesis.

> *Paper E:*
> Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. High-dimensional filtering using nested sequential Monte Carlo. *arXiv:1612.09162*, 2016.

The idea originated from discussions between the co-authors of the paper. The majority of the article is written by the author of this thesis. All theoretical and empirical results are by the author of this thesis.

> *Paper F:*
> Tom Rainforth, Christian A. Naesseth, Fredrik Lindsten, Brooks Paige, Jan-Willem Vandemeent, Arnaud Doucet, and Frank Wood. Interacting particle Markov chain Monte Carlo. In *International Conference on Machine Learning (ICML)*, pages 2616–2625, 2016.

The idea originated from discussions between the author of this thesis and Fredrik Lindsten, subsequently refined by the co-authors of the paper. The majority of the method development is written by the author of this thesis. All theoretical results are by the author of this thesis. The empirical results of Figure 1 are by the author of this thesis.

1.2.4 Variational Monte Carlo

Variational methods are another powerful tool for approximate inference, turning the integration problem in inference into an optimization problem which we can solve more efficiently. Classical approaches have relied on so-called mean

field approximations to the posterior to derive tractable coordinate ascent algorithms for the optimization procedure. To allow for more flexible and accurate posterior inferences one needs to resort to stochastic optimization.

The first paper and contribution within this theme develops low-variance reparameterization gradients for a class of variational approximations that rely on the rejection sampler for simulation. The second paper combines variational methods with SMC, viewing the (expected) posterior distribution approximation from SMC as the variational approximation we need to optimize. This leads to a more flexible and accurate distribution, trading off fidelity to the posterior with computational cost.

Relevant publications

> *Paper G:*
> 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.

The idea originated from discussions between the co-authors of the paper. The majority of the article is written by the author of this thesis. All theoretical and empirical results are by the author of this thesis.

> *Paper H:*
> Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei. Variational sequential Monte Carlo. In *Artificial Intelligence and Statistics (AISTATS)*, pages 968–977, 2018a.

The idea originated from the author of this thesis, and was subsequently refined in discussion with the co-authors. The majority of the article is written by the author of this thesis. All theoretical and empirical results are by the author of this thesis.

1.3 Thesis outline

The thesis is divided into two parts. The first part contains a brief introduction to machine learning from a probabilistic or statistical point of view, explaining the three central concepts *modeling*, *inference*, and *decision making*. The first part concludes with an introduction to the two main methods for performing approximate inference, *variational* and *Monte Carlo* methods. The second part contains edited versions of eight publications.

1.4 Publications

The author's published work is listed below in reverse chronological order. Publications indicated by a ⋆ are included in Part II of this thesis.

⋆ Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Elements of sequential Monte Carlo. *Foundations and Trends in Machine Learning*, 2018b. (proposal accepted, manuscript in preparation).

⋆ Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei. Variational sequential Monte Carlo. In *Artificial Intelligence and Statistics (AISTATS)*, pages 968–977, 2018a.

⋆ 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.

Fredrik Lindsten, Adam M. Johansen, Christian A. Naesseth, Brent Kirkpatrick, Thomas B. Schön, John Aston, and Alexandre Bouchard-Côté. Divide-and-conquer with sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

Sina Khoshfetrat Pakazad, Christian A. Naesseth, Fredrik Lindsten, and Anders Hansson. Distributed, scalable and gossip-free consensus optimization with application to data analysis. *arXiv:1705.02469*, 2017.

⋆ Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. High-dimensional filtering using nested sequential Monte Carlo. *arXiv:1612.09162*, 2016.

⋆ Tom Rainforth, Christian A. Naesseth, Fredrik Lindsten, Brooks Paige, Jan-Willem Vandemeent, Arnaud Doucet, and Frank Wood. Interacting particle Markov chain Monte Carlo. In *International Conference on Machine Learning (ICML)*, pages 2616–2625, 2016.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Towards automated sequential Monte Carlo for probabilistic graphical models. In *NIPS Workshop on Black Box Learning and Inference*, 2015b.

Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai. Sequential Monte Carlo methods for system identification. *IFAC-PapersOnLine (SYSID)*, 48(28):775–786, 2015.

⋆ Christian Naesseth, Fredrik Lindsten, and Thomas Schön. Nested sequential Monte Carlo methods. In *International Conference on Machine Learning (ICML)*, pages 1292–1301, 2015a.

⋆ Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1862–1870, 2014b.

⋆ Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *IEEE Information Theory Workshop (ITW)*, pages 431–435, 2014a.

2

Probabilistic machine learning

Machine learning is a science that focuses on the study and design of computer programs that learn automatically from data and experience. Probabilistic (or statistical) machine learning does this while explicitly representing uncertainty. With probabilistic machine learning we make use of probability theory to represent the uncertainty. Uncertainty, for the purpose of this thesis, can take many forms. It can be an inherent randomness in a system, or it could be about our own imperfect or partial information about the system we observe. This is also tightly connected to the definition of what probabilities and random variables actually are. We take a very pragmatic approach, making use of ideas from both Bayesian and frequentist perspectives on probability.

In this chapter we introduce some fundamentals to probabilistic machine learning: the concepts of *modeling*, *inference*, and *decision*. Modeling is the procedure of writing down explicit assumptions on how the data was generated, usually in the form of a probability distribution. Inference means taking our model and combining it with our observed data to reach a conclusion; perhaps to predict future data or answer a causal question. Decision is using our inferences to make a choice.

The field of machine learning is closely related to, amongst other fields, computational statistics (Efron and Hastie, 2016) and mathematical optimization (Bertsekas, 2016; Nocedal and Wright, 2006). Throughout this thesis, we will see how making use of methods from these fields will empower us with more powerful methods for machine learning.

This chapter is intentionally kept quite brief. For a more thorough introduction to probabilistic and statistical machine learning, we refer the reader to the textbooks by Bishop (2006) or Friedman et al. (2009).

2.1 Modeling

A model describes data from a system that we observe. We will focus on the use of probabilistic models: mathematical models making use of probability distributions to encode uncertainty. The key variables of the model are the data $\mathbf{y}$ and the parameters θ, a type of latent (unobserved) variable. The data $\mathbf{y}$ usually consists of T separate observations $\mathbf{y} = \{y_t\}_{t=1}^T$ indexed by t.

To prior, or not to prior One fundamental consideration is whether we define a joint probability of both data and parameters, i.e. $p(\mathbf{y}, \theta) = p(\mathbf{y} \mid \theta)p(\theta)$, or if we focus solely on the *likelihood* $p(\mathbf{y} \mid \theta)$. Pragmatically the two differ in the prior $p(\theta)$ and the interpretation of the parameters. For frequentist statistics we are interested in an estimator of θ that works well for repeated use, where we could potentially observe many different realisations of $\mathbf{y}$. In Bayesian statistics, we assign a prior *degree of belief* for the potential values of the parameter θ and interpret the parameter as a random variable. Then based on the observed data, a particular *realization* of $\mathbf{y}$, we update our belief about the parameter θ. Bayesian statistics is usually criticized for being subjective because of the prior, which introduces preferences for different values of the parameters. These priors can be different for different people, meaning that the inference can also be different. However, the likelihood (common to both approaches) is also a subjective choice. Either way, we will see both of these types of interpretations on unknown parameters in Part II, with a focus on the Bayesian approach.

For a more thorough discussion of the fundamentals of the different schools of statistical inference we recommend Casella and Berger (2002); Gelman et al. (2013); Robert (2007).

Example We present a simple toy example in Example 2.1, the data is modeled as independent and identically distributed from a normal distribution with unknown mean. This example will be used throughout this chapter to illustrate the different concepts.

Example 2.1: Toy Example: Model

For illustrative purposes we will consider a very simple model for which inference and decision is usually analytically tractable. We simulate data from the following joint probability

$$p(\mathbf{y}, \theta) = p(\mathbf{y} \mid \theta)p(\theta) = \mathcal{N}(\theta \mid 0, 1) \cdot \prod_{t=1}^{3} \mathcal{N}(y_t \mid \theta, 1),$$

where $\mathcal{N}(\cdot \mid \mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2. We will use the dataset $y_1 = -0.65$, $y_2 = 0.072$, and $y_3 = -0.54$.

Local latent variables A common technique to make modeling and inference easier is introducing extra (local) latent variables, also known as *data augmentation* (Van Dyk and Meng, 2001). This means that we have extra latent variables $\mathbf{x} = \{x_t\}_{t=1}^T$, where each local latent variable x_t is often associated with a data point y_t. With this we can define the likelihood

$$p(\mathbf{y} \mid \theta) = \int p(\mathbf{y}, \mathbf{x} \mid \theta) \, \mathrm{d}\mathbf{x}, \tag{2.1}$$

where $p(\mathbf{y}, \mathbf{x} \mid \theta)$ is known as the complete data likelihood. We present three examples of model classes that are ubiquitous in practice:

- **Conditionally independent model:** In the conditionally independent model the complete data likelihood will satisfy the following factorization

$$p(\mathbf{y}, \mathbf{x} \mid \theta) = \prod_{t=1}^{T} p(x_t \mid \theta) p(y_t \mid x_t, \theta), \tag{2.2}$$

  i.e. the data are conditionally independent given θ. This is a common model for *exchangeable* data (Gelman et al., 2013): data y_t, $t = 1, \ldots, T$ where a reordering does not change the joint distribution $p(\mathbf{y})$. Examples of this type are used in Papers G and H.

- **State space model:** The local latent variables in a state space model (SSM) satisfy a Markov property (Cappé et al., 2005). This means that the prior for x_t only depends on its immediate preceding latent variable x_{t-1}, and the data y_t only directly depends on x_t. With these assumptions we get

$$p(\mathbf{y}, \mathbf{x} \mid \theta) = p(x_1 \mid \theta) p(y_1 \mid x_1, \theta) \prod_{t=2}^{T} p(x_t \mid x_{t-1}, \theta) p(y_t \mid x_t, \theta). \tag{2.3}$$

  Examples of this type are used in Papers E and H.

- **Probabilistic graphical model:** The local latent variables in a PGM (Koller et al., 2009) have a dependence defined by a graph. The complete data likelihood for the model, based on a graph with edges defined in the edge set $\mathcal{E}$, can be written as

$$p(\mathbf{y}, \mathbf{x} \mid \theta) = \frac{1}{Z(\theta)} \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j \mid \theta) \prod_{t=1}^{T} \phi(x_t, y_t \mid \theta), \tag{2.4}$$

  where $Z(\theta)$ is the normalization constant, ensuring that the right-hand side is a probability distribution. The positive functions ψ, ϕ are known as interaction and observation potentials, respectively. Examples of this type are used in Papers B, C, D, and E.

Local latent-variable-based models are mainstay modeling tool used throughout the field of probabilistic machine learning.

Mechanistic-algorithmic continuum One way to distinguish different models on a high level is on a mechanistic-algorithmic continuum. On the one side, purely mechanistic models are fully specified models based on physical or natural processes where the parameters have a physical or natural interpretation. An example of this type of model is Newton's second law of motion. On the other side, we have the purely algorithmic model where the parameters have no physical or natural interpretation and we are only interested in its predictive abilities. An example that comes close to this are models making use of (artificial) neural networks and deep learning. Machine learning methods tend to make use of models that fall closer to the algorithmic part of the spectrum. However, no matter where a specific model falls within this continuum, we want to infer its parameters and other unknown latent variables.

2.2 Inference

Inference means taking the observed data and combining it with our model assumptions to deduce properties on the latent variables. The goal of inference usually takes one of two forms: we are interested in either prediction or causality. Prediction means learning to forecast future values of new, as of yet unobserved, data. Causality focuses on understanding the parameter, the values it takes, and how it affects the data. Our main focus in this thesis is predictive inference based on the posterior distribution and the maximum likelihood estimator.

Posterior distribution When we have a joint probabilistic model for both data and parameters, inference is conceptually straightforward. By Bayes' rule, a fundamental result of conditional probabilities, the posterior distribution of θ given $\mathbf{y}$ is

$$p(\theta \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \theta) p(\theta)}{p(\mathbf{y})}, \tag{2.5}$$

where $p(\mathbf{y}) = \int p(\mathbf{y}, \theta) \, \mathrm{d}\theta$ is known as the marginal likelihood. The posterior distribution is the fundamental object for Bayesian inference. It is then used both in prediction and causality to compute expectations of functions $\mathrm{f}(\theta)$ with respect to the posterior,

$$\mathbb{E}_{p(\theta \mid \mathbf{y})} [\mathrm{f}(\theta)] = \int \mathrm{f}(\theta) p(\theta \mid \mathbf{y}) \, \mathrm{d}\theta. \tag{2.6}$$

For example, when we want to predict new potential values $y^\star$ for our data we can use the *posterior predictive distribution*. This distribution can be written as an expectation of $\mathrm{f}(\theta) = p(y^\star \mid \theta)$ with respect to the posterior distribution,

$$p(y^\star \mid \mathbf{y}) = \int p(y^\star \mid \theta) p(\theta \mid \mathbf{y}) \, \mathrm{d}\theta. \tag{2.7}$$

Bayesian statistics and the posterior distribution can be traced back to early work by the English statistician and reverend Thomas Bayes (1701–1761) and the French mathematician Pierre-Simon Laplace (1749–1827) (Bayes, 1763; Laplace, 1774; Stigler, 1986). Our current interpretation of Bayesian probability has its root in Laplace's extensive work on subjective probability.

Maximum likelihood When we only have access to the likelihood function, we can take several approaches to find a good value of the parameter. One of the most common and sensible approaches is to maximize the likelihood $p(\mathbf{y} \mid \theta)$ with respect to the parameters: the maximum likelihood parameter estimate $\widehat{\theta}^{\mathrm{ML}}$ is

$$\widehat{\theta}^{\mathrm{ML}} = \arg\max_{\theta} \log p(\mathbf{y} \mid \theta). \tag{2.8}$$

The logarithm is introduced purely for computational convenience. Since it is a monotone function of its argument, it does not change the maximizing argument $\widehat{\theta}^{\mathrm{ML}}$.

Prediction in the maximum likelihood framework focuses on using the likelihood evaluated at the maximum likelihood value for the parameters, i.e. $p(y^\star \mid \widehat{\theta}^{\mathrm{ML}})$.

The rise and popularization of frequentist statistics and maximum likelihood estimators can be traced back mainly to work by the British statistician Sir Ronald Fisher (1890–1962) during the early 20th century (Fisher, 1922). However, the principle and idea had previously been used by Hagen, Gauss, and Edgeworth (Hald, 1999).

Example We return to our Gaussian toy example to perform Bayesian and frequentist inference, see Example 2.2. We focus on the posterior distribution and maximimum likelihood estimate.

Example 2.2: Toy Example: Inference

Under the model assumptions of Example 2.1 we can analytically solve for the posterior distribution and maximum likelihood parameter:

- **Posterior distribution:**

$$p(\theta \mid \mathbf{y}) = \mathcal{N}\left(\theta \,\middle|\, \frac{1}{T+1}\sum_{t=1}^{T} y_t, \frac{1}{T+1}\right) \tag{2.9}$$

- **Maximum likelihood:**

$$\widehat{\theta}^{\mathrm{ML}} = \frac{1}{T}\sum_{t=1}^{T} y_t \tag{2.10}$$

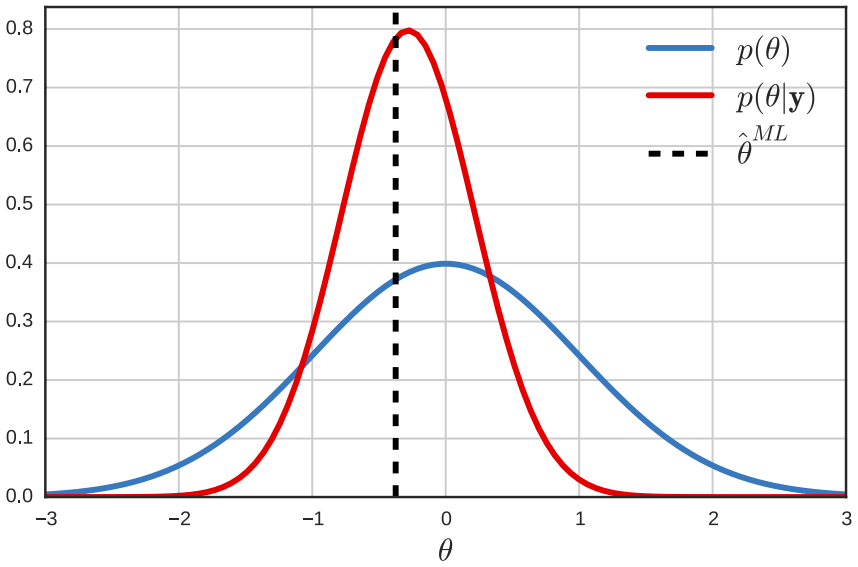


Figure 2.1: *The prior distribution $p(\theta)$, posterior distribution $p(\theta \mid \mathbf{y})$, and maximum likelihood estimate $\widehat{\theta}^{ML}$, for the dataset $y_1 = -0.65$, $y_2 = 0.072$, and $y_3 = -0.54$ based on our Gaussian model.*

With our example dataset $y_1 = -0.65$, $y_2 = 0.072$, and $y_3 = -0.54$ we illustrate the model, posterior distribution, and maximum likelihood estimator in Figure 2.1. The true value of the parameter that generated this particular dataset is $\theta^\star = -0.78$.

We have seen a simple example were exact inference is analytically tractable, i.e. closed-form solutions exist. However, this will not be the case in general. Most models and data that we encounter in practice will lead to an intractable inference problem, i.e. one where we can not evaluate the posterior distribution or find the maximum likelihood value. For the posterior distribution it is usually the marginal likelihood that is computationally intractable, because it requires us to evaluate a (potentially) high-dimensional integral, which is difficult. This means that we will have to resort to some form of approximation, which is the topic of Chapter 3 and the publications in Part II of this thesis.

2.3 Decision

Decision theory is associated with the notion of *risk* and *loss*. The typical setting is that we are interested in making a choice for the value of parameter θ, by picking an estimator $\delta(\mathbf{y})$ that approximates the parameter of interest as well as possible. To do this we first define the concepts of *loss* and *risk*.

The loss $\mathrm{L}(\theta, \delta)$ is a non-negative function of the parameter and decision. It lets us evaluate the penalty of taking the decision δ when the parameter is in fact θ. The risk $\mathrm{R}(\cdot)$ defines the average loss that we would like to optimize with respect to the decision δ. The risk differs between a frequentist and Bayesian interpretation, by changing what is interpreted as random and what is fixed.

Frequentist risk: In the frequentist approach to decision theory, we average the loss for the potential datasets we could have observed. The frequentist risk is defined by computing the expectation of our loss with respect to the likelihood, i.e.

$$\mathrm{R}(\theta, \delta) = \int \mathrm{L}(\theta, \delta(\mathbf{y}))p(\mathbf{y} \mid \theta)\, \mathrm{d}\mathbf{y}. \tag{2.11}$$

The frequentist risk is a function not only of our decision δ, but also of our (unknown) parameter θ. The actual observed data is not taken into account any further, instead we average over the likelihood and any potential dataset we could see if the experiment was repeated.

Bayes risk: With the Bayesian approach we instead integrate over our uncertainty on the parameters. The *posterior expected loss* is given by

$$\mathrm{r}(\delta(\mathbf{y}), \mathbf{y}) = \int \mathrm{L}(\theta, \delta(\mathbf{y}))p(\theta \mid \mathbf{y})\, \mathrm{d}\theta, \tag{2.12}$$

which is a function of the observed data $\mathbf{y}$ and our decision δ. The *Bayes estimator* is given by

$$\delta^{\mathrm{B}}(\mathbf{y}) := \arg\min_{\delta} \mathrm{r}(\delta(\mathbf{y}), \mathbf{y}), \tag{2.13}$$

for each dataset $\mathbf{y}$ that we could potentially observe. Unlike the frequentist risk, the posterior expected loss only depends on the model and dataset $\mathbf{y}$ (which are known).

The expected posterior loss in Equation (2.12) is the only risk we care about from a strictly Bayesian point of view. All calculations are conditional on our known data. However, with the prior we can connect the posterior expect loss to the frequentist risk by defining the *integrated risk*,

$$\mathrm{R}(\delta) = \int \mathrm{r}(\delta(\mathbf{y}), \mathbf{y})p(\mathbf{y})\, \mathrm{d}\mathbf{y} = \int \mathrm{R}(\theta, \delta)p(\theta)\, \mathrm{d}\theta, \tag{2.14}$$

where $p(\theta)$ is the prior distribution and $p(\mathbf{y})$ is the marginal likelihood. It is possible to show that δ^{B} is the estimator that minimizes the integrated risk (Robert, 2007). The *Bayes risk* is then defined by $\mathrm{R}(\delta^{\mathrm{B}})$. If the Bayes risk is finite we call δ^{B} *admissible*. It dominates and outperforms any other estimator in terms of frequentist risk.

For the reader interested in more information about the foundations of Bayesian decision theory, and its connection to the frequentist approach, we refer to Robert (2007).

Example Returning to the toy example we consider two loss functions: the squared error and absolute error losses. See Example 2.3 for a derivation of the optimal Bayesian decisions in this case.

Example 2.3: Toy Example: Decision

Using the model definition in Example 2.1 and the posterior distribution from Example 2.2, we can derive the optimal Bayes estimators for the two loss functions:

- **Squared error:** $\mathrm{L}(\theta, \delta) = (\theta - \delta(\mathbf{y}))^2$

  For the squared error loss we obtain the Bayes estimator

  $$\delta^{\mathrm{B}}(\mathbf{y}) = \int \theta p(\theta \,|\, \mathbf{y}) \,\mathrm{d}\theta, \tag{2.15}$$

  i.e. the posterior mean. For the dataset in our toy example, we get

  $$\delta^{\mathrm{B}}(\mathbf{y}) = \frac{1}{4} \sum_{t=1}^{3} y_t = -0.28.$$

- **Absolute error:** $\mathrm{L}(\theta, \delta) = |\theta - \delta(\mathbf{y})|$

  For the absolute error loss, we obtain the posterior median as the Bayes estimator

  $$\delta^{\mathrm{B}}(\mathbf{y}) = \hat{\delta}, \quad \text{where } \mathbb{P}(\theta \leq \hat{\delta} \,|\, \mathbf{y}) = \mathbb{P}(\theta \geq \hat{\delta} \,|\, \mathbf{y}) = \frac{1}{2}, \tag{2.16}$$

  where $\mathbb{P}(\theta \leq \hat{\delta} \,|\, \mathbf{y}) = \int_{-\infty}^{\hat{\delta}} p(\theta \,|\, \mathbf{y}) \,\mathrm{d}\theta$ is the probability that the parameter θ is smaller than $\hat{\delta}$ given the data $\mathbf{y}$. For the dataset in our toy example we get

  $$\delta^{\mathrm{B}}(\mathbf{y}) = \frac{1}{4} \sum_{t=1}^{3} y_t = -0.28,$$

  since the median and mean of the normal distribution coincide.

Decision theory does not feature prominently in this thesis; we focus instead on the task of approximate inference. However, as we can see above, being able to evaluate the posterior distribution (and expectations with respect to it) is key for Bayesian decision theory. This means that even in this setting, we have a need for efficient and accurate approximations to the posterior distribution. This is the main focus of most publications in Part II.

3

Approximate inference

Approximate inference is mainly focused on developing, and studying, approaches to estimate the posterior distribution. We can then make use of the approximation of the posterior and compute expectations with respect to it. We have seen from Chapter 2 that this is central to probabilistic machine learning. Approximate inference lets us trade off fidelity of the posterior approximation with computational complexity; the accuracy of the approximation typically depends on the amount of computation used.

Variational and Monte Carlo methods are two of the most popular approaches to approximate inference in machine learning. We will focus this chapter on introducing these two types of methods. First, we give a short introduction to Monte Carlo methods. These methods use (pseudo-)random numbers, usually referred to as *samples*, approximately distributed according to the posterior distribution. The samples are then averaged to estimate the posterior and posterior expectations. Second, we introduce and explain variational methods for approximate inference. Variational methods postulate a family of approximating distributions, e.g. the normal distribution family parameterized by the mean and variance. We then use a suitable cost function to optimize the parameters such that the variational approximation fits as close as possible to the true distribution.

This chapter is intentionally kept brief. For a more thorough introduction to Monte Carlo and variational methods, we refer the reader to Liu (2004); Robert and Casella (2004), Paper A, Blei et al. (2017), and Bishop (2006).

3.1 Monte Carlo methods

Monte Carlo methods are a class of algorithms relying on (pseudo-)random numbers to approximate high-dimensional integrals. As discussed in Eckhardt (1987), the roots of Monte Carlo methods can be found in work by the Polish–American scientist Stanislaw Ulam (1909–1984) and Hungarian–American mathematician John von Neumann (1903–1957) during the 1940s.

We will here focus on the basic idea behind the various MC methods, and explain rejection sampling which is used in Paper G. The remaining papers in Part II rely on importance sampling (IS) and SMC methods, which we give a thorough introduction to in Paper A.

3.1.1 The Monte Carlo idea

We have shown that probabilistic machine learning relies on performing inference. In the Bayesian approach this means that we are interested in estimating the posterior distribution or compute expectations with respect to it, i.e.

$$\mathbb{E}_{p(\theta \mid \mathbf{y})}[\mathrm{f}(\theta)] = \int \mathrm{f}(\theta) p(\theta \mid \mathbf{y}) \, \mathrm{d}\theta. \tag{3.1}$$

The key Monte Carlo idea (see e.g. Metropolis and Ulam (1949) for an early reference discussing the idea) is to draw samples, random numbers, that are either exactly or approximately distributed according to $p(\theta \mid \mathbf{y})$ and estimate the expectation by averaging

$$\mathbb{E}_{p(\theta \mid \mathbf{y})}[\mathrm{f}(\theta)] \approx \frac{1}{N} \sum_{i=1}^{N} \mathrm{f}(\theta^i), \quad \theta^i \sim p(\theta \mid \mathbf{y}), \; i = 1, \ldots, N. \tag{3.2}$$

We can view the samples $\{\theta^i\}_{i=1}^N$ as defining an empirical distribution that approximates the posterior,

$$p(\, \mathrm{d}\theta \mid \mathbf{y}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta_{\theta^i}(\, \mathrm{d}\theta), \tag{3.3}$$

where $\delta_{\theta^i}(\, \mathrm{d}\theta)$ is the Dirac measure at $\theta = \theta^i$.

This basic Monte Carlo method has many favorable properties. It is

- **unbiased:**

$$\mathbb{E}\left[\frac{1}{N} \sum_{i=1}^{N} \mathrm{f}(\theta^i)\right] = \mathbb{E}_{p(\theta \mid \mathbf{y})}[\mathrm{f}(\theta)], \tag{3.4}$$

- **consistent:**

$$\frac{1}{N}\sum_{i=1}^{N} \mathrm{f}(\theta^i) \xrightarrow{a.s.} \mathbb{E}_{p(\theta\,|\,\mathbf{y})}\left[\mathrm{f}(\theta)\right], \quad N \to \infty, \tag{3.5}$$

- **asymptotically normal:**

$$\frac{\sqrt{N}}{\sigma_{\mathrm{f}}}\left(\frac{1}{N}\sum_{i=1}^{N} \mathrm{f}(\theta^i) - \mathbb{E}_{p(\theta\,|\,\mathbf{y})}\left[\mathrm{f}(\theta)\right]\right) \xrightarrow{d} \mathcal{N}(0,1), \quad N \to \infty, \tag{3.6}$$

where $\xrightarrow{a.s.}$ and $\xrightarrow{d}$ denotes convergence almost surely and convergence in distribution, respectively. The asymptotic normality holds if the variance of the function $\mathrm{f}(\cdot)$ with respect to the posterior distribution $p(\theta\,|\,\mathbf{y})$, $\sigma_{\mathrm{f}}^2 = \mathrm{Var}_{p(\theta\,|\,\mathbf{y})}(\mathrm{f}(\theta))$, is finite. One of the fundamental strengths of MC methods is that the rate of improvement as a function of the number of samples N, illustrated by the central limit theorem (asymptotic normality), does not depend on the dimensionality of the parameter θ.

In practice it is often difficult or impossible to simulate exactly from the posterior distribution. In the next section we describe rejection sampling, a method that accomplishes exact simulation using auxiliary variables and samples from a different distribution.

3.1.2 Rejection sampling

Rejection sampling (Devroye, 1986; von Neumann, 1951), or acceptance-rejection sampling, is a method for exact simulation. In most cases we use a rejection sampler to generate random samples from a distribution for which standard sampling methods, such as inverse transform sampling (Robert and Casella, 2004), are unavailable or impractical. For notational convenience we will simply denote any distribution we are interested in generating samples from as $\gamma(\theta)$. We will refer to this as the *target distribution*.

The fundamental idea in rejection sampling is based on the straightforward idea that we can rewrite any density as an integral

$$\gamma(\theta) = \int_0^{\gamma(\theta)} 1 \,\mathrm{d}u, \tag{3.7}$$

for some *auxiliary variable* u. This means that the distribution $\gamma(\theta)$ is the marginal density of the uniform distribution on $\{(\theta, u) : 0 < u < \gamma(\theta)\}$, which we denote by $\mathrm{U}(\{(\theta, u) : 0 < u < \gamma(\theta)\})$. This corresponds to putting a uniform distribution on the area under the graph defined by the θ-axis and $\gamma(\theta)$, see the shaded area in Figure 3.1 for an example.

Now, if we could sample $(\theta, u) \sim \mathrm{U}(\{(\theta, u) : 0 < u < \gamma(\theta)\})$ we would have essentially solved the problem. However, directly trying to sample this distribution can be difficult. The straightforward way would be $\theta \sim \gamma(\theta)$ and $u \sim \mathrm{U}(0, \gamma(\theta))$. Because this approach relies on sampling θ from $\gamma(\theta)$ it defeats the purpose. What we can do instead is to sample uniformly over a larger area that covers the one we are interested in. Then we simply keep only the ones that fall within the area of interest defined by the constraint $0 < u < \gamma(\theta)$ and reject the rest.

To accomplish this we need to make use of a *proposal distribution* $q(\theta)$. We assume that the proposal is simple to sample from, that we can evaluate $q(\theta)$, and that we can find a finite constant $M \geq 1$ such that $\gamma(\theta) \leq Mq(\theta)$ for all values θ. We generate samples from the distribution

$$(\theta', u') \sim \mathrm{U}(\{(\theta, u) : 0 < u < Mq(\theta)\}), \tag{3.8}$$

simply by first simulating $\theta' \sim q(\theta)$, and then $u' \sim \mathrm{U}(0, Mq(\theta'))$. Then we accept the sample (θ', u') if $u' < \gamma(\theta')$, and otherwise repeat the procedure. We summarize: repeat

$$\theta' \sim q(\theta), \quad u' \,|\, \theta' \sim \mathrm{U}(0,\, Mq(\theta')), \tag{3.9}$$

until $u' < \gamma(\theta')$. The accepted value θ' is then distributed $\theta' \sim \gamma(\theta)$ as required. We only have to know the desired $\gamma(\theta)$ up to a normalization constant Z, i.e. $\gamma(\theta) = \frac{1}{Z}\tilde{\gamma}(\theta)$. The normalization constant Z may be subsumed into the factor M. In Bayesian inference we would have $\tilde{\gamma}(\theta) = p(\mathbf{y}, \theta)$, and the normalization constant $Z = p(\mathbf{y})$ would be subsumed by our choice of M such that $p(\mathbf{y}, \theta) \leq Mq(\theta)$.

We illustrate rejection sampling with a simple example.

Example 3.1: Rejection sampling

We consider a simple scalar example. The probability distribution $\gamma(\theta)$ is defined by

$$\gamma(\theta) = \frac{1}{Z} \underbrace{e^{-\frac{1}{2}\theta^2}\left(1 + \sin^2(4\theta) + 3\cos^2(\theta)\sin^2(\theta)\right)}_{\tilde{\gamma}(\theta)}, \tag{3.10}$$

and we use the standard normal, which we can easily generate samples from, as our proposal distribution

$$q(\theta) = \mathcal{N}(\theta \,|\, 0, 1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\theta^2}. \tag{3.11}$$

If we choose $M = 2$ we have that $\gamma(\theta) \leq M \cdot q(\theta)$ for all θ. We illustrate the setup in Figure 3.1. We found this value of M by numerically computing Z. If we only assume that we can evaluate the unnormalized distribution $\tilde{\gamma}(\theta)$, we could instead choose e.g. $M = 5\sqrt{2\pi}$. Higher values for M will give rise to a lower acceptance probability, requiring us to propose more samples (on average).

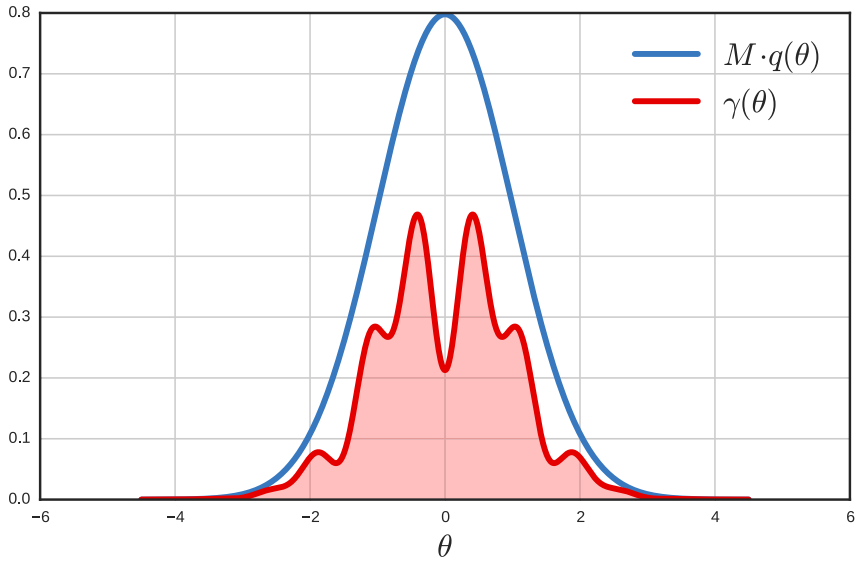


Figure 3.1: *Illustration of rejection sampling with target* $\gamma(\theta) \propto e^{-\frac{1}{2}\theta^2}\left(1 + \sin^2(4\theta) + 3\cos^2(\theta)\sin^2(\theta)\right)$*, proposal* $q(\theta) = \mathcal{N}(\theta \mid 0, 1)$*, and constant* $M = 2$*.*

One of the main limitations of rejection sampling is the requirement to find the constant M and proposal $q(\theta)$ that satisfies the constraints and results in few rejected samples. The computational complexity of rejection sampling tends to scale poorly with the dimension of θ. For these reasons, improving on rejection sampling is key to more efficient approximate inference.

3.2 Variational inference

Variational methods, also known as variational Bayes, turn the problem of integrating over the parameters (to find the marginal likelihood) into an optimization problem. The solution to the optimization problem results in a simpler distribution that we can efficiently work with. This distribution is chosen so that the discrepancy between it and the posterior is as small as possible. We trace the beginnings of variational methods in machine learning back to at least the early works by Peterson and Anderson (1987) and Hinton and van Camp (1993). This together with insight from Parisi (1988) led to a flurry of work in the area (Jordan et al., 1999; Waterhouse et al., 1996).

We will in this section describe the fundamentals of variational methods for approximating the posterior distribution. Variational inference, together with connections to Monte Carlo methods, are studied in Papers G and H.

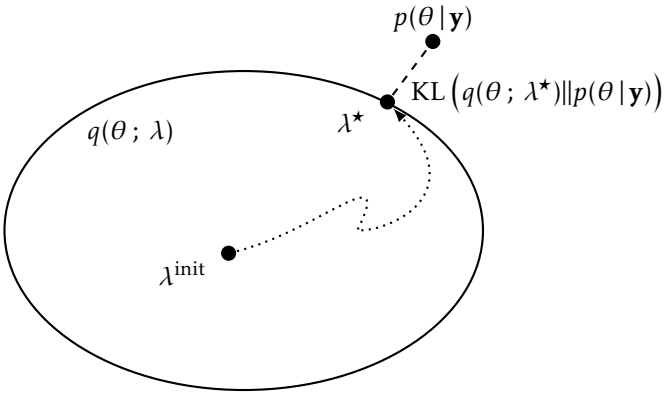


Figure 3.2: *Conceptual illustration of variational methods. Each point corresponds to a distribution on θ. The ellipse contains the approximating family $q(\theta\,;\,\lambda)$ indexed by variational parameters λ.*

3.2.1 The variational idea

Just like in MC methods, with variational methods we are interested in approximating the posterior distribution. We do this by postulating a variational family of approximations, $q(\theta\,;\,\lambda)$ indexed by the variational parameters λ. A common example is to use a normal distribution, $q(\theta\,;\,\lambda) = \mathcal{N}(\theta \mid \mu, \sigma^2)$ where $\lambda = (\mu, \sigma)$. The key idea is then to turn to mathematical optimization, choosing λ such that we minimize a cost function representing the discrepancy between the variational approximation and the posterior distribution. A common choice of cost function is the Kullback-Leibler (KL) divergence from the variational approximation to the posterior,

$$\mathrm{KL}\left(q(\theta\,;\,\lambda)\|p(\theta \mid \mathbf{y})\right) = \mathbb{E}_{q(\theta\,;\,\lambda)}\left[\log q(\theta\,;\,\lambda) - \log p(\theta \mid \mathbf{y})\right]. \tag{3.12}$$

For an illustration of generic variational methods for approximate inference see Figure 3.2. However, this expression still requires us to evaluate the posterior distribution, the problem we are trying to solve in the first place.

To resolve the issue we note that minimizing the KL divergence is equivalent to maximizing the *evidence lower bound* (ELBO) (Jordan et al., 1999),

$$\mathrm{L}(\lambda) := \mathbb{E}_{q(\theta\,;\,\lambda)}\left[\log p(\mathbf{y}, \theta) - \log q(\theta\,;\,\lambda)\right]. \tag{3.13}$$

This is true because we can rewrite the KL divergence as follows

$$\mathrm{KL}\left(q(\theta\,;\,\lambda) \,|\, p(\theta \mid \mathbf{y})\right) = \log p(\mathbf{y}) - \mathbb{E}_{q(\theta\,;\,\lambda)}\left[\log p(\mathbf{y}, \theta) - \log q(\theta\,;\,\lambda)\right]. \tag{3.14}$$

Because $\log p(\mathbf{y})$ does not depend on the variational parameters λ, we can minimize the KL by minimizing the second expression (negative ELBO) on the right hand side.

The rest of this section will be focused on various ways for maximizing the ELBO, focusing on an explicit coordinate ascent algorithm for *mean field* approximations and *stochastic gradient* methods for generic variational approximating families.

3.2.2 Coordinate ascent variational inference

Coordinate ascent variational inference (CAVI) is a method for finding an optimal variational approximation when we restrict our family to be independent over the components of θ, i.e.

$$q(\theta) = \prod_{k=1}^{K} q_k(\theta_k), \tag{3.15}$$

where we assume $\theta = (\theta_1, \ldots, \theta_K)^\top$. Note that we have not made any parametric assumptions on the factors $q_k(\cdot)$, except that they are probability distributions. This variational family of approximations is known as the *mean field variational family*.

Under the mean field assumption it is possible to design a coordinate ascent method to optimize the ELBO in Equation (3.13). We update each factor $q_k(\cdot)$ one by one, keeping the other $K-1$ factors fixed. A necessary condition in optimization for a point to be optimal is that the derivative is equal to zero. However, in our setting each factor $q_k(\cdot)$ is a *functional* and we instead rely on calculus of variations and functional derivatives (Forsyth, 1960). To ensure that $q_k(\cdot)$ is a probability density function we can study the Lagrangian (Bertsekas, 2016; Boyd and Vandenberghe, 2004) $\widetilde{\mathrm{L}}$ for Equation (3.13),

$$\begin{aligned}\widetilde{\mathrm{L}}(q_{1:K}, \nu_{1:K}) &= \mathbb{E}_{\prod_k q_k(\theta_k)}\left[\log p(\mathbf{y}, \theta_{1:K}) - \sum_{k=1}^{K} \log q_k(\theta_k)\right] - \sum_{k=1}^{K} \nu_k \left(\int q_k(\theta_k)\,\mathrm{d}\theta_k - 1\right) \\ &= \mathbb{E}_{q_k(\theta_k)}\left[\mathbb{E}_{\prod_{m\neq k} q_m(\theta_m)}\left[\log p(\mathbf{y}, \theta_{1:K})\right] - \log q_k(\theta_k) - \nu_k\right] + \text{const.},\end{aligned} \tag{3.16}$$

where $\theta_{1:K} = (\theta_1, \ldots, \theta_K)^\top = \theta$, the ν_k's are the Lagrange multipliers, and const. includes all terms constant with respect to $q_k(\theta_k)$. Using the Euler-Lagrange equation we can compute the functional derivative with respect to $q_k(\theta_k)$

$$\frac{\partial \widetilde{\mathrm{L}}(q_{1:K}, \nu_{1:k})}{\partial q_k(\theta_k)} = \mathbb{E}_{\prod_{m\neq k} q_m(\theta_m)}\left[\log p(\mathbf{y}, \theta_{1:K})\right] - \log q_k(\theta_k) + \text{const.} \tag{3.17}$$

Setting the right-hand side equal to zero gives us the solution for factor $q_k^\star(\theta_k)$,

$$q_k^\star(\theta_k) \propto \exp\left(\mathbb{E}_{\prod_{m\neq k} q_m^\star(\theta_m)}\left[\log p(\mathbf{y}, \theta_{1:K})\right]\right). \tag{3.18}$$

This requires us to be able to evaluate the expectation, and re-normalize the right-hand side of Equation (3.18) with respect to θ_k. This is possible when

the complete conditional $p(\theta_k \mid \mathbf{y}, \theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_K)$ is in a specific class of exponential family distributions (Blei et al., 2017). For more information on CAVI, and a detailed *Bayesian mixture* example, see e.g. Bishop (2006); Blei et al. (2017).

CAVI is not applicable if we can not evaluate the expectation and the normalization constant in Equation (3.18). This might be true if our model $p(\mathbf{y}, \theta)$ is too complex. The mean field approximation also makes a strong assumption on the independence between components of the parameter θ. When either of these two assumptions does not hold, we need to resort to a different approach to optimize the ELBO. In the next section, we discuss applying stochastic gradient methods for optimization.

3.2.3 Stochastic gradient variational inference

We return to the case when we have a variational family of approximations parameterized by λ, i.e. $q(\theta\,;\,\lambda)$. When either or both of the model $p(\mathbf{y}, \theta)$ and the variational approximation $q(\theta\,;\,\lambda)$ do not satisfy the requirements for CAVI, we must look to other methods for optimizing the ELBO. A recent approach that has garnered a considerable amount of success is to make use of stochastic gradients to optimize the ELBO (Kingma and Welling, 2014; Mnih and Gregor, 2014; Paisley et al., 2012; Ranganath et al., 2014; Salimans and Knowles, 2013). Essentially we use standard gradient descent with decreasing step-size, but instead of exact gradients we use gradients approximated via MC methods. Stochastic gradient descent is an iterative method for λ. The update is

$$\lambda^n = \lambda^{n-1} + \alpha_n \widehat{\mathrm{g}}(\lambda^{n-1}), \tag{3.19}$$

where $\widehat{\mathrm{g}}(\lambda^{n-1})$ is a MC estimator of the ELBO gradient

$$\mathrm{g}(\lambda^{n-1}) := \nabla_\lambda \mathrm{L}(\lambda^{n-1}) \tag{3.20}$$

and the stepsizes satisfy $\alpha_n \geq 0$, $\sum_n \alpha_n = \infty$, $\sum_n \alpha_n^2 < \infty$ (Robbins and Monro, 1951).

We review below two of the most common estimators of the ELBO gradient: the score function estimator and the reparameterization estimator.

Score function estimator The score function gradient estimator is based on a reformulation of the ELBO gradient (Mnih and Gregor, 2014; Paisley et al., 2012; Ranganath et al., 2014) as an expectation with respect to the variational approximation $q(\theta\,;\,\lambda)$. The estimator is also known by its other names: the log-derivative trick or REINFORCE (Glynn, 1990; Williams, 1992). It is a generic estimator with very few assumptions on the parameter space or the variational family. It works

whether θ is continuous or discrete. We have

$$\begin{aligned}\nabla_\lambda \mathrm{L}(\lambda) &= \nabla_\lambda \mathbb{E}_{q(\theta\,;\,\lambda)}\left[\log p(\mathbf{y},\theta) - \log q(\theta\,;\,\lambda)\right] \\ &= \mathbb{E}_{q(\theta\,;\,\lambda)}\left[(\log p(\mathbf{y},\theta) - \log q(\theta\,;\,\lambda)) \cdot \nabla_\lambda \log q(\theta\,;\,\lambda)\right], \end{aligned} \tag{3.21}$$

where we have made use of the fact that the expectation of the *score function* $\nabla_\lambda \log q(\theta\,;\,\lambda)$ is zero, i.e.

$$\mathbb{E}_{q(\theta\,;\,\lambda)}\left[\nabla_\lambda \log q(\theta\,;\,\lambda)\right] = 0, \tag{3.22}$$

and the log-derivative trick $\nabla_\lambda q(\theta\,;\,\lambda) = q(\theta\,;\,\lambda)\nabla_\lambda \log q(\theta\,;\,\lambda)$.

Equation (3.21) suggests estimating the ELBO gradient using the standard MC idea,

$$\nabla_\lambda \mathrm{L}(\lambda) \approx \widehat{\mathrm{g}}_{\mathrm{score}} = \frac{1}{N}\sum_{i=1}^{N}\Big(\log p(\mathbf{y},\theta^i) - \log q(\theta^i\,;\,\lambda)\Big) \cdot \nabla_\lambda \log q(\theta^i\,;\,\lambda), \tag{3.23}$$

where the samples are iid $\theta^i \sim q(\cdot\,;\,\lambda)$, just like we discussed in Section 3.1. From the theoretical results of standard MC we know that this estimator is unbiased and consistent.

The main limitation of the score function estimator is that it tends to give quite high variance gradient estimators. This is generally undesirable for an efficient optimization method. It can be partially alleviated by various variance reducing adjustments, such as control variates (Robert and Casella, 2004) and Rao-Blackwellization (Casella and Robert, 1996) as explained by e.g. Ranganath et al. (2014).

Reparameterization estimator The reparameterization trick (Bonnet, 1964; Kingma and Welling, 2014; Price, 1958; Salimans and Knowles, 2013) usually results in a gradient estimator with lower variance than the score function estimator. This comes at the price of applicability, where the reparameterization trick only works for a certain class of continuous distributions. We require that the model $p(\mathbf{y},\theta)$ is differentiable with respect to the parameters θ. Furthermore, we require that the variational approximation $q(\theta\,;\,\lambda)$ is differentiable with respect to θ, and that it is *reparameterizable* by a differentiable function f. What this means is that we assume that we can simulate from the variational approximation through a *non-centered parameterization* (Papaspiliopoulos et al., 2003), i.e.

$$\theta \sim q(\theta\,;\,\lambda) \Longleftrightarrow \theta = \mathrm{f}(\varepsilon,\lambda), \quad \varepsilon \sim p(\varepsilon), \tag{3.24}$$

where f is differentiable with respect to λ, and the distribution $p(\varepsilon)$ does not depend on the variational parameters λ. A commonly used variational family is the normal distribution $q(\theta\,;\,\lambda) = \mathcal{N}(\theta \,|\, \mu, \sigma^2)$, with $\lambda = (\mu,\sigma)$, which has the following non-centered parameterization

$$\mathrm{f}(\varepsilon,\lambda) = \mu + \sigma\varepsilon, \quad \varepsilon \sim \mathcal{N}(0,1).$$

With the non-centered parameterization we can rewrite the gradient of the ELBO using the reparameterization trick

$$\begin{aligned}\nabla_\lambda \mathrm{L}(\lambda) &= \nabla_\lambda \mathbb{E}_{q(\theta\,;\,\lambda)}\left[\log p(\mathbf{y},\theta) - \log q(\theta\,;\,\lambda)\right] \\ &= \nabla_\lambda \mathbb{E}_{p(\varepsilon)}\left[\log p(\mathbf{y}, \mathrm{f}(\varepsilon,\lambda)) - \log q(\mathrm{f}(\varepsilon,\lambda)\,;\,\lambda)\right] \\ &= \mathbb{E}_{p(\varepsilon)}\left[\nabla_\theta \left(\log p(\mathbf{y},\theta) - \log q(\theta\,;\,\lambda)\right)\Big|_{\theta=\mathrm{f}(\varepsilon,\lambda)} \nabla_\lambda \mathrm{f}(\varepsilon,\lambda)\right] \end{aligned} \tag{3.25}$$

where in the last equation we have again made use of that the expectation of the score function is zero (Roeder et al., 2017). Equation (3.25) suggests the following MC estimator of the ELBO gradient

$$\nabla_\lambda \mathrm{L}(\lambda) \approx \widehat{\mathrm{g}}_{\mathrm{reparam}} = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \left(\log p(\mathbf{y},\theta) - \log q(\theta\,;\,\lambda)\right)\Big|_{\theta=\mathrm{f}(\varepsilon^i,\lambda)} \nabla_\lambda \mathrm{f}(\varepsilon^i,\lambda), \tag{3.26}$$

where the samples are iid $\varepsilon^i \sim p(\varepsilon)$. In practice, the variance of the reparameterization estimator is often low enough that a single sample, $N = 1$, is sufficient (Kingma and Welling, 2014; Roeder et al., 2017).

The restriction that the variational approximation must be reparameterizable can be alleviated using *partial reparameterization*, see e.g. Ruiz et al. (2016) and Paper G.

Large-scale data Datasets where the number of (exchangeable) datapoints T is very large works well in the setting of stochastic gradient-based variational inference. We simply replace evaluating the exact log-likelihood $\log p(\mathbf{y}\,|\,\theta)$ with an unbiased estimate of it, i.e.

$$\log p(\mathbf{y}\,|\,\theta) = \sum_{t=1}^{T} \log p(y_t\,|\,\theta) \approx \frac{T}{B}\sum_{b=1}^{B} \log p(y_{\tau_b}\,|\,\theta), \quad \tau_b \sim \mathrm{U}(\{1,\ldots,T\}). \tag{3.27}$$

Here τ_b are discrete random variables drawn uniformly over the support $\{1,\ldots,T\}$. When used together with either of the two stochastic gradient methods above, the resulting algorithm is known as a *doubly stochastic* algorithm (Titsias and Lázaro-Gredilla, 2014).

3.2.4 Variational expectation-maximization

The expectation–maximization (EM) algorithm (Dempster et al., 1977) is a method for finding the maximum likelihood estimate $\widehat{\theta}^{\mathrm{ML}}$ when the model depends on local latent (unobserved) variables $\mathbf{x}$. This means that our probabilistic model is $p(\mathbf{y},\mathbf{x}\,|\,\theta)$, and we are interested in maximizing the marginal likelihood $p(\mathbf{y}\,|\,\theta) =$

$\int p(\mathbf{y}, \mathbf{x} \mid \theta) \, \mathrm{d}\mathbf{x}$ with respect to the parameters θ. The EM algorithm solves this through coordinate ascent for the ELBO defined by

$$\mathrm{L}(q(\mathbf{x}), \theta) = \mathbb{E}_{q(\mathbf{x})} \left[\log p(\mathbf{y}, \mathbf{x} \mid \theta) - \log q(\mathbf{x}) \right], \tag{3.28}$$

where $q(\mathbf{x})$ is a distribution over $\mathbf{x}$. The EM algorithm consists of iteratively solving

$$q^n(\mathbf{x}) = \arg\max_{q(\mathbf{x})} \mathrm{L}(q(\mathbf{x}), \theta^{n-1}), \tag{3.29a}$$

$$\theta^n = \arg\max_{\theta} \mathrm{L}(q^n(\mathbf{x}), \theta), \tag{3.29b}$$

where the solution to the first line is given by $q^n(\mathbf{x}) = p(\mathbf{x} \mid \mathbf{y}, \theta^{n-1})$.

One of the first approaches to variational expectation–maximization (VEM) (Beal and Ghahramani, 2003)[1] replaces Equation (3.29a) (maximization for $q(\mathbf{x})$), by CAVI. While the exact EM (under suitable conditions) is guaranteed to converge to a local maxima of $\log p(\mathbf{y} \mid \theta)$, this is generally not true for VEM. Just like variational inference and MC methods are approximate methods for inference, so is VEM.

Because VEM relies on CAVI, it has restricted applicability. Recent research has focused on using stochastic gradient-based variational inference instead. In this setting we iteratively do only partial updates for q and θ, unlike in Equation (3.29). We let the variational approximation be defined by $q(\mathbf{x}\,;\,\lambda)$, and design an update scheme for λ, θ based on stochastic gradient descent for the ELBO

$$\mathrm{L}(\lambda, \theta) = \mathbb{E}_{q(\mathbf{x}\,;\,\lambda)} \left[\log p(\mathbf{y}, \mathbf{x} \mid \theta) - \log q(\mathbf{x}\,;\,\lambda) \right]. \tag{3.30}$$

Based on step-size sequences $\alpha_n^\lambda, \alpha_n^\theta$ (satisfying the same requirements as detailed in Section 3.2.3), we get

$$\lambda^n = \lambda^{n-1} + \alpha_n^\lambda \hat{\mathrm{g}}_n^\lambda, \tag{3.31a}$$

$$\theta^n = \theta^{n-1} + \alpha_n^\theta \hat{\mathrm{g}}_n^\theta, \tag{3.31b}$$

where $\hat{\mathrm{g}}_n^\theta$ is

$$\hat{\mathrm{g}}_n^\theta = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log p(\mathbf{y}, \mathbf{x}^i \mid \theta) \Big|_{\theta=\theta^{n-1}}, \quad \mathbf{x}^i \sim q(\mathbf{x}\,;\,\lambda^{n-1}). \tag{3.32}$$

For the gradient with respect to λ, $\hat{\mathrm{g}}_n^\lambda$, we make use of either the score function or reparameterization trick estimators of Section 3.2.3, evaluated for λ^{n-1}, θ^{n-1}. Because the algorithm we described is a form of stochastic gradient descent-algorithm, it will (under suitable conditions) converge to a local maxima of the *evidence lower bound* in Equation (3.30) (Bottou et al., 2018). However, there are no guarantees for the original problem of maximum marginal likelihood maximization. We are maximizing a *lower bound* to the marginal likelihood, where the gap between them that we neglect is a function of both λ and θ.

[1] The authors actually consider model selection: $p(\mathbf{y}, \mathbf{x}, \theta \mid m)$ and the mean-field model $q(\theta)q(\mathbf{x})$ where m is a model selection variable. The fundamental idea is the same.

4

Concluding remarks

We conclude Part I of this thesis by a short summary, and discuss potential avenues for future work in approximate inference. However, we note that more discussion is also provided at the end of most articles in Part II.

In this thesis we introduce various new methods for approximate inference and learning in latent variable models. The focus has been on sequential Monte Carlo and variational inference, as well as the connection between them. At the heart of each paper lies MC methods, used to estimate intractable integrals. We present a tutorial on, new applications for, and extensions to sequential Monte Carlo. We also study reparameterization gradients for variational inference (VI), and derive the variational sequential Monte Carlo (VSMC) family of approximations to the posterior.

We would like to take the opportunity to focus the discussion on *nested Monte Carlo* and *variational Monte Carlo*, two open avenues for future research.

Nested Monte Carlo We refer to the use of MC methods within other MC methods as nested Monte Carlo (NMC) algorithms. Examples include, but are not limited to,

- Markov chain Monte Carlo within SMC, e.g. sequential Monte Carlo samplers (Del Moral et al., 2006),
- SMC within Markov chain Monte Carlo, e.g. particle Markov chain Monte Carlo (Andrieu et al., 2010),
- SMC within SMC, e.g. nested sequential Monte Carlo (Papers D, E) or SMC^2 (Chopin et al., 2012).

This area is by no means exhausted, particularly the intersection between different NMC algorithms. We believe that nested sequential Monte Carlo (NSMC) is potentially useful for a much wider range of applications than it has been applied to so far. More generally, how can we design NMC algorithms such that we preserve the theoretical guarantees? Can we automatically choose the right algorithm for a particular task? Is it possible to give practical guarantees on the approximation we obtain?

Variational Monte Carlo By synthesizing approximate inference from both variational and MC methods, we get variational Monte Carlo (VMC). We can also think of this as a form of *adaptive* MC method. Examples include, but are not limited to,

- Markov chain Monte Carlo-inspired approximations, e.g. Hamiltonian variational inference (Salimans et al., 2015),
- importance sampling within variational inference, e.g. variational importance sampling (Paper H), (Cremer et al., 2017),
- SMC within variational inference, e.g. variational sequential Monte Carlo (Paper H).

Explicitly making use of the MC approximation of the posterior when defining a variational approximation is a largely unexplored area. Can we efficiently learn the corresponding variational parameters? Will investigating different cost functions help improve the approximation? Is it possible to extract informative theoretical guarantees for this class of approximations? How can we better take into account the discrete nature, introduced by the use of MC, of the approximations?

The above two examples are areas that can potentially have a significant impact on approximate inference for probabilistic machine learning.

Bibliography

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

Thomas Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society*, pages 330–418, 1763.

Matthew J. Beal and Zoubin Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian statistics*, 2003.

Dimitri P. Bertsekas, editor. *Nonlinear Programming*. Athena Scientific, 2016.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, 2006.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2017.

G. Bonnet. Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire. *Annals of Telecommunications*, 19(9):203–220, 1964.

L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer Science+Business Media, 2005.

G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

George Casella and Roger L Berger. *Statistical inference*. Duxbury, 2002.

N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. Smc2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2012.

C. Cremer, Q. Morris, and D. Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv:1704.02916*, April 2017.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 68(3):411–436, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

Roger Eckhardt. Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science*, 15(131-136):30, 1987.

Bradley Efron and Trevor Hastie, editors. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge University Press, 2016.

Ronald A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 222(594-604):309–368, 1922.

A. S. Forsyth. *Calculus of variations*. Dover Publications Inc., 1960.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer Science+Business Media, 2009.

Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 2015.

P. W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, oct 1990.

Anders Hald. On the history of maximum likelihood in relation to inverse probability and least squares. *Statistical Science*, 14(2):214–222, 1999.

Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13, New York, NY, USA, 1993. ACM.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Pierre-Simon Laplace. Mémoire sur la probabilité des causes par led évènemens. *Mémoires de Mathématque et Physique, Presentés à l'Académie Royale des Sciences, par divers Savans & lûs dans ses Assemblées, Tome Sixiéme*, 66:621–56, 1774.

Fredrik Lindsten, Adam M. Johansen, Christian A. Naesseth, Brent Kirkpatrick, Thomas B. Schön, John Aston, and Alexandre Bouchard-Côté. Divide-and-conquer with sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science+Business Media, 2004.

Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, 2014.

Christian Naesseth, Fredrik Lindsten, and Thomas Schön. Nested sequential Monte Carlo methods. In *International Conference on Machine Learning (ICML)*, pages 1292–1301, 2015a.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *IEEE Information Theory Workshop (ITW)*, pages 431–435, 2014a.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1862–1870, 2014b.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Towards automated sequential Monte Carlo for probabilistic graphical models. In *NIPS Workshop on Black Box Learning and Inference*, 2015b.

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 W. Linderman, Rajesh Ranganath, and David M. Blei. Variational sequential Monte Carlo. In *Artificial Intelligence and Statistics (AISTATS)*, pages 968–977, 2018a.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Elements of sequential Monte Carlo. *Foundations and Trends in Machine Learning*, 2018b. (proposal accepted, manuscript in preparation).

Jorge Nocedal and S. Wright, editors. *Numerical Optimization*. Springer Science+Business Media, 2006.

J. W. Paisley, D. M. Blei, and M. I. Jordan. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*, 2012.

Sina Khoshfetrat Pakazad, Christian A. Naesseth, Fredrik Lindsten, and Anders Hansson. Distributed, scalable and gossip-free consensus optimization with application to data analysis. *arXiv:1705.02469*, 2017.

Omiros Papaspiliopoulos, Gareth O. Roberts, and Martin Sköld. Non-centered parameterisations for hierarchical models and data augmentation. In *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, page 307. Oxford University Press, USA, 2003.

Giorgio Parisi. *Statistical field theory*. Addison-Wesley, 1988.

Carsten Peterson and James R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1(5), 1987.

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

Tom Rainforth, Christian A. Naesseth, Fredrik Lindsten, Brooks Paige, Jan-Willem Vandemeent, Arnaud Doucet, and Frank Wood. Interacting particle Markov chain Monte Carlo. In *International Conference on Machine Learning (ICML)*, pages 2616–2625, 2016.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 09 1951.

Christian P. Robert. *The Bayesian Choice*. Springer Science+Business Media, 2007.

Christian P. Robert and George Casella. *Monte Carlo statistical methods*. Springer Science+Business Media, 2004.

Geoffrey Roeder, Yuhuai Wu, and David K Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, pages 6925–6934, 2017.

Francisco J. R. Ruiz, Michalis K. Titsias, and David M. Blei. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems*, 2016.

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

Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, 2015.

Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai. Sequential Monte Carlo methods for system identification. *IFAC-PapersOnLine (SYSID)*, 48(28):775–786, 2015.

Stephen M Stigler. *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press, 1986.

M. K. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, 2014.

David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

John von Neumann. Various Techniques Used in Connection with Random Digits. *Journal of Research of the National Bureau of Standards*, 12:36–38, 1951.

Steve Waterhouse, David Mackay, and Tony Robinson. Bayesian methods for mixtures of experts. In *Advances in Neural Information Processing Systems*, pages 351–357. MIT Press, 1996.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Part II

Publications

Paper A

Elements of Sequential Monte Carlo

Authors: Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön

Edited version of the paper:

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Elements of sequential Monte Carlo. *Foundations and Trends in Machine Learning*, 2018b. (proposal accepted, manuscript in preparation).

This paper has been formatted to fit this layout.

Elements of Sequential Monte Carlo

Christian A. Naesseth*, Fredrik Lindsten†, and Thomas B. Schön†

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

†Dept. of Information Technology
Uppsala University
Uppsala, Sweden
{fredrik.lindsten,thomas.schon}@it.uu.se

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, 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 and review its main building block, importance sampling. Then, we discuss practical issues and theoretical results. Finally, we examine the two main user design choiches: the *proposal distribution* and the *target distribution*.

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 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) (Le et al., 2018; Maddison et al., 2017;

Naesseth et al., 2018), inference evaluation (Cusumano-Towner and Mansinghka, 2017; Grosse et al., 2015), 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; Kitagawa, 1993; Stewart and McCarty, 1992). In this introduction we provide relevant background material, introduce a running example, and discuss the use of code snippets throughout the tutorial.

This manuscript is an extract from a tutorial paper in preparation. The full paper will further include a section on pseudo-marginal methods (Andrieu et al., 2009, 2010) and a section on conditional SMC methods (Andrieu et al., 2010).

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 (Hastings, 1970; Metropolis et al., 1953), 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; Kitagawa, 1993; Stewart and McCarty, 1992). 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). However, 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 γ_t is 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 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 there is not necessarily a direct one-to-one correspondence between the latent variables $x_{1:T}$ of the target distribution and the latent variables $\mathbf{x}$ of the probabilistic model.

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 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 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 x_1. This class of models is especially common for data that has an inherent time 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 Equation (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 $\gamma_T(x_{1:T}) = p(\mathbf{x} \mid \mathbf{y})$ 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 Equation (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 latents $x_{1:t-1}$ to generate x_t, y_t

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

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

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

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

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

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

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

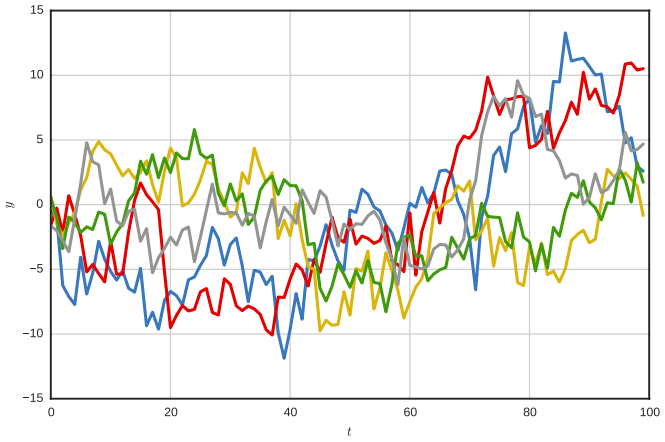


Figure 1: *Five sample paths of $y_{1:T}$ from our running example for $T = 100$.*

For both these sequences of target distributions the final iteration T is the posterior distribution, i.e. $\gamma_T(x_{1:T}) = p(x_{1:T} \mid y_{1:T}) = p(\mathbf{x} \mid \mathbf{y})$. 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 Equation (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.

Example 1: Non-Markovian Gaussian Sequence Model

As running example for illustration purposes we use a non-Markovian Gaussian sequence model. It is

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

with observed variables y_t (data), and where

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

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

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

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

where $p(\mathbf{y} \,|\, \mathbf{x})$ 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 model index k coincides with the target index t. We define the target distribution

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

where the distributions $s_k(x_k \,|\, x_{k+1})$ 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. $\gamma_T(x_T) = p(\mathbf{x} \,|\, \mathbf{y})$. In fact, at each step we have that the target distribution is a partial posterior $\gamma_t(x_t) = p(\mathbf{x} \,|\, y_{1:t})$.

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

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

where $0 = \tau_1 < \ldots < \tau_T = 1$, and again make use of the user chosen backward kernels $s_k(x_k \,|\, x_{k+1})$. 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 $\gamma_T(x_T) = p(\mathbf{x} \,|\, \mathbf{y})$.

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 γ_T coincides with the distribution of interest, we can leverage SMC for inference.

1.3 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:

```
1 import numpy as np
2 import numpy.random as npr
3 from scipy.misc import logsumexp
4 from scipy.stats import norm
```

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

1.4 Outline

The remainder of this tutorial is organized as follows. In Section 2, we first introduce importance sampling (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.

The tutorial concludes with a discussion and outlook in Section 4.

2 Importance Sampling to Sequential Monte Carlo

Typical applications require us to be able to evaluate or sample from the target distributions γ_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 expectations of test functions h_t with respect to the target distribution γ_t,

$$\gamma_t(h_t) := \mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right]. \tag{14}$$

Examples include posterior predictive distributions, Bayesian p-values, and point estimates such as the posterior mean. Computing Equation (14) is intractable, but by a clever trick we can rewrite it as follows

$$\mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right] = \frac{1}{Z_t}\mathbb{E}_{q_t}\left[\frac{\widetilde{\gamma}_t(x_{1:t})}{q_t(x_{1:t})}h_t(x_{1:t})\right] = \frac{\mathbb{E}_{q_t}\left[\frac{\widetilde{\gamma}_t(x_{1:t})}{q_t(x_{1:t})}h_t(x_{1:t})\right]}{\mathbb{E}_{q_t}\left[\frac{\widetilde{\gamma}_t(x_{1:t})}{q_t(x_{1:t})}\right]}. \tag{15}$$

The PDF q_t is a user choosen proposal distribution, we assume it is simple to sample from and evaluate. We can now estimate the right hand side of Equation (15) using the Monte Carlo method,

$$\mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right] \approx \frac{\frac{1}{N}\sum_{i=1}^{N}\widetilde{w}_t(x_{1:t}^i)h_t(x_{1:t}^i)}{\frac{1}{N}\sum_{j=1}^{N}\widetilde{w}_t(x_{1:t}^j)}, \tag{16}$$

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 Equation (16) more compactly as

$$\mathbb{E}_{\gamma_t}\left[h_t(x_{1:t})\right] \approx \sum_{i=1}^{N} w_t^i h_t(x_{1:t}^i), \quad x_{1:t}^i \overset{\text{iid}}{\sim} q_t, \tag{17}$$

where the normalized weights w_t^i are defined by

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

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 γ_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 $i = 1, \ldots, N$

with $\widetilde{w}_t^i$ a shorthand for $\widetilde{w}_t(x_{1:t}^i)$. The estimate in Equation (17) is strongly consistent, converging (almost surely) to the true expectation as the number of samples tend to infinity. An alternate view of IS is to consider it an (empirical) approximation of γ_t,

$$\gamma_t(x_{1:t}) \approx \sum_{i=1}^N w_t^i \delta_{x_{1:t}^i}(x_{1:t}) =: \widehat{\gamma}_t(x_{1:t}), \tag{18}$$

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

$$Z_t \approx \frac{1}{N} \sum_{i=1}^N \widetilde{w}_t^i =: \widehat{Z}_t \tag{19}$$

Because the weights depend on the random samples, $x_{1:t}^i$, it is itself a random variable. One of its key properties is that it is *unbiased*, which 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.

A 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, 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.

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 γ_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

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

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

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 Equation (19). 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 Equation (19), is still unbiased and consistent.

In Example 2 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 2: 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

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

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

$$x_t = \phi x_{t-1} + \sqrt{q}\varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, 1). \tag{20}$$

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

$$\widetilde{w}_t(x_{1:t}) = \widetilde{w}_{t-1}(x_{1:t-1}) \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1})} \tag{21}$$

$$= \widetilde{w}_{t-1}(x_{1:t-1})\, \mathcal{N}\left(y_t \mid \sum_{k=1}^{t} \beta^{t-k} x_k, r\right), \tag{22}$$

where $\widetilde{w}_0 = 1$. We provide Example Code A.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 A.2: *Sequential importance sampling for Example 1.*

For improved numerical stability we update the log-weights $\log \widetilde{w}_t$.

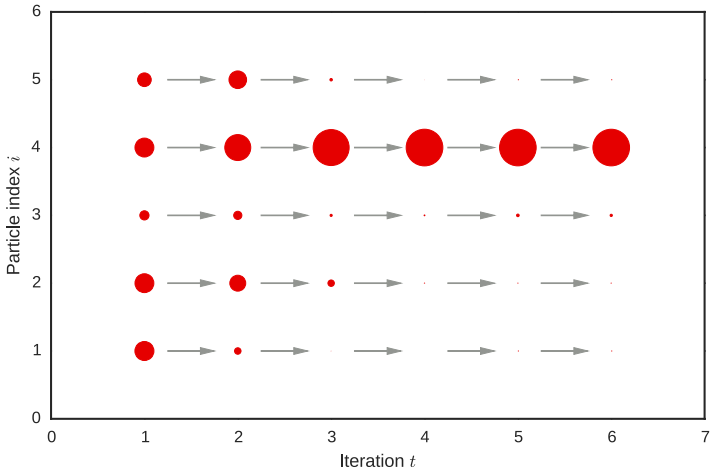


Figure 2: *Weight degeneracy of the* SIS *method. Size of the disks represent the size of the corresponding weights* w_t^i.

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; 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 3 using the running example.

Example 3: SIS weight degeracy

We return to our running example, set the length $T = 6$, number of particles $N = 5$, and $(\phi, q, \beta, r) = (0.9, 1.0, 0.5, 1.0)$. Figure 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 concen-

tration 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 essentially solve 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 the proposal distribution. For certain sequence models the weight degeneracy issue can be resolved altogether, 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 issue 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 γ_1 using standard IS. Simulating N times independently from the first proposal

$$x_1^i \overset{\text{iid}}{\sim} q_1(x_1), \quad i = 1, \ldots, N, \tag{23}$$

and assigning corresponding weights

$$\widetilde{w}_1^i = \frac{\widetilde{\gamma}_1(x_1^i)}{q_1(x_1^i)}, \quad w_1^i = \frac{\widetilde{w}_1^i}{\sum_{j=1}^N \widetilde{w}_1^j}, \quad i = 1, \ldots, N, \tag{24}$$

lets us approximate γ_1 (cf. Equation (18)) by

$$\widehat{\gamma}_1(x_1) = \sum_{i=1}^N w_1^i \delta_{x_1^i}(x_1). \tag{25}$$

The key innovation of the SMC algorithm is that it takes advantage of the information provided in $\widehat{\gamma}_1(x_1)$, Equation (25), when constructing a proposal for the next target distribution γ_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

$$x_{1:2}^i \overset{\text{iid}}{\sim} \widehat{\gamma}_1(x_1)q_2(x_2 \mid x_1), \quad i = 1, \ldots, N, \tag{26}$$

and assign weights

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

Simulating $x_{1:2}^i$, Equation (26), 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 A.3. In Sections 2.2.1 and 2.2.2 we revisit resampling and present alternative resampling algorithms, increasing efficiency by correlation and adaptation.

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

Example Code A.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 $\widehat{\gamma}_1(x_1)q_2(x_2 \mid x_1)$. The approximation of γ_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

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 γ_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 Equation (27))
    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 Equation (28))
  end
  Set $w_t^i = \frac{\widetilde{w}_t^i}{\sum_j \widetilde{w}_t^j}$, for $i = 1, \ldots, N$
end

a conditional distribution

$$q_t(x_{1:t}) = \widehat{\gamma}_{t-1}(x_{1:t-1}) q_t(x_t \mid x_{1:t-1}). \tag{27}$$

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

$$\widetilde{w}_t^i = \frac{\widetilde{\gamma}_t(x_{1:t}^i)}{\widetilde{\gamma}_{t-1}(x_{1:t-1}^i) q_t(x_t^i \mid x_{1:t-1}^i)}, \quad w_t^i = \frac{\widetilde{w}_t^i}{\sum_{j=1}^N \widetilde{w}_t^j}, \quad i = 1, \ldots, N, \tag{28}$$

and approximate γ_t by

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

The normalization constant Z_t can be estimated by

$$\widehat{Z}_t = \prod_{k=1}^t \frac{1}{N} \sum_{i=1}^N \widetilde{w}_k^i. \tag{29}$$

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 4 we return to our running example, using the same settings as in Example 3, to study the sample diversity and quality of the basic SMC method compared to SIS.

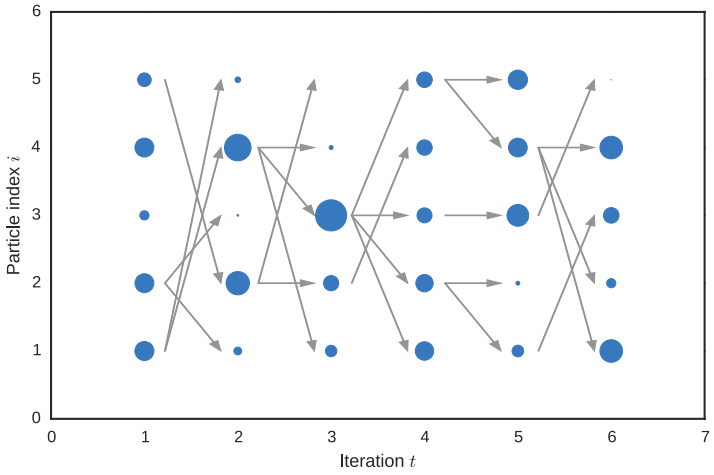


Figure 3: *Diversity of samples in the SMC method. Size of the disks represent the size of the weights w_t^i, and the grey arrows represent resampling.*

Example 4: SMC sample diversity

We illustrate the weights and resampling dependencies in Figure 3. The grey arrows represent what sample from iteration $t-1$ that generated the current sample at iteration t. 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.

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

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 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, they do however suffer from what is known as *path degeneracy*. We illustrate this property in Example 5.

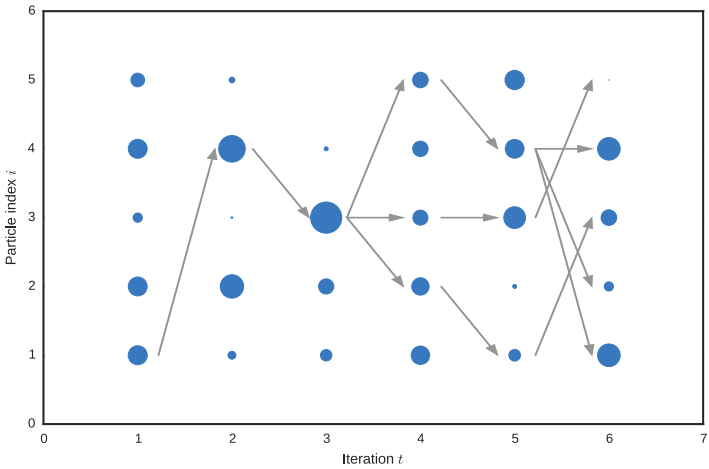


Figure 4: *Path degeneracy of the SMC method for smoothing approximation. Size of the disks represent the size of the weights w_t^i, and the grey arrows represent resampling.*

Example 5: SMC path degeneracy

In Figure 4 we reiterate the result from our previous example, Example 4. 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 sample, e.g. we have $N = 5$ identical copies of x_1^1 in Figure 4. 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 Figure 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 β correspond to more long-range dependencies in our non-Markovian LVM. We can see that for modest values of β 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

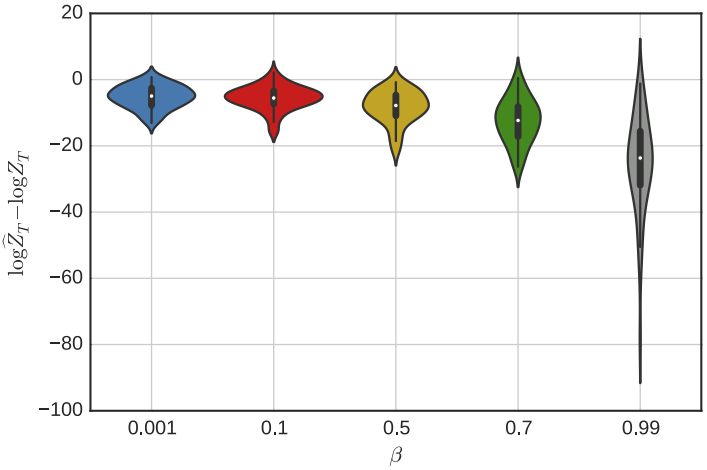


Figure 5: *Violinplots for the log of the* SMC *estimate of the normalization constant divided by the true value, i.e.* $\log \widehat{Z}_T - \log Z_T$. *The number of particles are* $N = 20$, *length of the data is* $T = 100$, *and we study five different settings of* $\beta = (0.001, 0.1, 0.5, 0.7,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

resampling, which essentially means that we do not resample 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 improvement.

2.2.1 Low-variance Resampling

The resampling step introduces extra variance in the short-term, using $\widehat{\gamma}_{t-1}$ to estimate γ_{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_{t-1}^i. We have that

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

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

$$\sum_{i=1}^{a-1} w_{t-1}^i \leq u < \sum_{i=1}^{a} w_{t-1}^i, \tag{30}$$

for $u \sim \mathrm{U}(0,1)$. Repeating the above process independently N times gives multinomial resampling. That is, we draw $u^i \sim \mathrm{U}(0,1)$ and find the corresponding a^i for each $i = 1, \ldots, N$.

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 $u^i \sim \mathrm{U}(i-1/N, i/N)$ for each strata $i = 1, \ldots, N$. Finally, the corresponding ancestor variables a^i are given by studying Equation (30).

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

```
1 def stratified_resampling(w, x):
2     Np = w.shape[0]
3     u = (np.arange(Np) + npr.rand(Np))/Np
4     bins = np.cumsum(w)
5     return x[np.digitize(u,bins)]
```

Example Code A.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 $u \sim \mathrm{U}(0,1)$ 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 $u \sim \mathrm{U}(0,1)$ is identical for all $i = 1, \ldots, N$. Then, we find corresponding ancestors a^i by again studying Equation (30). 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 the most computationally efficient way for resampling.

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

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

$$\mathrm{ESS}_t = \frac{1}{\sum_{i=1}^{N} \left(w_t^i\right)^2}. \tag{31}$$

The ESS is a positive variable taking values in the continuous range between 1 and N. For IS the ESS is an approximation to the number of exact samples from γ_t 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 Equations (27) and (28). If ESS_{t-1} is above the prespecified threshold we simply omit the resampling step in Equation (27) and obtain

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

This means that for iterations where we do not resample, we simply revert to using SIS. The corresponding weight update is

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

Note then dependence on the previous weight $\widetilde{w}_{t-1}^i$ just like in standard SIS.

When the ESS falls below our pre-set threshold, we use the standard update Equations (27) and (28) with resampling instead.

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

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 Appendix 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 in the violin plot discussed in Example 5.

We will delve deeper into the applications of the unbiasedness property and its consequences in Section 3.

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 γ_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).

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

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

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

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 from the SMC approximation compared to that of 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 γ_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 Kullback-Leibler (KL) divergence from the expected SMC approximation $\mathbb{E}\left[\widehat{\gamma}_t\right]$ to the target distribution γ_T.

Theorem 4 (Sample Bound).

$$KL\left(\mathbb{E}\left[\widehat{\gamma}_T\right] \| \gamma_T\right) \leq \frac{\mathcal{C}}{N},$$

for a finite constant $\mathcal{C}$.

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 (Huggins and Roy, 2015; Naesseth et al., 2018; Wood et al., 2014).

3 Learning Proposals and Twisting Targets

The two main design choices of sequential Monte Carlo methods are the proposal distributions q_t and the intermediate target distributions $\widetilde{\gamma}_t$. Carefully choosing these can drastically improve the efficiency of the algorithm and accuracy of our estimation. Adapting both the proposal and target distribution is especially important if the latent space dimension is high.

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 $t-1$, i.e. $\widehat{\gamma}_{t-1} \overset{d}{=} \gamma_{t-1}$. Then, choose the proposal q_t that minimizes the KL divergence from the joint distribution $\gamma_{t-1}(x_{1:t-1})q_t(x_t \,|\, x_{1:t-1})$ 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* $\mathrm{KL}(\gamma_{t-1}(x_{1:t-1})q_t(x_t \mid x_{1:t-1}) \| \gamma_t(x_{1:t}))$ *is given by*

$$q_t^\star(x_t \mid x_{1:t-1}) = \gamma_t(x_t \mid x_{1:t-1}) = \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_t(x_{1:t-1})}, \tag{32}$$

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

Example 6: Optimal proposal for Example 1

If we let $\widetilde{\gamma}_t(x_{1:t}) = \widetilde{\gamma}_{t-1}(x_{1:t-1}) f(x_t \mid x_{t-1}) g(x_t \mid x_{1:t})$, 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 ∇l_t and $\nabla^2 l_t$ are the gradient and Hessian of the log-PDF with respect to x_t, respectively. A natural approximation to the optimal proposal is then

$$q_t(x_t \mid x_{1:t-1}) = \mathcal{N}\left(x_t \mid \bar{x} - \left(\nabla^2 l_t(\bar{x})\right)^{-1} \nabla l_t(\bar{x}), -\nabla^2 l_t(\bar{x})^{-1}\right). \tag{33}$$

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 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,

$$x_t = a(x_{1:t-1}, v_t), \tag{34a}$$
$$y_t = c(x_{1:t}, e_t), \tag{34b}$$

where v_t, 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 $q_t^\star(x_t \mid x_{1:t-1}) = p(x_t \mid x_{1:t-1}, y_t)$.

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

$$p(x_t, y_t \mid x_{1:t-1}) \approx \widehat{p}(x_t, y_t \mid x_{1:t-1}) = \mathcal{N}\left(\begin{pmatrix} x_t \\ y_t \end{pmatrix} \middle| \, \widehat{\mu}, \widehat{\Sigma}\right), \tag{35}$$

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

$$\widehat{\mu} = \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}, \quad \widehat{\Sigma} = \begin{pmatrix} \widehat{\Sigma}_{xx} & \widehat{\Sigma}_{xy} \\ \widehat{\Sigma}_{yx} & \widehat{\Sigma}_{yy} \end{pmatrix}. \tag{36}$$

Under the assumptions of the approximation in Equation (35), the distribution of $x_t \mid x_{1:t-1}, y_t$ is tractable:

$$\widehat{p}(x_t \mid x_{1:t-1}, y_t) = \mathcal{N}(x_t \mid \mu_t, \Sigma_t), \tag{37}$$

with

$$\mu_t = \widehat{\mu}_x + \widehat{\Sigma}_{xy}\widehat{\Sigma}_{yy}^{-1}\left(y_t - \widehat{\mu}_y\right), \tag{38a}$$

$$\Sigma_t = \widehat{\Sigma}_{xx} - \widehat{\Sigma}_{xy}\widehat{\Sigma}_{yy}^{-1}\widehat{\Sigma}_{yx}. \tag{38b}$$

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 B, 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 Equation (79) 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 sigma 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 Equation (81) in the appendix.

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 7 we study a simple example when x and y are one-dimensional random variables.

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 Equation (79) (EKF) and in Equation (81) (UKF), respectively.

Example 7: Gaussian Proposal Approximations

We illustrate the analytic Gaussian approximations from Algorithm 4 based on a simple scalar example

$$x = v, \quad v \sim \mathcal{N}(0, 1), \tag{39a}$$
$$y = \frac{(x+1)(x-1)(x-3)}{6} + e, \quad e \sim \mathcal{N}(0, 0.5). \tag{39b}$$

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 Figure 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 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

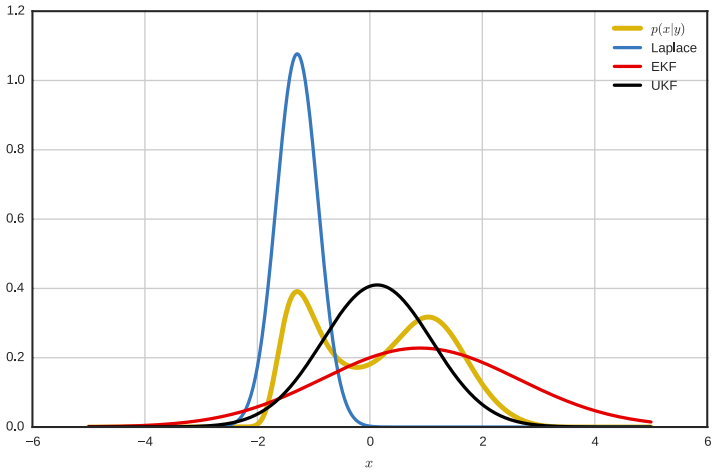


Figure 6: *Analytic Gaussian approximations of $p(x \mid y)$ for the model in Equation* (39).

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 actually improve accuracy compared to SMC methods with standard proposals (Naesseth et al., 2015), even for equal compute.

We are interested in approximating the locally optimal proposal distribution, i.e. $q_t^\star(x_t \mid x_{1:t-1}) \propto \widetilde{\gamma}_t(x_{1:t})$, 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 $q_t^\star(x_t \mid x_{1:t-1})$. We construct an approximation of the optimal proposal independently for each particle $x_{1:t-1}^i$ as follows

$$\widehat{q}_t^\star(x_t \mid x_{1:t-1}^i) = \sum_{j=1}^{M} \frac{\widetilde{v}_t^{j,i}}{\sum_l \widetilde{v}_t^{l,i}} \delta_{\bar{x}^{j,i}}(x_t), \quad \bar{x}^{j,i} \sim r_t(x_t \mid x_{1:t-1}^i), \tag{40}$$

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

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

We replace $q_t(x_t \mid x_{1:t-1})$ in Equation (27) with samples from the distribution defined in Equation (40). The corresponding weight update, under the assumption

Algorithm 5: Nested IS Approximation

input : Unnormalized target distributions $\widetilde{\gamma}_t$, $\widetilde{\gamma}_{t-1}$, resampled particle $x_{1:t-1}^i$, nested proposal r_t, number of samples M.
output: Samples and weights $x_t^i, \widetilde{w}_t^i$.
Sample $\bar{x}^{j,i} \sim r_t(x_t \mid x_{1:t-1}^i),\ j = 1, \ldots, M$.
Set $\widetilde{v}_t^{j,i} = \frac{\widetilde{\gamma}_t((x_{1:t-1}^i, \bar{x}^{j,i}))}{\widetilde{\gamma}_{t-1}(x_{1:t-1}^i) r_t(\bar{x}^{j,i} \mid x_{1:t-1}^i)}$.
Sample $x_t^i \sim \sum_{j=1}^M \frac{\widetilde{v}_t^{j,i}}{\sum_l \widetilde{v}_t^{l,i}} \delta_{\bar{x}^{j,i}}(x_t)$. (see Equation (40))
Set $\widetilde{w}_t^i = \frac{1}{M} \sum_{j=1}^M \widetilde{v}_t^{j,i}$. (see Equation (41))

that we have resampled $x_{1:t-1}^i$, is given by

$$\widetilde{w}_t^i = \frac{1}{M} \sum_{j=1}^M \widetilde{v}_t^{j,i}. \tag{41}$$

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. However, the memory requirement is lower 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.

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 λ, we denote this as $q_t(x_t \mid x_{1:t-1} ; \lambda)$. A common choice is to use a conditional Gaussian distribution

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

where $\mu_\lambda(\cdot), \sigma^2_\lambda(\cdot)$ are neural networks parameterized by λ. 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 (Krishnan et al., 2017; Louizos et al., 2017).

Below we discuss two classes of methods to learn the parameters λ, 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 γ_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 is 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 (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 γ_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

$$\begin{aligned}\mathrm{KL}(p(x_{1:T} \mid y_{1:T}) \| q_T(x_{1:T} ; \lambda)) &= \int p(x_{1:T} \mid y_{1:T}) \log \frac{p(x_{1:T} \mid y_{1:T})}{q_T(x_{1:T} ; \lambda)} \, \mathrm{d}x_{1:T} \\ &= -\int p(x_{1:T} \mid y_{1:T}) \log q_T(x_{1:T} ; \lambda) \, \mathrm{d}x_{1:T} + \text{const}, \end{aligned} \tag{42}$$

where "const" includes all terms constant with respect to the proposal distribution. If we could compute the gradients of Equation (42) with respect to λ we

could employ a standard gradient descent method to optimize it. Unfortunately the gradients, given by

$$\mathrm{g}_{\mathrm{KL}} = \nabla_\lambda \mathrm{KL}(p(x_{1:T} \mid y_{1:T}) \| q_T(x_{1:T}\,;\,\lambda)) = \\ -\mathbb{E}_{p(x_{1:T}\mid y_{1:T})}\left[\sum_{t=1}^{T} \nabla_\lambda \log q_t(x_t \mid x_{1:t-1}\,;\,\lambda)\right], \tag{43}$$

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 λ and then the SMC procedure itself to approximate the gradient in Equation (43). Given this estimate of the gradient we can do an update step based on standard stochastic gradient descent methods. Suppose we have the iterate λ^{n-1}, we estimate the gradient in Equation (43) using Algorithm 3 with proposals $q_t(x_t \mid x_{1:t-1}\,;\,\lambda^{n-1})$, and get

$$\mathrm{g}_{\mathrm{KL}}^n \approx \widehat{\mathrm{g}_{\mathrm{KL}}}^n := \sum_{i=1}^{N} w_T^i \nabla_\lambda \log q_T(x_{1:T}^i\,;\,\lambda)\Big|_{\lambda=\lambda^{n-1}}. \tag{44}$$

With stochastic gradient descent we update our iterate by

$$\lambda^n = \lambda^{n-1} - \alpha_n \widehat{\mathrm{g}_{\mathrm{KL}}}^n, \tag{45}$$

where α_n are a set of positive step-sizes, typically choosen 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 $p(y_{1:T})$. Instead of the KL divergence in Equation (42), we consider minimizing

$$\mathrm{KL}\left(p(x_{1:T}, y_{1:T}) \| p(y_{1:T}) q_T(x_{1:T}\,;\,\lambda(y_{1:T}))\right) \\ = -\mathbb{E}_{p(x_{1:T},y_{1:T})}\left[\log q_T(x_{1:T}\,;\,\lambda(y_{1:T}))\right] + \text{const.} \tag{46}$$

If we let $\lambda_\eta(\cdot)$ be a parametric function with parameters η, we can compute stochastic gradients of Equation (46) w.r.t. η with no need to resort to SMC. A common choice is to let λ_η be defined by a neural network where η are the weights

and biases. We define the gradient

$$\mathrm{g}_{\mathrm{KL}} = -\mathbb{E}_{p(x_{1:T}, y_{1:T})}\left[\nabla_\eta \log q_T(x_{1:T}\,;\,\lambda_\eta(y_{1:T}))\right], \tag{47}$$

and estimate it, for the current iterate η^n, using MC

$$\begin{aligned}&\mathrm{g}_{\mathrm{KL}}^n \approx \widehat{\mathrm{g}_{\mathrm{KL}}}^n := \\ &-\nabla_\eta \log q_T(\bar{x}_{1:T}\,;\,\lambda_\eta(\bar{y}_{1:T}))\Big|_{\eta=\eta^{n-1}}, \quad (\bar{x}_{1:T}, \bar{y}_{1:T}) \sim p(x_{1:T}, y_{1:T}).\end{aligned} \tag{48}$$

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 Equation (45) when updating η^n, replacing the gradient with the expression from Equation (48)

$$\eta^n = \eta^{n-1} - \alpha_n \widehat{\mathrm{g}_{\mathrm{KL}}}^n. \tag{49}$$

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 Equation (49) will ensure convergence to a local minima of its cost function Equation (46) 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.

Variational Sequential Monte Carlo The key idea in variational sequential Monte Carlo (VSMC) (Le et al., 2018; Maddison et al., 2017; Naesseth et al., 2018) 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 γ_T. The dependence on λ comes in 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.

Algorithm 6: Adaptive SMC

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

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

where $\widehat{\mathrm{g}_{\mathrm{KL}}}^n$ is given in Equation (44).

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

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

where $\widehat{\mathrm{g}_{\mathrm{KL}}}^n$ is given in Equation (48).

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

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 γ_T can be bounded from above

$$\mathrm{KL}\left(\mathbb{E}\left[\widehat{\gamma}_T(x_{1:T}\,;\,\lambda)\right] \| \gamma_T(x_{1:T})\right) \leq -\mathbb{E}\left[\log \frac{\widehat{Z}_T}{Z_T}\right], \tag{50}$$

where the log-normalization constant estimate, cf. Equation (29), is

$$\log \widehat{Z}_T = \sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_t\left(x_{1:t}^i\,;\,\lambda\right)\right). \tag{51}$$

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 λ, minimizing Equation (50) is equivalent to maximizing

$$\mathbb{E}\left[\log \widehat{Z}_T\right] = \mathbb{E}\left[\sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_t\left(x_{1:t}^i\,;\,\lambda\right)\right)\right]. \tag{52}$$

The KL divergence $\mathrm{KL}(\mathbb{E}[\widehat{\gamma}_T] \| \gamma_T)$ is non-negative, which means Equation (52) is a lower bound to the log-normalization constant $\log Z_T$. This is why Equation (52) is typically known as an *evidence lower bound* in the variational inference literature. By maximizing the cost function in Equation (52), with respect to λ, 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, we need to resort to approximations.

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

$$x_t = h_t(x_{1:t-1}, \varepsilon_t\,;\,\lambda), \quad \varepsilon_t \sim p(\varepsilon), \tag{53}$$

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 $\lambda^n = \lambda^{n-1} + \alpha_n \widehat{\mathrm{g}_{\mathrm{VSMC}}}^n$, where $\widehat{\mathrm{g}_{\mathrm{VSMC}}}^n$ is given in Equation (55).
(The stepsizes satisfy $\alpha_n > 0$, $\sum_n \alpha_n = \infty$, and $\sum_n \alpha_n^2 < \infty$.)

where the distribution of the random variable ε_t is independent of the parameters λ. If we further assume that h_t is differentiable we can use

$$\nabla_\lambda \mathbb{E}\left[\log \widehat{Z}_T\right] \approx \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} w_t^i \nabla_\lambda \log \widetilde{w}_t\left(x_{1:t}^i\,;\,\lambda\right)\right] =: \mathrm{g}_{\mathrm{VSMC}}, \tag{54}$$

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 Equation (53). The approximation follows from ignoring the gradient part that results from the resampling step, which has been shown to work well in practice (Le et al., 2018; Maddison et al., 2017; Naesseth et al., 2018).

Just like for the adaptive SMC methods, Equation (54) suggests a stochastic gradient method to optimize λ. Given an iterate λ^{n-1}, we compute the gradient estimate by running SMC with proposals $q_t(x_t \mid x_{1:t-1}\,;\,\lambda^{n-1})$ and evaluate

$$\mathrm{g}_{\mathrm{VSMC}} \approx \widehat{\mathrm{g}_{\mathrm{VSMC}}}^n = \sum_{t=1}^{T}\sum_{i=1}^{N} w_t^i \nabla_\lambda \log \widetilde{w}_t\left(x_{1:t}^i\,;\,\lambda\right)\Big|_{\lambda=\lambda^{n-1}}. \tag{55}$$

With stochastic gradient ascent we update our iterate by

$$\lambda^n = \lambda^{n-1} + \alpha_n \widehat{\mathrm{g}_{\mathrm{VSMC}}}^n, \tag{56}$$

where α_n are positive step-sizes, choosen 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 γ_T, we are free to choose the intermediate target distributions γ_t to maximize the accuracy of our estimate $\widehat{\gamma}_T$. By making use of information from future iterations, 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 Equation (32), we still would not be getting exact samples from our target distribution. The reason for this is because 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. Equation (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 $x_{1:t-1}^i$) 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.

$$\gamma_t^\star(x_{1:t}) = \gamma_T(x_{1:t}). \tag{57}$$

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

$$q_t^\star(x_t \mid x_{1:t-1}) = \gamma_t^\star(x_t \mid x_{1:t-1}) = \gamma_T(x_t \mid x_{1:t-1}) = p(x_t \mid x_{1:t-1}, y_{1:T}),$$

where in the final equality we have assumed that the target γ_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 $\widetilde{\gamma}_T(x_{1:T}) = p(x_{1:T}, y_{1:T})$ can be split into a product of factors

$$p(x_{1:T}, y_{1:T}) = \prod_{t=1}^{T} \widetilde{f}_t(x_{1:t})\widetilde{g}_t(x_{1:t}, y_{1:T}). \tag{58}$$

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

$$\widetilde{\gamma}_t(x_{1:t}) = \widetilde{\gamma}_{t-1}(x_{1:t-1})\widetilde{f}_t(x_{1:t})\widetilde{g}_t(x_{1:t}, y_{1:T})\frac{\psi_t(x_{1:t})}{\psi_{t-1}(x_{1:t-1})}, \tag{59}$$

where $\psi_t \geq 0$ are our so-called *twisting potentials*, with $\psi_0 \equiv \psi_T \equiv 1$. Then we can confirm that the unnormalized target at the final iteration is the full joint PDF

$$\widetilde{\gamma}_T(x_{1:T}) = p(x_{1:T}, y_{1:T}) \prod_{t=1}^{T} \frac{\psi_t(x_{1:t})}{\psi_{t-1}(x_{1:t-1})} = p(x_{1:T}, y_{1:T}),$$

and our normalized target distribution $\gamma_T(x_{1:T})$ is the posterior $p(x_{1:T} \mid y_{1:T})$. The target distribution structure postulated in Equation (59) 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 targets must fulfill

$$\widetilde{\gamma}_t^\star(x_{1:t}) = \int \widetilde{\gamma}_{t+1}^\star(x_{1:t+1})\,\mathrm{d}x_{t+1}, \quad t = 1, \ldots, T-1. \tag{60}$$

This follows from Equation (57), each target $\gamma_t^\star$ should be the marginal distribution of $x_{1:t}$ under the final target distribution γ_T. By replacing $\widetilde{\gamma}_t^\star$ in Equation (60) with the definition from Equation (59) we get

$$\psi_t^\star(x_{1:t}) = \int \widetilde{f}_{t+1}(x_{1:t+1})\widetilde{g}_{t+1}(x_{1:t+1}, y_{1:T})\psi_{t+1}^\star(x_{1:t+1})\,\mathrm{d}x_{t+1}. \tag{61}$$

While Equation (61) 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 with the following structure

$$\widetilde{f}_t(x_{1:t}) = f_t(x_t \mid x_{1:t-1}), \tag{62a}$$

$$\widetilde{g}_t(x_{1:t}, y_{1:T}) = g_t(y_t \mid x_{1:t}, y_{1:t-1}). \tag{62b}$$

For this model we can rewrite Equation (61) as follows

$$\begin{aligned}\psi_t^\star(x_{1:t}) &= \mathbb{E}_{f_{t+1}(x_{t+1} \mid x_{1:t})}\left[g_{t+1}(y_{t+1} \mid x_{1:t+1}, y_{1:t})\psi_{t+1}^\star(x_{1:t+1})\right] \\ &= \mathbb{E}_{\prod_{k=t+1}^{T} f_k(x_k \mid x_{1:k-1})}\left[\prod_{l=t+1}^{T} g_l(y_l \mid x_{1:l}, y_{1:l-1})\right] = p(y_{t+1:T} \mid x_{1:t}),\end{aligned} \tag{63}$$

where the final expression is the predictive likelihood of $y_{t+1:T}$ given the latent variables $x_{1:t}$. In Example 8 we derive the optimal twisting potentials for our Gaussian non-Markovian LVM.

Example 8: 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. Equation (62)) 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

$$y_{t\,|\,l} = \left(\sum_{k=l+1}^{t} \beta^{t-k}\phi^{k-l}\right) x_l + \sum_{k=1}^{l} \beta^{t-k} x_k + e_t + \sum_{k=l+1}^{t} \gamma_k v_k, \tag{64}$$

where γ_k is

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

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 Equation (64) 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 Equation (59) we can express this class of models as follows

$$\widetilde{f}_t(x_{1:t}) = s_t(x_{t-1}\,|\,x_t), \quad t = 2, \ldots, T, \tag{65a}$$

$$\widetilde{g}_t(x_{1:t}, y_{1:T}) = \frac{p(x_t)g_t(x_t, y_{1:T})}{p(x_{t-1})g_{t-1}(x_{t-1}, y_{1:T})}, \quad t = 2, \ldots, T, \tag{65b}$$

where $p(x_t)$ is the prior distribution on the latent variable. We initialize by $\widetilde{f}_1 \equiv 1$ and $\widetilde{g}_1 = p(x_1)g_1(x_1, y_{1:T})$. The potential g_t usually takes either of the two following forms

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

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

where $0 = \tau_1 < \tau_2 < \ldots < \tau_T = 1$. The distribution 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 $\widetilde{f}_t$ and $\widetilde{g}_t$ in Equation (61) with their respective definitions from Equation (65) we get

$$\begin{aligned}\psi_t^\star(x_{1:t}) &= \int \frac{p(x_{t+1})g_{t+1}(x_{t+1}, y_{1:T})}{p(x_t)g_t(x_t, y_{1:T})} s_{t+1}(x_t \mid x_{t+1}) \psi_{t+1}^\star(x_{1:t+1}) \,\mathrm{d}x_{t+1} \\ &= \frac{1}{p(x_t)g_t(x_t, y_{1:T})} \int p(x_T) g_T(x_T, y_{1:T}) \prod_{k=t+1}^{T} s_k(x_{k-1} \mid x_k) \,\mathrm{d}x_{t+1:T}, \end{aligned} \tag{66}$$

where we can see that because of the (reverse) Markov structure in the annealing model from Equation (65), 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 Equation (62) 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 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 (Johansen and Doucet, 2008; Pitt and Shephard, 1999). 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, Equation (61), for one iteration we get a locally optimal twisting potential

$$\psi_t(x_{1:t}) = \int \widetilde{f}_{t+1}(x_{1:t+1}) \widetilde{g}_{t+1}(x_{1:t+1}, y_{1:T}) \,\mathrm{d}x_{t+1}. \tag{67}$$

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.

$$q_t(x_t \mid x_{1:t-1}) = \frac{\widetilde{f}_t(x_{1:t}) \widetilde{g}_t(x_{1:t}, y_{1:T})}{\psi_{t-1}(x_{1:t-1})}.$$

With these choices of twisting and proposal, based on the twisted target structure Equation (59), we obtain the fully adapted SMC (Johansen and Doucet, 2008; Pitt and Shephard, 1999) with corresponding weights

$$\widetilde{w}_t(x_{1:t}) = \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_t(x_{1:t-1})q_t(x_t \mid x_{1:t-1})} = \psi_t(x_{1:t}). \tag{68}$$

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

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

including information from iteration $t+1$ already at iteration t. In some situations we might be interested in the filtering target $\prod_{k=1}^{t} \widetilde{f}_k(x_{1:k})\widetilde{g}_k(x_{1:k}, y_{1:T})$. If so, we can make use of the approach described above to generate the particles $x^i_{1:t}$. Then to estimate the filtering target we simply average the particles $x^i_{1:t}$, ignoring the weights in Equation (68) because of the discrepancy between the filtering and predictive targets.

The proposal used above is not the locally optimal proposal distribution for our (predictive) target distribution. The optimal proposal for the target based on Equation (67) is

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

which would result in weights

$$\widetilde{w}_t(x_{1:t}) = \frac{\widetilde{\gamma}_t(x_{1:t})}{\widetilde{\gamma}_t(x_{1:t-1})q_t(x_t \mid x_{1:t-1})} = \frac{\int \widetilde{f}_t(x_{1:t})\widetilde{g}_t(x_{1:t}, y_{1:T})\psi_t(x_{1:t}) \,\mathrm{d}x_t}{\psi_{t-1}(x_{1:t-1})}. \tag{69}$$

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 Equation (67) are available, the integration in Equation (69) is not necessarily tractable.

Except for a few special cases the optimal 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 Equation (61) 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 $\widetilde{f}_t = f_t(x_t \mid x_{1:t-1})$ and $\widetilde{g}_t = g_t(y_t \mid x_{1:t})$, cf. Equation (62). 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.

$$\psi_t(x_{1:t}\,;\,\rho_{t+1}) := \mathbb{E}_{f_t(x_{t+1}\mid x_{1:t})}\Big[\bar{\psi}_{t+1}(x_{1:t+1}\,;\,\rho_{t+1})\Big], \quad t = 1,\dots,T-1, \tag{70}$$

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 Equation (61) with twisting potentials defined by Equation (70), we get the recursive approximation criteria for the $\bar{\psi}_t$'s

$$\bar{\psi}_T(x_{1:T}\,;\,\rho_T) \approx g_T(y_T \mid x_{1:T}), \tag{71a}$$

$$\bar{\psi}_t(x_{1:t}\,;\,\rho_t) \approx g_t(y_t \mid x_{1:t})\mathbb{E}_{f_t(x_{t+1}\mid x_{1:t})}\Big[\bar{\psi}_{t+1}(x_{1:t+1}\,;\,\rho_{t+1})\Big]. \tag{71b}$$

Because we assumed that $\mathbb{E}_{f_t(x_t \mid x_{1:t-1})}\Big[\bar{\psi}_t(x_{1:t}\,;\,\rho_t)\Big]$ is available analytically, and Equation (71) 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

$$q_t(x_t \mid x_{1:t-1}\,;\,\rho_t) = \frac{f_t(x_t \mid x_{1:t-1})\bar{\psi}_t(x_{1:t}\,;\,\rho_t)}{\mathbb{E}_{f_t(x_t \mid x_{1:t-1})}\Big[\bar{\psi}_t(x_{1:t}\,;\,\rho_t)\Big]}. \tag{72}$$

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

$$\widetilde{w}_t(x_{1:t}\,;\,\rho_{t:t+1}) = \frac{g_t(y_t \mid x_{1:t})\psi_t(x_{1:t}\,;\,\rho_{t+1})}{\bar{\psi}_t(x_{1:t}\,;\,\rho_t)}. \tag{73}$$

What remains is how to use Equation (71) 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 ρ_t^n by recursively minimizing

$$\rho_T^n = \arg\min_{\rho_T} \sum_{i=1}^{N} w_T^i \left(\bar{\psi}_T(x_{1:T}^i\,;\,\rho_T) - g_T(y_T \mid x_{1:T}^i)\right)^2, \tag{74a}$$

$$\rho_t^n = \arg\min_{\rho_t} \sum_{i=1}^{N} w_t^i \left(\bar{\psi}_t(x_{1:t}^i\,;\,\rho_t) - g_t(y_t \mid x_{1:t}^i)\psi_t(x_{1:t}^i\,;\,\rho_{t+1}^n)\right)^2, \tag{74b}$$

where the last step is for $t = T-1,\dots,1$. With the updated set of parameters, we re-run SMC and repeat the updates in Equation (74) 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 recurrent neural network (RNN) in Example 9.

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 Equation (72) and Equation (73), respectively.
  Update the parameters to $\rho_1^n, \ldots, \rho_T^n$ by solving Equation (74).
end

Example 9: Stochastic Recurrent Neural Network

A stochastic RNN is a non-Markovian LVM where the parameters of the transition and observation models are defined by RNNs. A common example is using the conditional Gaussian distribution to define the transition PDF

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

where the functions $\mu_t(\cdot), \Sigma_t(\cdot)$ are defined by RNNs. 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 ρ_t—for notational brevity we have not made this dependence explicit. The proposal (cf. Equation (72)) is given by

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

and the twisting potential

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

Twisting the target distribution is an area of active research (Guarniero et al., 2017; Heng et al., 2017). 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 ψ_t and proposals q_t might lead to even more accurate algorithms.

4 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 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. Finally, 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.

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 drastically improve accuracy. However, jointly learning twisting potentials and proposal distributions for complex models is a challenging and largely unsolved problem. Parallelization and distributed computing for SMC algorithms are also topics of active research.

We hope that this paper can serve as a catalyst for further research on SMC methods and their application in machine learning.

Appendix

A Proof of Unbiasedness

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

$$\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right) = \prod_{i=1}^{N} q_1(x_1^i) \cdot \prod_{t=2}^{T} \prod_{i=1}^{N} \left[w_{t-1}^{a_{t-1}^i} q_t\left(x_t^i \mid x_{1:t-1}^{a_{t-1}^i}\right)\right], \tag{75}$$

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 $a_{1:T-1}^{1:N} = \bigcup_{t=1}^{T-1} \left\{a_t^i\right\}_{i=1}^{N}$. We are going to prove that

$$\mathbb{E}_{\mathrm{Q}_{\mathrm{SMC}}\left(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\right)}\left[\frac{\widehat{Z}_T}{Z_T}\right] = 1, \tag{76}$$

from which the result follows. To do this 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 $x_{1:T}^i = (x_1^{b_1^i}, \ldots, x_T^{b_T^i})$. Using this notation we can rewrite the integrand in Equation (76) as follows,

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

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

Now, using Equation (76) and the above rewrite of the integrand we obtain

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

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 $x_{1:T}^1$. The two final equalities follow because we are averaging N^{T-1} values that are all equal to 1. This concludes the proof.

B Taylor and Unscented Transforms

In this appendix we detail the EKF- and UKF-based approximations to the distribution $p(x_t, y_t \,|\, x_{1:t-1})$. 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

$$x_t \approx \bar{a} + J_v^a(\bar{a})v_t, \tag{77a}$$
$$y_t \approx \bar{c} + J_{x_t}^c(\bar{c})(x_t - \bar{a}) + J_{e_t}^c(\bar{c})e_t, \tag{77b}$$

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 Equation (77):

$$x_t \approx \bar{a} + J_v^a(\bar{a})v_t, \tag{78a}$$
$$y_t \approx \bar{c} + J_{x_t}^c(\bar{c})J_v^a(\bar{a})v_t + J_{e_t}^c(\bar{c})e_t. \tag{78b}$$

For the approximation in Equation (78) $x_t, y_t \mid x_{1:t-1}$ 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 $\widehat{\Sigma}$:

$$\widehat{\mu}_x = \bar{a}, \tag{79a}$$
$$\widehat{\mu}_y = \bar{c}, \tag{79b}$$
$$\widehat{\Sigma}_{xx} = J_v^a(\bar{a})\, Q\, J_v^a(\bar{a})^\top, \tag{79c}$$
$$\widehat{\Sigma}_{xy} = \widehat{\Sigma}_{yx}^\top = J_v^a(\bar{a})\, Q\, J_v^a(\bar{a})^\top J_{x_t}^c(\bar{c})^\top, \tag{79d}$$
$$\widehat{\Sigma}_{yy} = J_{x_t}^c(\bar{c}) J_v^a(\bar{a})\, Q\, J_v^a(\bar{a})^\top J_{x_t}^c(\bar{c})^\top + J_{e_t}^c(\bar{c})\, R\, J_{e_t}^c(\bar{c})^\top. \tag{79e}$$

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 Equation (34) 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,

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} a(x_{1:t-1}, v_t) \\ c\left((x_{1:t-1}, a(x_{1:t-1}, v_t)), e_t\right) \end{pmatrix} =: \tilde{a}(x_{1:t-1}, z_t), \tag{80}$$

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

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

where the scalars σ_l and vectors u_l correspond to the singular value decomposition of Σ_z. We then choose the $2n_x + 2n_y + 1$ sigma points for z as follows,

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

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 α and κ below. The mean and variance of $(x_t, y_t)^\top$

is estimated by

$$\mathbb{E}\left[(x_t, y_t)^\top\right] \approx \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix} = \sum_{l=-(n_x+n_y)}^{n_x+n_y} \omega^l \tilde{a}(x_{1:t-1}, \mathrm{z}^l), \tag{81a}$$

$$\begin{aligned} \mathrm{Var}\left((x_t, y_t)^\top\right) &\approx \begin{pmatrix} \widehat{\Sigma}^{xx} & \widehat{\Sigma}^{xy} \\ \widehat{\Sigma}^{yx} & \widehat{\Sigma}^{yy} \end{pmatrix} = \\ &(1 - \alpha^2 + \beta)\left(\tilde{a}(x_{1:t-1}, \mathrm{z}^0) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)\left(\tilde{a}(x_{1:t-1}, \mathrm{z}^0) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)^\top + \\ &\sum_{l=-(n_x+n_y)}^{n_x+n_y} \omega^l \left(\tilde{a}(x_{1:t-1}, \mathrm{z}^l) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)\left(\tilde{a}(x_{1:t-1}, \mathrm{z}^l) - \begin{pmatrix} \widehat{\mu}_x \\ \widehat{\mu}_y \end{pmatrix}\right)^\top, \end{aligned} \tag{81b}$$

where the coefficients ω^l are

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

and β is another design parameter.

Like in the EKF-based approximation we still need access to the densities corresponding to Equation (34). Unlike in the EKF case the, UKF-based approximation does not need access to derivatives. However, we need to choose the three design parameters: α, β, κ. Both α and κ control the spread of the sigma points, and β is related to the distribution of z_t (Julier, 2002). Typical values of these parameters are $(\alpha, \kappa, \beta) = (10^{-3}, 0, 2)$.

Bibliography

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.

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.

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.

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.

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.

Paul Fearnhead and Hans R Künsch. Particle filters and data assimilation. *Annual Review of Statistics and Its Application*, 5:421–449, 2018.

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.

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.

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.

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 ∞-ess, and an application to particle gibbs. *arXiv:1503.00966*, 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.

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.

Ming Lin, Rong Chen, Jun S Liu, et al. Lookahead strategies for sequential monte carlo. *Statistical Science*, 28(1):69–94, 2013.

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.

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.

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.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 09 1951.

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.

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.

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.

Frank Wood, Jan Willem Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pages 1024–1032, 2014.

Paper B

Capacity estimation of two-dimensional channels using Sequential Monte Carlo

Authors: Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön

Edited version of the paper:

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *IEEE Information Theory Workshop (ITW)*, pages 431–435, 2014a.

This paper has been formatted to fit this layout. The reference style has been changed from the Vancouver system to the Harvard system.

Capacity estimation of two-dimensional channels using Sequential Monte Carlo

Christian A. Naesseth*, Fredrik Lindsten†, and Thomas B. Schön†

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

†Dept. of Information Technology
Uppsala University
Uppsala, Sweden
{fredrik.lindsten,thomas.schon}@it.uu.se

Abstract

We derive a new Sequential-Monte-Carlo-based algorithm to estimate the capacity of two-dimensional channel models. The focus is on computing the noiseless capacity of the 2-D $(1, \infty)$ run-length limited constrained channel, but the underlying idea is generally applicable. The proposed algorithm is profiled against a state-of-the-art method, yielding more than an order of magnitude improvement in estimation accuracy for a given computation time.

1 Introduction

With ever increasing demands on storage system capacity and reliability there has been increasing interest in page-oriented storage solutions. For these types of systems variations of two-dimensional constraints can be imposed to help with, amongst other things, timing control and reduced intersymbol interference (Immink, 2004). This has sparked an interest in analyzing information theoretic properties of two-dimensional channel models for use in e.g. holographic data storage (Siegel, 2006).

Our main contribution is a new algorithm, based on sequential Monte Carlo (SMC) methods, for numerically estimating the capacity of two-dimensional channels. We show how we can utilize structure in the model to sample the auxiliary target distributions in the SMC algorithm exactly. The focus in this paper is on computing the noiseless capacity of constrained finite-size two-dimensional models. However, the proposed algorithm works also for various generalizations and noisy channel models.

Recently, several approaches have been proposed to solve the capacity estimation problem in two-dimensional constrained channels. These methods rely either on variational approximations (Sabato and Molkaraie, 2012) or on Markov chain Monte Carlo (Loeliger and Molkaraie, 2009; Molkaraie and Loeliger, 2013). Compared to these methods our algorithm is fundamentally different; samples are drawn sequentially from a sequence of probability distributions of increasing dimensions using SMC coupled with a finite state-space forward-backward procedure. We compare our proposed algorithm to a state-of-the-art Monte Carlo estimation algorithm proposed in (Loeliger and Molkaraie, 2009; Molkaraie and Loeliger, 2013). Using SMC algorithms has earlier been proposed to compute the information rate of one-dimensional continuous channel models with memory (Dauwels and Loeliger, 2008). Although both approaches are based on SMC, the methods, implementation and goals are very different.

2 Two-dimensional channel models

As in (Molkaraie and Loeliger, 2013) we consider the 2-D $(1,\infty)$ run-length limited constrained channel. The 2-D $(1,\infty)$ run-length limited constraint implies that no two horizontally or vertically adjacent bits on a 2-D lattice may be both be equal to 1. An example is given below:

$$\begin{array}{ccccc} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{array}$$

This channel can be modelled as a probabilistic graphical model (PGM). A PGM is a probabilistic model which *factorizes* according to the structure of an underlying graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. In this article we will focus on square lattice graphical models with pair-wise interactions, see Figure 1. That means that the joint probability mass function (PMF) of the set of random variables, $X := \{x_{1,1}, \ldots, x_{1,M}, x_{2,M}, \ldots, x_{M,M}\}$, can be represented as a product of factors over the pairs of variables in the graph:

$$p(X) = \frac{1}{Z} \prod_{(\ell j, mn) \in \mathcal{E}} \psi(x_{\ell,j}, x_{m,n}). \tag{1}$$

Here, Z—the partition function—is given by

$$Z = \sum_{X} \prod_{(\ell j, mn) \in \mathcal{E}} \psi(x_{\ell,j}, x_{m,n}), \tag{2}$$

and $\psi(x_{\ell,j}, x_{m,n})$ denotes the so-called potential function encoding the pairwise interaction between $x_{\ell,j}$ and $x_{m,n}$. For a more in-depth exposition of graphical models we refer the reader to (Koller and Friedman, 2009).

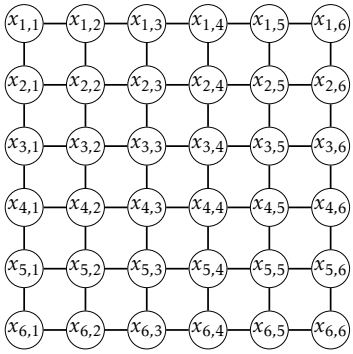


Figure 1: *$M \times M$ square lattice graphical model with pair-wise interactions. The nodes correspond to random variables $x_{\ell,j}$ and the edges encodes the interactions $\psi(x_{\ell,j}, x_{m,n})$.*

2.1 Constrained channels and PGM

The noiseless 2-D $(1, \infty)$ run-length limited constrained channel can be described by a square lattice graphical model as in Figure 1, with binary variables $x_{\ell,j} \in \{0, 1\}$ and pair-wise interactions between adjacent variables. Defining the factors as

$$\psi(x_{\ell,j}, x_{m,n}) = \begin{cases} 0, & \text{if } x_{\ell,j} = x_{m,n} = 1, \\ 1, & \text{otherwise,} \end{cases} \tag{3}$$

results in a joint PMF given by

$$p(X) = \frac{1}{Z} \prod_{(\ell j, mn) \in \mathcal{E}} \psi(x_{\ell,j}, x_{m,n}), \tag{4}$$

where the partition function Z is the number of satisfying configurations or, equivalently, the cardinality of the support of $p(X)$. For a channel of dimension $M \times M$ we can write the finite-size noiseless capacity as

$$C_M = \frac{1}{M^2} \log_2 Z. \tag{5}$$

Hence, to compute the capacity of the channel we need to compute the partition function Z. Unfortunately, calculating Z exactly is in general computationally intractable. This means that we need a way to approximate the partition function. Note that for this particular model, known upper and lower bounds of the infinite-size noiseless capacity, $M \to \infty$, agree on more than eight decimal digits (Kato and Zeger, 1999; Nagy and Zeger, 2000). However, our proposed method is applicable in the finite-size case, as well as to other models where no tight bounds are known.

2.2 High-dimensional undirected chains

In the previous section we described how we can calculate the noiseless capacity for 2-D channel models by casting the problem as a partition function estimation problem in the PGM framework. In our running example the corresponding graph is the $M \times M$ square lattice PGM depicted in Figure 1. We now show how we can turn these models into high-dimensional undirected chains by introducing a specific new set of variables. We will see that this idea, although simple, is a key enabler of our proposed algorithm.

We define $\mathbf{x}_k$ to be the M-dimensional variable corresponding to all the original variables in column k, i.e.

$$\mathbf{x}_k = \{x_{1,k}, \ldots, x_{M,k}\}, \qquad k = 1, \ldots, M. \tag{6}$$

The resulting graphical model in the $\mathbf{x}_k$'s will be an undirected chain with joint PMF given by

$$p(X) = \frac{1}{Z} \prod_{k=1}^{M} \phi(\mathbf{x}_k) \prod_{k=2}^{M} \psi(\mathbf{x}_k, \mathbf{x}_{k-1}), \tag{7}$$

where the partition function Z is the same as for the original model and the $\phi(\mathbf{x}_k)$'s and $\psi(\mathbf{x}_k, \mathbf{x}_{k-1})$'s are the in-column and between-column interaction potentials, respectively. In terms of the original factors of the 2-D $(1, \infty)$ run-length limited constrained channel model we get

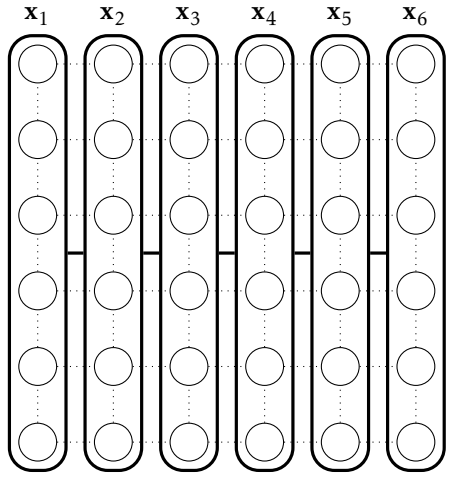
$$\phi(\mathbf{x}_k) = \prod_{j=1}^{M-1} \psi(x_{j+1,k}, x_{j,k}), \tag{8a}$$

$$\psi(\mathbf{x}_k, \mathbf{x}_{k-1}) = \prod_{j=1}^{M} \psi(x_{j,k}, x_{j,k-1}). \tag{8b}$$

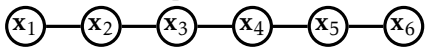
We illustrate this choice of auxiliary variables and the resulting undirected chain in Figure 2. This transformation of the PGM is a key enabler for the partition function estimation algorithm we propose in the subsequent section.

3 Sequential Monte Carlo

Sequential Monte Carlo methods, also known as particle filters, are designed to sample sequentially from some sequence of target distributions: $\bar{\gamma}_k(\mathbf{x}_{1:k})$, $k = 1, 2 \ldots$. While SMC is most commonly used for inference on directed chains, in particular for state-space models, these methods are in fact much more generally applicable. Specifically, as we shall see below, SMC can be used to simulate from the joint PMF specified by an undirected chain. Consequently, by using the representation introduced in Section 2 it is possible to apply SMC to estimate the



(a) *$M \times M$ square lattice PGM*



(b) *Corresponding M-dimensional chain*

Figure 2: *$M \times M$ square lattice graphical model converted to an M-dimensional undirected chain model.*

partition function of the 2-D $(1, \infty)$ run-length limited constrained channel. We start this section with a short introduction to SMC samplers with some known theoretical results. These results are then used to compute an unbiased estimate of the partition function. We leverage the undirected chain model with the SMC sampler and show how we can perform the necessary steps using *Forward Filtering/Backward Sampling* (FF/BS) (Carter and Kohn, 1994; Frühwirth-Schnatter, 1994). For a more thorough description of SMC methods see e.g. (Doucet and Johansen, 2011; Doucet et al., 2001).

3.1 Estimating the partition function using fully adapted SMC

We propose to use a *fully adapted* SMC algorithm (Pitt and Shephard, 1999). That the sampler is *fully adapted* means that the proposal distributions for the resampling and propagation steps are optimally chosen with respect to minimizing the variance of the importance weights, i.e. the importance weights for a fully adapted sampler are all equal. Using the optimal proposal distributions—which can significantly reduce the variance of estimators derived from the sampler—is not tractable in general. However, as we shall see below, this is in fact possible for the square lattice PGM described above.

For the undirected chain model (see Figure 2b), we let $\bar{\gamma}_k(\mathbf{x}_{1:k})$ be the PMF in-

duced by the sub-graph corresponding to the first k variables. Specifically, $\bar{\gamma}_k(\mathbf{x}_{1:k}) = \frac{\gamma_k(\mathbf{x}_{1:k})}{Z_k}$, where the unnormalized distributions $\gamma_k(\mathbf{x}_{1:k})$ are given by

$$\gamma_1(\mathbf{x}_1) = \boldsymbol{\phi}(\mathbf{x}_1), \tag{9a}$$

$$\gamma_k(\mathbf{x}_{1:k}) = \gamma_{k-1}(\mathbf{x}_{1:k-1})\boldsymbol{\phi}(\mathbf{x}_k)\boldsymbol{\psi}(\mathbf{x}_k, \mathbf{x}_{k-1}), \tag{9b}$$

with $\boldsymbol{\phi}(\cdot)$, $\boldsymbol{\psi}(\cdot)$ as defined in (8) and Z_k being the normalizing constant for $\gamma_k(\mathbf{x}_{1:k})$. We take the sequence of distributions $\bar{\gamma}_k(\mathbf{x}_{1:k})$ for $k = 1, \ldots, M$ as the target distributions for the SMC sampler. Note that $\bar{\gamma}_k(\mathbf{x}_{1:k})$ for $k < M$ is *not*, in general, a marginal distribution under $p(X)$. This is, however, not an issue since by construction $\bar{\gamma}_M(\mathbf{x}_{1:M}) = p(X)$ (where $\mathbf{x}_{1:M}$ identifies to X), i.e. at iteration $k = M$ we still recover the correct target distribution.

At iteration k, the SMC sampler approximates $\bar{\gamma}_k(\mathbf{x}_{1:k})$ by a collection of particles $\{\mathbf{x}^i_{1:k}\}_{i=1}^N$, where $\mathbf{x}_{1:k} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ is the set of all variables in column 1 through k of the PGM. These samples define an empirical point-mass approximation of the target distribution,

$$\widehat{\bar{\gamma}}^N_k(\mathbf{x}_{1:k}) := \frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{x}_{1:k} - \mathbf{x}^i_{1:k}),$$

where $\delta(x)$ is the Kronecker delta. The standard SMC algorithm produces a collection of weighted particles. However, as mentioned above, in the fully adapted setting we use a specific choice of proposal distribution and resampling probabilities, resulting in equally weighted particles (Pitt and Shephard, 1999).

Consider first the initialization at iteration $k = 1$. The auxiliary probability distribution $\bar{\gamma}_1(\mathbf{x}_1)$ corresponds to the PGM induced by the first column of the original square lattice model. That is, the graphical model for $\bar{\gamma}_1(\mathbf{x}_1)$ is a chain (the first column of Figure 2a). Consequently, we can sample from this distribution exactly, as well as compute the normalizing constant Z_1, using FF/BS. The details are given in the subsequent section. Simulating N times from $\bar{\gamma}(\mathbf{x}_1)$ results in an *equally weighted* sample $\{\mathbf{x}^i_1\}_{i=1}^N$ approximating this distribution.

We proceed inductively and assume that we have at hand a sample $\{\mathbf{x}^i_{1:k-1}\}_{i=1}^N$, approximating $\bar{\gamma}_{k-1}(\mathbf{x}_{1:k-1})$. This sample is propagated forward by simulating, conditionally independently given the particle generation up to iteration $k-1$, as follows: We decide which particle among $\{\mathbf{x}^j_{1:k-1}\}_{j=1}^N$ that should be used to generate a new particle $\mathbf{x}^i_{1:k}$ (for each $i \in \{1, \ldots, N\}$) by drawing an *ancestor index* a^i_k with probability

$$\mathbb{P}(a^i_k = j) = \frac{\nu^j_{k-1}}{\sum_l \nu^l_{k-1}}, \qquad j \in \{1, \ldots, N\}, \tag{10}$$

where ν^i_{k-1} are resampling weights. The variable a^i_k is the index of the particle at iteration $k-1$ that will be used to construct $\mathbf{x}^i_{1:k}$. Generating the ancestor indices

corresponds to a selection—or resampling—process that will put emphasis on the most likely particles. This is a crucial step of the SMC sampler. For the fully adapted sampler, the resampling weights $\nu_{k-1}^i = \nu_{k-1}(\mathbf{x}_{k-1}^i)$ are chosen in order to adapt the resampling to the *consecutive target distribution* $\bar{\gamma}_k$ (Pitt and Shephard, 1999). Intuitively, a particle $\mathbf{x}_{1:k-1}^i$ that is probable under the marginal distribution $\sum_{\mathbf{x}_k} \bar{\gamma}_k(\mathbf{x}_{1:k})$ will be assigned a large weight. Specifically, in the fully adapted algorithm we pick the resampling weights according to

$$\nu_{k-1}(\mathbf{x}_{k-1}) = \sum_{\mathbf{x}_k} \frac{\gamma_k(\mathbf{x}_{1:k})}{\gamma_{k-1}(\mathbf{x}_{1:k-1})} = \sum_{\mathbf{x}_k} \phi(\mathbf{x}_k)\psi(\mathbf{x}_k, \mathbf{x}_{k-1}). \tag{11}$$

Given the ancestors, we simulate $\mathbf{x}_k^i$ from the optimal proposal distribution: $\mathbf{x}_k^i \sim q(\cdot \mid \mathbf{x}_{k-1}^{a_k^i})$ for $i = 1, \ldots, N$, where

$$q(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \frac{\phi(\mathbf{x}_k)\psi(\mathbf{x}_k, \mathbf{x}_{k-1})}{\sum_{\mathbf{x}_k'} \phi(\mathbf{x}_k')\psi(\mathbf{x}_k', \mathbf{x}_{k-1})}. \tag{12}$$

Again, simulating from this distribution, as well as computing the resampling weights (11), can be done exactly by running FF/BS on the kth column of the model. Finally, we augment the particles as, $\mathbf{x}_{1:k}^i := (\mathbf{x}_{1:k-1}^{a_k^i}, \mathbf{x}_k^i)$. As pointed out above, with the choices (11) and (12) we obtain a collection of equally weighted particles $\{\mathbf{x}_{1:k}^i\}_{i=1}^N$, approximating $\bar{\gamma}_k(\mathbf{x}_{1:k})$.

At iteration $k = M$, the SMC sampler provides a Monte Carlo approximation of the joint PMF $p(X) = \bar{\gamma}_M(\mathbf{x}_{1:k})$. While this can be of interest on its own, we are primarily interested in the normalizing constant Z (i.e. the partition function). However, it turns out that the SMC algorithm in fact provides an estimator of Z_k as a byproduct, given by

$$\widehat{Z}_k^N := Z_1 \prod_{\ell=1}^{k-1} \left(\frac{1}{N} \sum_{i=1}^{N} \nu_\ell^i \right). \tag{13}$$

It may not be obvious to see why (3) is a natural estimator of the normalizing constant Z_k. However, it has been shown that this SMC-based estimator is unbiased for any $N \geq 1$ and $k = 1, \ldots, K$. This result is due to (Del Moral, 2004, Proposition 7.4.1). Specifically, for our 2-D constrained channel example, it follows that at the last iteration $k = M$ we have an unbiased estimator of the partition function

$$\mathbb{E}[\widehat{Z}_M^N] = Z. \tag{14}$$

Furthermore, under a weak integrability condition the estimator is asymptotically normal with a rate $\sqrt{N}$:

$$\sqrt{N}(\widehat{Z}_M^N - Z) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \tag{15}$$

where an explicit expression for σ^2 is given in (Del Moral, 2004, Proposition 9.4.1).

3.2 SMC samplers and Forward Filtering/Backward Sampling

To implement the fully adapted SMC sampler described above we are required to compute the sums involved in equations (11) and (12). By brute force calculation our method would be computationally prohibitive as the complexity is exponential in the dimensionality M of the chain. However, as we show below, it is possible to use FF/BS to efficiently carry out these summations. This development is key to our proposed solution to the problem of estimating the partition function, since the computational complexity of estimating the channel capacity is reduced from $\mathcal{O}(NM2^M)$ (brute force) to $\mathcal{O}(NM^2)$ (FF/BS).

Initially, at $k = 1$, the graph describing the target distribution $\bar{\gamma}_1(\mathbf{x}_1)$ is trivially a chain which can be sampled from exactly by using FF/BS. Additionally, the normalizing constant Z_1 can be computed in the forward pass of the FF/BS algorithm. However, this is true for any consecutive iteration k as well. Indeed, simulating $\mathbf{x}_k$ under $\bar{\gamma}_k$, conditionally on $\mathbf{x}_{1:k-1}$, again corresponds to doing inference on a chain. This means we can employ FF/BS to compute the resampling weights (11) (corresponding to a conditional normalizing constant computation) and to simulate $\mathbf{x}_k$ from the optimal proposal (12).

Let k be a fixed iteration of the SMC algorithm. The forward filtering step of FF/BS is performed by sending messages

$$m^i_{j+1}(x_{j+1,k}) = \sum_{x_{j,k}} \psi(x_{j+1,k}, x_{j,k})\psi(x_{j,k}, x^i_{j,k-1})m^i_j(x_{j,k}), \tag{16}$$

for $j = 1, \ldots, M-2$, i.e. from the top to the bottom of column k. The resampling weights are given as a byproduct from the message passing as

$$\begin{aligned} \nu_{k-1}(\mathbf{x}^i_{k-1}) &= \sum_{\mathbf{x}_k} \phi(\mathbf{x}_k)\psi(\mathbf{x}_k, \mathbf{x}^i_{k-1}) \\ &= \sum_{x_{M,k}} \psi(x_{M,k}, x^i_{M,k-1})m^i(x_{M-1,k}). \end{aligned} \tag{17}$$

After sampling the ancestor indices a^i_k as in (10), we perform backward sampling to sample the full column of variables $\mathbf{x}_k$, one at a time in reverse order $j = M, \ldots, 1$,

$$x^i_{j,k} \sim \frac{\psi(x_{j,k}, x^i_{j+1,k})\psi(x_{j,k}, x^{a^i_k}_{j,k-1})m^{a^i_k}(x_{j,k})}{\sum_{x'_{j,k}} \psi(x'_{j,k}, x^i_{j+1,k})\psi(x'_{j,k}, x^{a^i_k}_{j,k-1})m^{a^i_k}(x'_{j,k})}, \tag{18}$$

with straightforward modifications for $j = 1$ and M. This results in a draw $\mathbf{x}^i_k = (x^i_{1,k}, \ldots, x^i_{M,k})$ from the optimal proposal $q(\cdot \mid \mathbf{x}^{a^i_k}_{k-1})$ defined in (12). A summary of the resulting solution is provided in Algorithm 1. Furthermore, for increased numerical stability we pass normalized messages and compute the resampling weights and the partition function estimate directly in the log-space.

Algorithm 1: Channel capacity estimation

Perform each step for $i = 1, \ldots, N$, except setting $\widehat{Z}_k^N$.
Sample $\mathbf{x}_1^i$ using FF/BS (16), (18).
Set $\widehat{Z}_1^N = Z_1$.
for $k = 2$ **to** M **do**
  Calculate $\nu_{k-1}(\mathbf{x}_{k-1}^i)$ using forward filtering (16)-(17).
  Sample a_k^i according to (10).
  Sample $\mathbf{x}_k^i$ using backward sampling (18).
  Set $\mathbf{x}_{1:k}^i = (\mathbf{x}_{1:k-1}^{a_k^i}, \mathbf{x}_k^i)$.
  Set $\widehat{Z}_k^N = \widehat{Z}_{k-1}^N \left(\frac{1}{N} \sum_{i=1}^N \nu_{k-1}(\mathbf{x}_{k-1}^i) \right)$
end for

4 Experiments

We compare our algorithm to the state-of-the-art Monte Carlo approximation algorithm proposed in (Molkaraie and Loeliger, 2013) on the same example that they consider as explained in Section 2. Since the key enabler to the algorithm proposed in (Molkaraie and Loeliger, 2013) is tree sampling according to (Hamze and de Freitas, 2004)—a specific type of blocked Gibbs sampling—we will in the sequel refer to this algorithm as the *tree sampler*. All results are compared versus average wall-clock execution time. We run each algorithm 10 times independently to estimate error bars as well as mean-squared-errors (MSE) compared to the true value (computed using a long run of the tree sampler). For the MCMC-based tree sampler, we use a burn-in of 10% of the generated samples when estimating the capacity. The tree sampler actually gives two estimates of the capacity at each iteration; we use the average of these two when comparing to the SMC algorithm. Code for reproducing the results is available at (A. Naesseth et al., 2014).

Consider first a channel with dimension $M = 10$. We can see the results with error bars from 10 independent runs in Figure 3 of both algorithms. The rightmost data point corresponds to approximately 20k iterations/particles. Both algorithms converge to the value $C_{10} \approx 0.6082$. However, the SMC algorithm is clearly more efficient and with less error per fix computation time. We estimated the true value by running 10 independent tree samplers for 100k iterations, removed burn-in and taking the mean as our estimate.

The estimated true value was subsequently used to calculate the MSE as displayed in Figure 4. The central limit theorem for the SMC sampler (see (15)) tells us that the error should decrease at a rate of $1/N$ which is supported by this experiment. Furthermore, we can see that the SMC algorithm on average gives an order of magnitude more accurate estimate than the tree sampler per fix computation time. In our second example we scale up the model to $M = 60$,

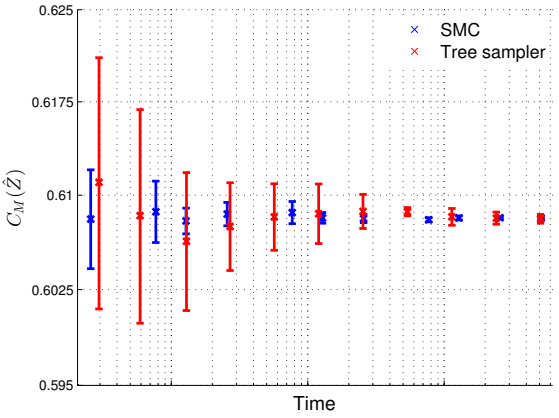


Figure 3: *Estimates of the capacity C_{10}, with error bars, based on* 10 *independet runs of our proposed SMC-based method and the tree sampler (Molkaraie and Loeliger, 2013). Plotted versus wall-clock time in log-scale. Note that this is also an upper bound on the infinite-size capacity, i.e. $C_M \geq C_\infty \approx 0.5879$.*

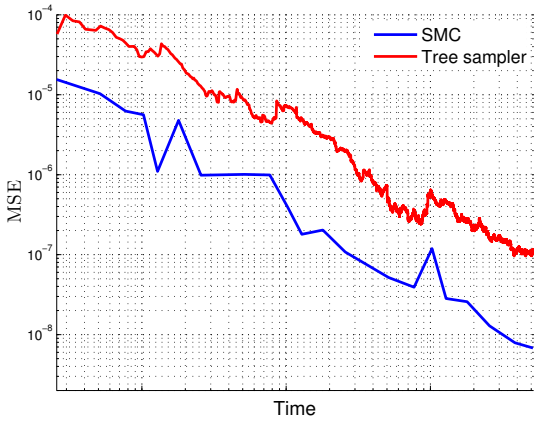


Figure 4: *Mean-squared-error of the capacity C_{10} estimates based on* 10 *independet runs of our proposed SMC-based method and the tree sampler (Molkaraie and Loeliger, 2013). Plotted versus wall-clock time in log-log-scale.*

i.e. a total of 3600 nodes as opposed to 100 in the previous example. The basic tree sampler performs poorly on this large model with very slow mixing and convergence. To remedy this problem (Molkaraie and Loeliger, 2013) propose to aggregate every W columns in the tree sampler and sample these exactly by simple enumeration, resulting in further blocking of the underlying Gibbs sampler. However, this results in an algorithm with a computational complexity exponential in W (Molkaraie and Loeliger, 2013). The same strategy can be applied to our algorithm and we compare the tree sampler and SMC for widths $W = 1$ and

3. There seems to be no gain in increasing the width higher than this for either method. The resulting MSEs[1] based on 10 independent runs of the tree sampler and the SMC algorithm are presented in Figure 5. As we can see the basic tree

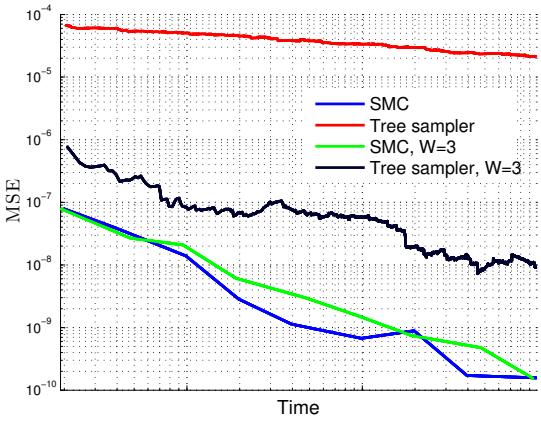


Figure 5: *Mean-squared-error of the capacity C_{60} estimates based on* 10 *independet runs of our proposed SMC-based method and the tree sampler (Molkaraie and Loeliger, 2013) for strip widths* 1 *(standard) and* 3 *respectively. Plotted versus wall-clock time in log-log-scale.*

sampler converges very slowly, in line with results from (Molkaraie and Loeliger, 2013). On the other hand, our proposed SMC sampling method performs very well, even with $W = 1$, and on average it has more than an order-of-magnitude smaller error than the tree sampler with $W = 3$. In comparing the two different SMC methods there seems to be no apparent gain in increasing the width of the strips added at each iteration in this case.

5 Conclusions

We have introduced an SMC method to compute the noiseless capacity of two-dimensional channel models. The proposed algorithm was shown to improve upon a state-of-the-art Monte Carlo estimation method by more than an order-of-magnitude. Furthermore, while this improvement was obtained using a sequential implementation, the SMC method is easily parallelizable over the particles (which is not the case for the MCMC-based tree sampler), offering further improvements by making use of modern computational architectures. This gain is of significant importance because the running time can be on the order of days for realistic scenarios. Extensions to calculate the information rate of noisy 2-

[1] For this model the basic tree sampler converges too slowly and the tree sampler with $W = 3$ was too computationally demanding to provide an accurate estimate of the "true" value. For this reason, we estimate the true value by averaging 10 independent runs of SMC with $N = 200$k.

D source/channel models by the method proposed in (Molkaraie and Loeliger, 2013) are straightforward.

Acknowledgment

Supported by the projects *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524) and *Learning of complex dynamical systems* (Contract number: 637-2014-466), both funded by the Swedish Research Council. We would also like to thank Lukas Bruderer for suggesting the application.

Bibliography

C. A. Naesseth, F. Lindsten, and T. B. Schön. smc2d, 2014. URL http://dx.doi.org/10.5281/zenodo.11538.

Chris K Carter and Robert Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

J. Dauwels and H.-A. Loeliger. Computation of information rates by particle methods. *IEEE Transactions on Information Theory*, 54(1):406–409, Jan 2008.

P. Del Moral. *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications*. Springer, 2004.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

Arnaud Doucet, Nando De Freitas, and Neil Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer, 2001.

Sylvia Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, 15(2):183–202, 1994.

F. Hamze and N. de Freitas. From fields to trees. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, Banff, Canada, July 2004.

Kees A Schouhamer Immink. *Codes for mass data storage systems*. Shannon Foundation Publisher, 2004.

Akiko Kato and Kenneth Zeger. On the capacity of two-dimensional run-length constrained channels. *IEEE Transactions on Information Theory*, 45(5):1527–1540, 1999.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Hans-Andrea Loeliger and Mehdi Molkaraie. Estimating the partition function of 2-D fields and the capacity of constrained noiseless 2-D channels using tree-based Gibbs sampling. In *Proceedings of the IEEE Information Theory Workshop*, pages 11–16, Taormina, Italy, October 2009.

M. Molkaraie and H.-A. Loeliger. Monte Carlo algorithms for the partition function and information rates of two-dimensional channels. *IEEE Transactions on Information Theory*, 59(1):495–503, 2013.

Christian A Naesseth, Fredrik Lindsten, and Thomas B Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *Information Theory Workshop (ITW), 2014 IEEE*, pages 431–435. IEEE, 2014.

Zsigmond Nagy and Kenneth Zeger. Capacity bounds for the three-dimensional run length limited channel. *IEEE Transactions on Information Theory*, 46(3): 1030–1033, 2000.

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

Giovanni Sabato and Mehdi Molkaraie. Generalized belief propagation for the noiseless capacity and information rates of run-length limited constraints. *IEEE Transactions on Communications*, 60(3):669–675, 2012.

Paul H. Siegel. Information-theoretic limits of two-dimensional optical recording channels. In *Proceedings of SPIE, the International Society for Optical Engineering*, 2006.

Paper C

Sequential Monte Carlo for Graphical Models

Authors: Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön

Edited version of the paper:

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1862–1870, 2014b.

This paper has been formatted to fit this layout.

Sequential Monte Carlo for Graphical Models

Christian A. Naesseth*, Fredrik Lindsten†, and Thomas B. Schön†

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

†Dept. of Information Technology
Uppsala University
Uppsala, Sweden
{fredrik.lindsten,thomas.schon}@it.uu.se

Abstract

We propose a new framework for how to use sequential Monte Carlo (SMC) algorithms for inference in probabilistic graphical models (PGM). Via a sequential decomposition of the PGM we find a sequence of auxiliary distributions defined on a monotonically increasing sequence of probability spaces. By targeting these auxiliary distributions using SMC we are able to approximate the full joint distribution defined by the PGM. One of the key merits of the SMC sampler is that it provides an unbiased estimate of the partition function of the model. We also show how it can be used within a particle Markov chain Monte Carlo framework in order to construct high-dimensional block-sampling algorithms for general PGMs.

1 Introduction

Bayesian inference in statistical models involving a large number of latent random variables is in general a difficult problem. This renders inference methods that are capable of efficiently utilizing structure important tools. Probabilistic Graphical Models (PGMs) are an intuitive and useful way to represent and make use of underlying structure in probability distributions with many interesting areas of applications (Jordan, 2004).

Our main contribution is a new framework for constructing non-standard (auxiliary) target distributions of PGMs, utilizing what we call a *sequential decomposition* of the underlying factor graph, to be targeted by a sequential Monte Carlo

(SMC) sampler. This construction enables us to make use of SMC methods developed and studied over the last 20 years, to approximate the full joint distribution defined by the PGM. As a byproduct, the SMC algorithm provides an unbiased estimate of the partition function (normalization constant). We show how the proposed method can be used as an alternative to standard methods such as the Annealed Importance Sampling (AIS) proposed in (Neal, 2001), when estimating the partition function. We also make use of the proposed SMC algorithm to design efficient, high-dimensional MCMC kernels for the latent variables of the PGM in a particle MCMC framework. This enables inference about the latent variables as well as learning of unknown model parameters in an MCMC setting.

During the last decade there has been substantial work on how to leverage SMC algorithms Doucet et al. (2001) to solve inference problems in PGMs. The first approaches were PAMPAS (Isard, 2003) and nonparametric belief propagation by Sudderth et al. (2003, 2010). Since then, several different variants and refinements have been proposed by e.g. Briers et al. (2005); Frank et al. (2009); Ihler and Mcallester (2009). They all rely on various particle approximations of messages sent in a loopy belief propagation algorithm. This means that in general, even in the limit of Monte Carlo samples, they are approximate methods. Compared to these approaches our proposed methods are consistent and provide an unbiased estimate of the normalization constant as a by-product.

Another branch of SMC-based methods for graphical models has been suggested by Hamze and de Freitas (2005). Their method builds on the SMC sampler by Del Moral et al. (2006), where the initial target is a spanning tree of the original graph and subsequent steps add edges according to an annealing schedule. Everitt (2012) extends these ideas to learn parameters using particle MCMC Andrieu et al. (2010). Yet another take is provided by Carbonetto and de Freitas (2007), where an SMC sampler is combined with mean field approximations. Compared to these methods we can handle both non-Gaussian and/or non-discrete interactions between variables and there is no requirement to perform MCMC steps within each SMC step.

The left-right methods described by Wallach et al. (2009) and extended by Buntine (2009) to estimate the likelihood of held-out documents in topic models are somewhat related in that they are SMC-inspired. However, these are not actual SMC algorithms and they do not produce an unbiased estimate of the partition function for finite sample set. On the other hand, a particle learning based approach was recently proposed by Scott and Baldridge (2009) and it can be viewed as a special case of our method for this specific type of model.

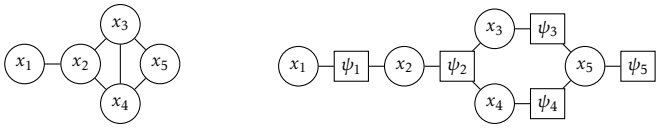
2 Graphical models

A graphical model is a probabilistic model which *factorizes* according to the structure of an underlying graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. By this

we mean that the joint probability density function (PDF) of the set of random variables indexed by $\mathcal{V}$, $X_\mathcal{V} := \{x_1, \ldots, x_{|\mathcal{V}|}\}$, can be represented as a product of factors over the cliques of the graph:

$$p(X_\mathcal{V}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(X_C), \tag{1}$$

where $\mathcal{C}$ is the set of cliques in $\mathcal{G}$, ψ_C is the factor for clique C, and $Z = \int \prod_{C \in \mathcal{C}} \psi_C(x_C) \mathrm{d}X_\mathcal{V}$ is the partition function.



(a) *Undirected graph.* ***(b)*** *Factor graph.*

Figure 1: *Undirected PGM and a corresponding factor graph.*

We will frequently use the notation $X_I = \bigcup_{i \in I} \{x_i\}$ for some subset $I \subseteq \{1, \ldots, |\mathcal{V}|\}$ and we write X_I for the range of X_I (i.e., $X_I \in \mathsf{X}_I$). To make the interactions between the random variables explicit we define a *factor graph* $\mathcal{F} = \{\mathcal{V}, \Psi, \mathcal{E}'\}$ corresponding to $\mathcal{G}$. The factor graph consists of two types of vertices, the original set of random variables $X_\mathcal{V}$ and the factors $\Psi = \{\psi_C : C \in \mathcal{C}\}$. The edge set $\mathcal{E}'$ consists only of edges from variables to factors. In Figure 1a we show a simple toy example of an undirected graphical model, and one possible corresponding factor graph, Figure 1b, making the dependencies explicit. Both directed and undirected graphs can be represented by factor graphs.

3 Sequential Monte Carlo

In this section we propose a way to sequentially decompose a graphical model which we then make use of to design an SMC algorithm for the PGM.

3.1 Sequential decomposition of graphical models

SMC methods can be used to approximate a sequence of probability distributions on a sequence of probability spaces of increasing dimension. This is done by recursively updating a set of samples—or *particles*—with corresponding nonnegative importance weights. The typical scenario is that of state inference in state-space models, where the probability distributions targeted by the SMC sampler are the joint smoothing distributions of a sequence of latent states conditionally on a sequence of observations; see e.g., Doucet and Johansen (2011) for applications of this type. However, SMC is not limited to these cases and it is applicable to a much wider class of models.

To be able to use SMC for inference in PGMs we have to define a sequence of target distributions. However, these target distributions *do not* have to be marginal distributions under $p(X_\mathcal{V})$. Indeed, as long as the sequence of target distributions is constructed in such a way that, at some final iteration, we recover $p(X_\mathcal{V})$, all the intermediate target distributions may be chosen quite arbitrarily.

This is key to our development, since it lets us use the structure of the PGM to define a sequence of intermediate target distributions for the sampler. We do this by a so called *sequential decomposition* of the graphical model. This amounts to simply adding factors to the target distribution, from the product of factors in (1), at each step of the algorithm and iterate until all the factors have been added. Constructing an artificial sequence of intermediate target distributions for an SMC sampler is a simple, albeit underutilized, idea as it opens up for using SMC samplers for inference in a wide range of probabilistic models; see e.g., Bouchard-Côté et al. (2012); Del Moral et al. (2006) for a few applications of this approach.

Given a graph $\mathcal{G}$ with cliques $\mathcal{C}$, let $\{\psi_k\}_{k=1}^K$ be a sequence of factors defined as follows $\psi_k(X_{\mathcal{I}_k}) = \prod_{C\in\mathcal{C}_k}\psi_C(X_C)$, where $\mathcal{C}_k \subset \mathcal{C}$ are chosen such that $\bigcup_{k=1}^K \mathcal{C}_k = \mathcal{C}$ and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset,\ i \neq j$, and where $\mathcal{I}_k \subseteq \{1,\ldots,|\mathcal{V}|\}$ is the index set of the variables in the domain of ψ_k, $\mathcal{I}_k = \bigcup_{C\in\mathcal{C}_k} C$. We emphasize that the cliques in $\mathcal{C}$ need not be maximal. In fact even auxiliary factors may be introduced to allow for e.g. annealing between distributions. It follows that the PDF in (1) can be written as $p(X_\mathcal{V}) = \frac{1}{Z}\prod_{k=1}^K \psi_k(X_{\mathcal{I}_k})$. Principally, the choices and the ordering of the $\mathcal{C}_k$'s is arbitrary, but in practice it will affect the performance of the proposed sampler. However, in many common PGMs an intuitive ordering can be deduced from the structure of the model, see Section 6.

The sequential decomposition of the PGM is then based on the auxiliary quantities $\widetilde{\gamma}_k(X_{\mathcal{L}_k}) := \prod_{\ell=1}^k \psi_\ell(X_{\mathcal{I}_\ell})$, with $\mathcal{L}_k := \bigcup_{\ell=1}^k \mathcal{I}_\ell$, for $k \in \{1,\ldots,K\}$. By construction, $\mathcal{L}_K = \mathcal{V}$ and the joint PDF $p(X_{\mathcal{L}_K})$ will be proportional to $\widetilde{\gamma}_K(X_{\mathcal{L}_K})$. Consequently, by using $\widetilde{\gamma}_k(X_{\mathcal{L}_k})$ as the basis for the target sequence for an SMC sampler, we will obtain the correct target distribution at iteration K. However, a further requirement for this to be possible is that all the functions in the sequence are normalizable. For many graphical models this is indeed the case, and then we can use $\widetilde{\gamma}_k(X_{\mathcal{L}_k})$, $k = 1$ to K, directly as our sequence of intermediate target densities. If, however, $\int \widetilde{\gamma}_k(X_{\mathcal{L}_k})\mathrm{d}X_{\mathcal{L}_k} = \infty$ for some $k < K$, an easy remedy is to modify the target density to ensure normalizability. This is done by setting $\gamma_k(X_{\mathcal{L}_k}) = \widetilde{\gamma}_k(X_{\mathcal{L}_k})q_k(X_{\mathcal{L}_k})$, where $q_k(X_{\mathcal{L}_k})$ is choosen so that $\int \gamma_k(X_{\mathcal{L}_k})\mathrm{d}X_{\mathcal{L}_k} < \infty$. We set $q_K(X_{\mathcal{L}_K}) \equiv 1$ to make sure that $\gamma_K(X_{\mathcal{L}_K}) \propto p(X_{\mathcal{L}_k})$. Note that the integral $\int \gamma_k(X_{\mathcal{L}_k})\mathrm{d}X_{\mathcal{L}_k}$ need not be computed explicitly, as long as it can be established that it is finite. With this modification we obtain a sequence of unnormalized intermediate target densities for the SMC sampler as $\gamma_1(X_{\mathcal{L}_1}) = q_1(X_{\mathcal{L}_1})\psi_1(X_{\mathcal{L}_1})$ and $\gamma_k(X_{\mathcal{L}_k}) = \gamma_{k-1}(X_{\mathcal{L}_{k-1}})\frac{q_k(X_{\mathcal{L}_k})}{q_{k-1}(X_{\mathcal{L}_{k-1}})}\psi_k(X_{\mathcal{I}_k})$ for $k = 2,\ldots,K$. The corresponding normalized PDFs are given by $\bar{\gamma}_k(X_{\mathcal{L}_k}) = \gamma_k(X_{\mathcal{L}_k})/Z_k$, where $Z_k = \int \gamma_k(X_{\mathcal{L}_k})\mathrm{d}X_{\mathcal{L}_k}$. Figure 2 shows two examples of possible subgraphs when applying the decompo-

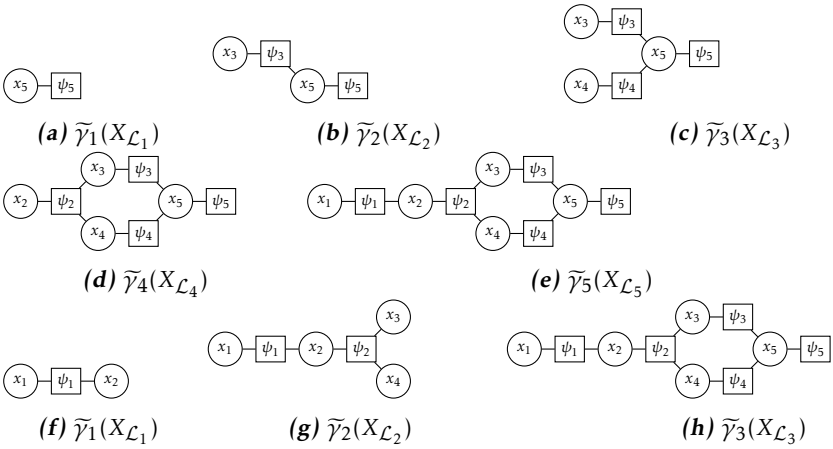


Figure 2: *Examples of five- (top) and three-step (bottom) sequential decomposition of Figure 1.*

sition, in two different ways, to the factor graph example in Figure 1.

3.2 Sequential Monte Carlo for PGMs

At iteration k, the SMC sampler approximates the target distribution $\bar{\gamma}_k$ by a collection of weighted particles $\{X^i_{\mathcal{L}_k}, w^i_k\}_{i=1}^N$. These samples define an empirical point-mass approximation of the target distribution. In what follows, we shall use the notation $\xi_k := X_{\mathcal{I}_k \setminus \mathcal{L}_{k-1}}$ to refer to the collection of random variables that are in the domain of γ_k, but not in the domain of γ_{k-1}. This corresponds to the collection of random variables, with which the particles are augmented at each iteration.

Initially, $\bar{\gamma}_1$ is approximated by importance sampling. We proceed inductively and assume that we have at hand a weighted sample $\{X^i_{\mathcal{L}_{k-1}}, w^i_{k-1}\}_{i=1}^N$, approximating $\bar{\gamma}_{k-1}(X_{\mathcal{L}_{k-1}})$. This sample is propagated forward by simulating, conditionally independently given the particle generation up to iteration $k-1$, and drawing an *ancestor index* a^i_k with $\mathbb{P}(a^i_k = j) \propto \nu^j_{k-1} w^j_{k-1}$, $j = 1, \ldots, N$, where $\nu^i_{k-1} := \nu_{k-1}(X^i_{\mathcal{L}_{k-1}})$—known as adjustment multiplier weights—are used in the auxiliary SMC framework to adapt the resampling procedure to the current target density $\bar{\gamma}_k$ (Pitt and Shephard, 1999). Given the ancestor indices, we simulate particle increments $\{\xi^i_k\}_{i=1}^N$ from a proposal density $\xi^i_k \sim r_k(\cdot | X^{a^i_k}_{\mathcal{L}_{k-1}})$ on $\mathsf{X}_{\mathcal{I}_k \setminus \mathcal{L}_{k-1}}$, and augment the particles as $X^i_{\mathcal{L}_k} := X^{a^i_k}_{\mathcal{L}_{k-1}} \cup \xi^i_k$.

After having performed this procedure for the N ancestor indices and particles, they are assigned importance weights $w^i_k = W_k(X^i_{\mathcal{L}_k})$. The weight function, for

$k \geq 2$, is given by

$$W_k(X_{\mathcal{L}_k}) = \frac{\gamma_k(X_{\mathcal{L}_k})}{\gamma_{k-1}(X_{\mathcal{L}_{k-1}})\nu_{k-1}(X_{\mathcal{L}_{k-1}})r_k(\xi_k|X_{\mathcal{L}_{k-1}})}, \tag{2}$$

where, again, we write $\xi_k = X_{\mathcal{I}_k \setminus \mathcal{L}_{k-1}}$. We give a summary of the SMC method in Algorithm 1.

Algorithm 1: Sequential Monte Carlo (SMC)

Perform each step for $i = 1, \ldots, N$.
Sample $X^i_{\mathcal{L}_1} \sim r_1(\cdot)$.
Set $w^i_1 = \gamma_1(X^i_{\mathcal{L}_1})/r_1(X^i_{\mathcal{L}_1})$.
for $k = 2$ **to** K **do**
  Sample a^i_k according to $\mathbb{P}(a^i_k = j) = \frac{\nu^j_{k-1} w^j_{k-1}}{\sum_l \nu^l_{k-1} w^l_{k-1}}$.
  Sample $\xi^i_k \sim r_k(\cdot|X^{a^i_k}_{\mathcal{L}_{k-1}})$ and set $X^i_{\mathcal{L}_k} = X^{a^i_k}_{\mathcal{L}_{k-1}} \cup \xi^i_k$.
  Set $w^i_k = W_k(X^i_{\mathcal{L}_k})$.
end for

In the case that $\mathcal{I}_k \setminus \mathcal{L}_{k-1} = \emptyset$ for some k, resampling and propagation steps are superfluous. The easiest way to handle this is to simply skip these steps and directly compute importance weights. An alternative approach is to bridge the two target distributions $\bar{\gamma}_{k-1}$ and $\bar{\gamma}_k$ similarly to Del Moral et al. (2006).

Since the proposed sampler for PGMs falls within a general SMC framework, standard convergence analysis applies. See e.g., Del Moral (2004) for a comprehensive collection of theoretical results on consistency, central limit theorems, and non-asymptotic bounds for SMC samplers.

The choices of proposal density and adjustment multipliers can quite significantly affect the performance of the sampler. It follows from (2) that $W_k(X_{\mathcal{L}_k}) \equiv 1$ if we choose $\nu_{k-1}(X_{\mathcal{L}_{k-1}}) = \int \frac{\gamma_k(X_{\mathcal{L}_k})}{\gamma_{k-1}(X_{\mathcal{L}_{k-1}})} \mathrm{d}\xi_k$ and $r_k(\xi_k|X_{\mathcal{L}_{k-1}}) = \frac{\gamma_k(X_{\mathcal{L}_k})}{\nu_{k-1}(X_{\mathcal{L}_{k-1}})\gamma_{k-1}(X_{\mathcal{L}_{k-1}})}$. In this case, the SMC sampler is said to be *fully adapted*.

3.3 Estimating the partition function

The partition function of a graphical model is a very interesting quantity in many applications. Examples include likelihood-based learning of the parameters of the PGM, statistical mechanics where it is related to the free energy of a system of objects, and information theory where it is related to the capacity of a channel. However, as stated by Hamze and de Freitas (2005), estimating the partition function of a loopy graphical model is a "notoriously difficult" task. Indeed, even for discrete problems simple and accurate estimators have proved to be elusive, and MCMC methods do not provide any simple way of computing the partition function.

On the contrary, SMC provides a straightforward estimator of the normalizing constant (i.e. the partition function), given as a byproduct of the sampler according to,

$$\widehat{Z}_k^N := \left(\frac{1}{N}\sum_{i=1}^{N} w_k^i\right)\left\{\prod_{\ell=1}^{k-1}\frac{1}{N}\sum_{i=1}^{N} v_\ell^i w_\ell^i\right\}. \tag{3}$$

It may not be obvious to see why (3) is a natural estimator of the normalizing constant Z_k. However, a by now well known result is that this SMC-based estimator is unbiased. This result is due to Del Moral (2004, Proposition 7.4.1) and, for the special case of inference in state-space models, it has also been established by Pitt et al. (2012). For completeness we also offer a proof using the present notation in the supplementary material. Since $Z_K = Z$, we thus obtain an estimator of the partition function of the PGM at iteration K of the sampler. Besides from being unbiased, this estimator is also consistent and asymptotically normal; see Del Moral (2004).

In (Naesseth et al., 2014) we have studied a specific information theoretic application (computing the capacity of a two-dimensional channel) and inspired by the algorithm proposed here we were able to design a sampler with significantly improved performance compared to the previous state-of-the-art.

4 Particle MCMC and partial blocking

Two shortcomings of SMC are: *(i)* it does not solve the parameter learning problem, and *(ii)* the quality of the estimates of marginal distributions $p(X_{\mathcal{L}_k}) = \int \bar{\gamma}_K(X_{\mathcal{L}_K})\mathrm{d}X_{\mathcal{L}_K\setminus\mathcal{L}_k}$ deteriorates for $k \ll K$ due to the fact that the particle trajectories degenerate as the particle system evolves (see e.g., Doucet and Johansen (2011)). Many methods have been proposed in the literature to address these problems; see e.g. Lindsten and Schön (2013) and the references therein. Among these, the recently proposed particle MCMC (PMCMC) framework Andrieu et al. (2010), plays a prominent role. PMCMC algorithms make use of SMC to construct (in general) high-dimensional Markov kernels that can be used within MCMC. These methods were shown by Andrieu et al. (2010) to be exact, in the sense that the apparent particle approximation in the construction of the kernel does not change its invariant distribution. This property holds for any number of particles $N \geq 2$, i.e., PMCMC does not rely on asymptotics in N for correctness.

The fact that the SMC sampler for PGMs presented in Algorithm 1 fits under a general SMC umbrella implies that we can also straightforwardly make use of this algorithm within PMCMC. This allows us to construct a Markov kernel (indexed by the number of particles N) on the space of latent variables of the PGM, $P_N(X'_{\mathcal{L}_K}, \mathrm{d}X_{\mathcal{L}_K})$, which leaves the full joint distribution $p(X_{\mathcal{V}})$ invariant. We do not dwell on the details of the implementation here, but refer instead to Andrieu et al. (2010) for the general setup and Lindsten et al. (2014) for the specific

method that we have used in the numerical illustration in Section 6.

PMCMC methods enable blocking of the latent variables of the PGM in an MCMC scheme. Simulating all the latent variables $X_{\mathcal{L}_K}$ jointly is useful since, in general, this will reduce the autocorrelation when compared to simulating the variables x_j one at a time (Robert and Casella, 2004). However, it is also possible to employ PMCMC to construct an algorithm in between these two extremes, a strategy that we believe will be particularly useful in the context of PGMs. Let $\{\mathcal{V}^m,\ m \in \{1, \ldots, M\}\}$ be a partition of $\mathcal{V}$. Ideally, a Gibbs sampler for the joint distribution $p(X_\mathcal{V})$ could then be constructed by simulating, using a systematic or a random scan, from the conditional distributions

$$p(X_{\mathcal{V}^m}|X_{\mathcal{V}\setminus\mathcal{V}^m}) \text{ for } m = 1, \ldots, M. \tag{4}$$

We refer to this strategy as *partial blocking*, since it amounts to simulating a subset of the variables, but not necessarily all of them, jointly. Note that, if we set $M = |\mathcal{V}|$ and $\mathcal{V}^m = \{m\}$ for $m = 1, \ldots, M$, this scheme reduces to a standard Gibbs sampler. On the other extreme, with $M = 1$ and $\mathcal{V}^1 = \mathcal{V}$, we get a fully blocked sampler which targets directly the full joint distribution $p(X_\mathcal{V})$.

From (1) it follows that the conditional distributions (4) can be expressed as

$$p(X_{\mathcal{V}^m}|X_{\mathcal{V}\setminus\mathcal{V}^m}) \propto \prod_{C\in\mathcal{C}^m} \psi_C(X_C), \tag{5}$$

where $\mathcal{C}^m = \{C \in \mathcal{C} : C \cap \mathcal{V}^m \neq \emptyset\}$. While it is in general not possible to sample exactly from these conditionals, we can make use of PMCMC to facilitate a partially blocked Gibbs sampler for a PGM. By letting $p(X_{\mathcal{V}^m}|X_{\mathcal{V}\setminus\mathcal{V}^m})$ be the target distribution for the SMC sampler of Algorithm 1, we can construct a PMCMC kernel P_N^m that leaves the conditional distribution (5) invariant. This suggests the following approach: with $X'_\mathcal{V}$ being the current state of the Markov chain, update block m by sampling

$$X_{\mathcal{V}^m} \sim P_N^m\langle X'_{\mathcal{V}\setminus\mathcal{V}^m}\rangle(X'_{\mathcal{V}^m}, \cdot). \tag{6}$$

Here we have indicated explicitly in the notation that the PMCMC kernel for the conditional distribution $p(X_{\mathcal{V}^m}|X_{\mathcal{V}\setminus\mathcal{V}^m})$ depends on both $X'_{\mathcal{V}\setminus\mathcal{V}^m}$ (which is considered to be fixed throughout the sampling procedure) and on $X'_{\mathcal{V}^m}$ (which defines the current state of the PMCMC procedure).

As mentioned above, while being generally applicable, we believe that partial blocking of PMCMC samplers will be particularly useful for PGMs. The reason is that we can choose the vertex sets $\mathcal{V}^m$ for $m = 1, \ldots, M$ in order to facilitate simple sequential decompositions of the induced subgraphs. For instance, it is always possible to choose the partition in such a way that all the induced subgraphs are chains.

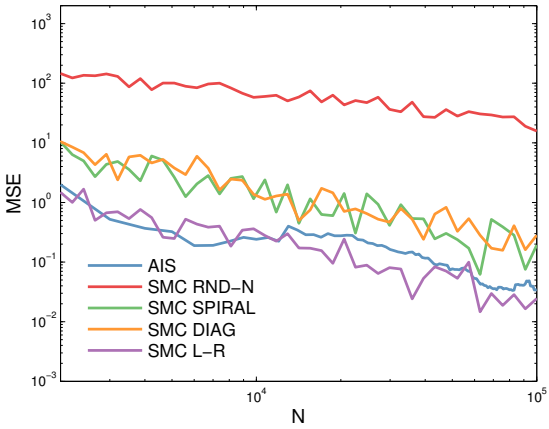


Figure 3: *Mean-squared-errors for sample size N in the estimates of* $\log Z$ *for AIS and four different orderings in the proposed SMC framework.*

5 Experiments

In this section we evaluate the proposed SMC sampler on three examples to illustrate the merits of our approach. Additional details and results are available in the supplementary material and code to reproduce results can be found in (A. Naesseth et al., 2014). We first consider an example from statistical mechanics, the classical XY model, to illustrate the impact of the sequential decomposition. Furthermore, we profile our algorithm with the "gold standard" AIS (Neal, 2001) and Annealed Sequential Importance Resampling (ASIR[1]) (Del Moral et al., 2006). In the second example we apply the proposed method to the problem of scoring of topic models, and finally we consider a simple toy model, a Gaussian Markov random field (MRF), which illustrates that our proposed method has the potential to significantly decrease correlations between samples in an MCMC scheme. Furthermore, we provide an *exact* SMC-approximation of the tree-sampler by Hamze and de Freitas (2004) and thereby extend the scope of this powerful method.

5.1 Classical XY model

The classical XY model (see e.g. (Kosterlitz and Thouless, 1973)) is a member in the family of *n-vector* models used in statistical mechanics. It can be seen as a generalization of the well known Ising model with a two-dimensional electromagnetic spin. The spin vector is described by its angle $x \in (-\pi, \pi]$. We will consider

[1] ASIR is a specific instance of the *SMC sampler* by (Del Moral et al., 2006), corresponding to AIS with the addition of resampling steps, but to avoid confusion with the proposed method we choose to refer to it as ASIR.

square lattices with periodic boundary conditions. The joint PDF of the classical XY model with equal interaction is given by

$$p(X_{\mathcal{V}}) \propto \mathrm{e}^{\beta \sum_{(i,j)\in\mathcal{E}} \cos(x_i - x_j)}, \tag{7}$$

where β denotes the inverse temperature.

To evaluate the effect of different sequence orders on the accuracy of the estimates of the log-normalizing-constant $\log Z$ we ran several experiments on a 16×16 XY model with $\beta = 1.1$ (approximately the critical inverse temperature (Tomita and Okabe, 2002)). For simplicity we add one node at a time and all factors bridging this node with previously added nodes. Full adaptation in this case is possible due to the optimal proposal being a von Mises distribution. We show results for the following cases: *Random neighbour (RND-N)* First node selected randomly among all nodes, concurrent nodes selected randomly from the set of nodes with a neighbour in $X_{\mathcal{L}_{k-1}}$. *Diagonal (DIAG)* Nodes added by traversing diagonally (45° angle) from left to right. *Spiral (SPIRAL)* Nodes added spiralling in towards the middle from the edges. *Left-Right (L-R)* Nodes added by traversing the graph left to right, from top to bottom.

We also give results of AIS with single-site-Gibbs updates and 1 000 annealing distributions linearly spaced from zero to one, starting from a uniform distribution (geometric spacing did not yield any improvement over linear spacing for this case). The "true value" was estimated using AIS with 10 000 intermediate distributions and 5 000 importance samples. We can see from the results in Figure 3 that designing a good sequential decomposition for the SMC sampler is important. However, the intuitive and fairly simple choice L-R does give very good results comparable to that of AIS.

Furthermore, we consider a larger size of 64×64 and evaluate the performance of the L-R ordering compared to AIS and the ASIR method. Figure 4 displays box-plots of 10 independent runs. We set $N = 10^5$ for the proposed SMC sampler and then match the computational costs of AIS and ASIR with this computational budget. A fair amount of time was spent in tuning the AIS and ASIR algorithms; 10 000 linear annealing distributions seemed to give best performance in these cases. We can see that the L-R ordering gives results comparable to fairly well-tuned AIS and ASIR algorithms; the ordering of the methods depending on the temperature of the model. One option that does make the SMC algorithm interesting for these types of applications is that it can easily be parallelized over the particles, whereas AIS/ASIR has limited possibilities of parallel implementation over the (crucial) annealing steps.

5.2 Likelihood estimation in topic models

Topic models such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003) are popular models for reasoning about large text corpora.

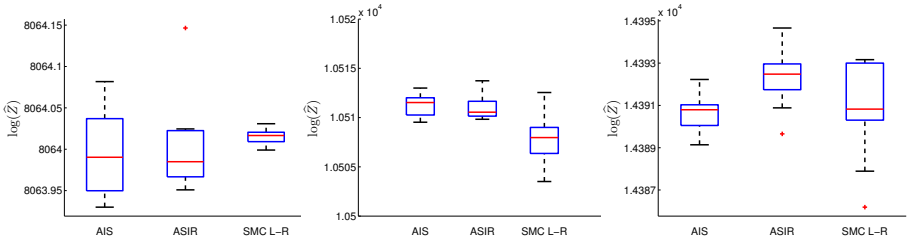


Figure 4: *The logarithm of the estimated partition function for the* 64×64 *XY model with inverse temperature* 0.5 *(left),* 1.1 *(middle) and* 1.7 *(right).*

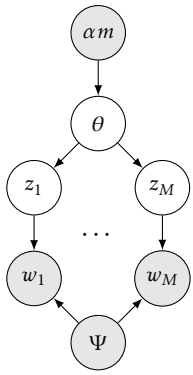
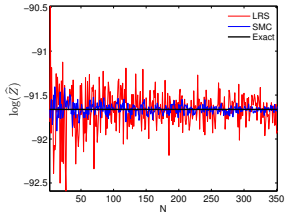


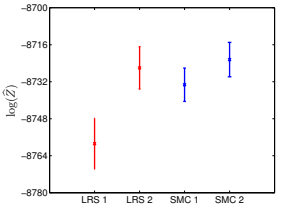
Figure 5: *LDA as graphical model.*

Model evaluation is often conducted by computing the likelihood of held-out documents w.r.t. a learnt model. However, this is a challenging problem on its own—which has received much recent interest (Buntine, 2009; Scott and Baldridge, 2009; Wallach et al., 2009)—since it essentially corresponds to computing the partition function of a graphical model; see Figure 5. The SMC procedure of Algorithm 1 can used to solve this problem by defining a sequential decomposition of the graphical model. In particular, we consider the decomposition corresponding to first including the node θ and then, subsequently, introducing the nodes z_1 to z_M in any order. Interestingly, if we then make use of a Rao-Blackwellization over the variable θ, the SMC sampler of Algorithm 1 reduces exactly to a method that has previously been proposed for this specific problem (Scott and Baldridge, 2009). In (Scott and Baldridge, 2009), the method is derived by reformulating the model in terms of its sufficient statistics and phrasing this as a particle learning problem; here we obtain the same procedure as a special case of the general SMC algorithm operating on the original model.

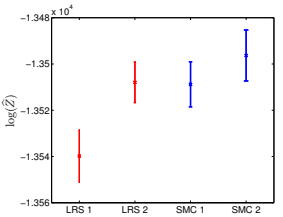
We use the same data and learnt models as Wallach et al. (2009), i.e. 20 newsgroups, and PubMed Central abstracts (PMC). We compare with the Left-Right-Sequential (LRS) sampler (Buntine, 2009), which is an improvement over the method proposed by Wallach et al. (2009). Results on simulated and real data experiments are provided in Figure 6. For the simulated example (Figure 6a), we use a small model with 10 words and 4 topics to be able to compute the exact log-likelihood. We keep the number of particles in the SMC algorithm equal to the number of Gibbs steps in LRS; this means LRS is about an order-of-magnitude more computationally demanding than the SMC method. Despite the fact that the SMC sampler uses only about a tenth of the computational time of the LRS sampler, it performs significantly better in terms of estimator variance. The other two plots show results on real data with 10 held-out documents for each dataset. For a fixed number of Gibbs steps we choose the number of particles for each document to make the computational cost approximately equal. Run #2 has twice



(a) Small simulated example.



(b) PMC.



(c) 20 newsgroups.

Figure 6: *Estimates of the log-likelihood of heldout documents for various datasets.*

the number of particles/samples as in run #1. We show the mean of 10 runs and error-bars estimated using bootstrapping with 10 000 samples. Computing the logarithm of $\hat{Z}$ introduces a negative bias, which means larger values of $\log \hat{Z}$ typically implies more accurate results. The results on real data do not show the drastic improvement we see in the simulated example, which could be due to degeneracy problems for long documents. An interesting approach that could improve results would be to use an SMC algorithm tailored to discrete distributions, e.g. Fearnhead and Clifford (2003).

5.3 Gaussian MRF

Finally, we consider a simple toy model to illustrate how the SMC sampler of Algorithm 1 can be incorporated in PMCMC sampling. We simulate data from a zero mean Gaussian 10×10 lattice MRF with observation and interaction standard deviations of $\sigma_i = 1$ and $\sigma_{ij} = 0.1$ respectively. We use the proposed SMC algorithm together with the PMCMC method by Lindsten et al. (2014). We compare this with standard Gibbs sampling and the tree sampler by Hamze and de Freitas (2004). We use a moderate number of $N = 50$ particles in the PMCMC sampler (recall that it admits the correct invariant distribution for any $N \geq 2$). In Figure 7 we can see the empirical autocorrelation funtions (ACF) centered around the true posterior mean for variable x_{82} (selected randomly from among $X_{\mathcal{V}}$; similar results hold for all the variables of the model). Due to the strong interaction between the latent variables, the samples generated by the standard Gibbs sampler are strongly correlated. Tree-sampling and PMCMC with partial blocking show nearly identical gains compared to Gibbs. This is interesting, since it suggest that simulating from the SMC-based PMCMC kernel can be almost as efficient as exact simulation, even using a moderate number of particles. Indeed, PMCMC with partial blocking can be viewed as an *exact* SMC-approximation of the tree sampler, extending the scope of tree-sampling beyond discrete and Gaussian models. The fully blocked PMCMC algorithm achieves the best ACF, dropping off to zero considerably faster than for the other methods. This is not surprising since this

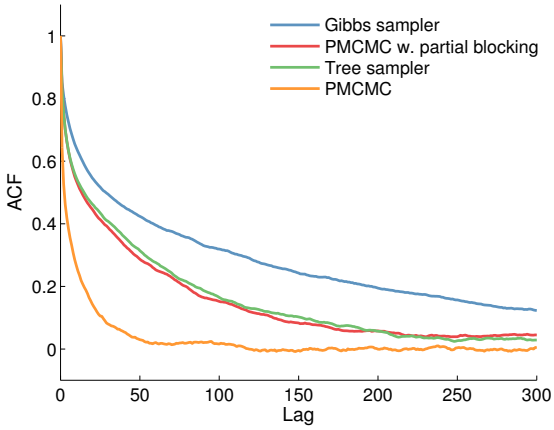


Figure 7: *The empirical ACF for Gibbs sampling, PMCMC, PMCMC with partial blocking, and tree sampling.*

sampler simulates all the latent variables jointly which reduces the autocorrelation, in particular when the latent variables are strongly dependent. However, it should be noted that this method also has the highest computational cost per iteration.

6 Conclusion

We have proposed a new framework for inference in PGMs using SMC and illustrated it on three examples. These examples show that it can be a viable alternative to standard methods used for inference and partition function estimation problems. An interesting avenue for future work is combining our proposed methods with AIS, to see if we can improve on both.

Acknowledgments

We would like to thank Iain Murray for his kind and very prompt help in providing the data for the LDA example. This work was supported by the projects: *Learning of complex dynamical systems* (Contract number: 637-2014-466) and *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524), both funded by the Swedish Research Council.

Bibliography

C. A. Naesseth, F. Lindsten, and T. B. Schön. smc-pgm, 2014. URL http://dx.doi.org/10.5281/zenodo.11947.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342, 2010.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

A. Bouchard-Côté, S. Sankararaman, and M. I. Jordan. Phylogenetic inference via sequential Monte Carlo. *Systematic Biology*, 61(4):579–593, 2012.

M. Briers, A. Doucet, and S. S. Singh. Sequential auxiliary particle belief propagation. In *Proceedings of the 8th International Conference on Information Fusion*, Philadelphia, PA, USA, 2005.

W. Buntine. Estimating likelihoods for topic models. In *Advances in Machine Learning*, pages 51–64. Springer, 2009.

P. Carbonetto and N. de Freitas. Conditional mean field. In *Advances in Neural Information Processing Systems (NIPS) 19*. MIT Press, 2007.

P. Del Moral. *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer, 2004.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 68(3):411–436, 2006.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, N. De Freitas, N. Gordon, et al. *Sequential Monte Carlo methods in practice*. Springer New York, 2001.

R. G. Everitt. Bayesian parameter estimation for latent Markov random fields and social networks. *Journal of Computational and Graphical Statistics*, 21(4): 940–960, 2012.

Paul Fearnhead and Peter Clifford. On-line inference for hidden markov models via particle filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(4):887–899, 2003.

A. Frank, P. Smyth, and A. T. Ihler. Particle-based variational inference for continuous systems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 826–834, 2009.

F. Hamze and N. de Freitas. From fields to trees. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence (UAI)*, Banff, Canada, July 2004.

F. Hamze and N. de Freitas. Hot coupling: a particle approach to inference and normalization on pairwise undirected graphs of arbitrary topology. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

A. T. Ihler and D. A. Mcallester. Particle belief propagation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, FL, USA, 2009.

M. Isard. PAMPAS: Real-valued graphical models for computer vision. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, USA, June 2003.

M. I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

J. M. Kosterlitz and D. J. Thouless. Ordering, metastability and phase transitions in two-dimensional systems. *J of Physics C: Solid State Physics*, 6(7): 1181, 1973.

F. Lindsten and T. B. Schön. Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends in Machine Learning*, 6(1):1–143, 2013.

F. Lindsten, M. I. Jordan, and T. B. Schön. Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research*, 15:2145–2184, june 2014.

C. A. Naesseth, F. Lindsten, and T. B. Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *Proceedings of the IEEE Information Theory Workshop (ITW)*, Hobart, Tasmania, Australia, November 2014.

R. M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2): 125–139, 2001.

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

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171:134–151, 2012.

C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer New York, 2004.

G. S. Scott and J. Baldridge. A recursive estimate for the predictive likelihood in a topic model. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1105–1112, Clearwater Beach, FL, USA, 2009.

E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, USA, 2003.

E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. *Communications of the ACM*, 53(10):95–103, 2010.

Y. Tomita and Y. Okabe. Probability-changing cluster algorithm for two-dimensional XY and clock models. *Physical Review B: Condensed Matter and Materials Physics*, 65:184405, 2002.

H. M Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1105–1112, 2009.

Paper D

Nested Sequential Monte Carlo Methods

Authors: Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön

Edited version of the paper:

Christian Naesseth, Fredrik Lindsten, and Thomas Schön. Nested sequential Monte Carlo methods. In *International Conference on Machine Learning (ICML)*, pages 1292–1301, 2015a.

This paper has been formatted to fit this layout. A correction has been added to Section 6, belief propagation (and not the Kalman filter) was used to compute the true solution for the Gaussian model.

Nested Sequential Monte Carlo Methods

Christian A. Naesseth*, Fredrik Lindsten†, and Thomas B. Schön†

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

†Dept. of Information Technology
Uppsala University
Uppsala, Sweden
{fredrik.lindsten,thomas.schon}@it.uu.se

Abstract

We propose *nested sequential Monte Carlo* (NSMC), a methodology to sample from sequences of probability distributions, even where the random variables are high-dimensional. NSMC generalises the SMC framework by requiring only approximate, *properly weighted*, samples from the SMC proposal distribution, while still resulting in a correct SMC algorithm. Furthermore, NSMC can in itself be used to produce such properly weighted samples. Consequently, one NSMC sampler can be used to construct an efficient high-dimensional proposal distribution for another NSMC sampler, and this *nesting* of the algorithm can be done to an arbitrary degree. This allows us to consider complex and high-dimensional models using SMC. We show results that motivate the efficacy of our approach on several filtering problems with dimensions in the order of 100 to 1 000.

1 Introduction

Inference in complex and high-dimensional statistical models is a very challenging problem that is ubiquitous in applications. Examples include, but are definitely not limited to, climate informatics (Monteleoni et al., 2013), bioinformatics (Cohen, 2004) and machine learning (Wainwright and Jordan, 2008). In particular, we are interested in *sequential* Bayesian inference, which involves computing integrals of the form

$$\bar{\pi}_k(f) := \mathbb{E}_{\bar{\pi}_k}[f(X_{1:k})] = \int f(x_{1:k})\bar{\pi}_k(x_{1:k})\mathrm{d}x_{1:k}, \tag{1}$$

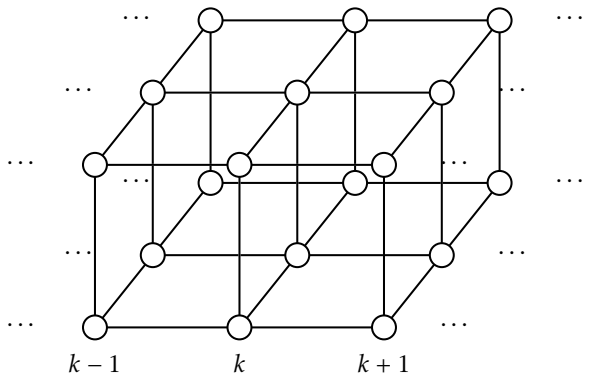


Figure 1: *Example of a spatio-temporal model where $\bar{\pi}_k(x_{1:k})$ is given by a $k \times 2 \times 3$ undirected graphical model and $x_k \in \mathbb{R}^{2\times 3}$.*

for some sequence of probability densities

$$\bar{\pi}_k(x_{1:k}) = Z_{\pi_k}^{-1}\pi_k(x_{1:k}), \qquad k \geq 1, \tag{2}$$

with normalisation constants $Z_{\pi_k} = \int \pi_k(x_{1:k})\mathrm{d}x_{1:k}$. Note that $x_{1:k} := (x_1, \ldots, x_k) \in \mathsf{X}_k$. The typical scenario that we consider is the well-known problem of inference in time series or state space models (Cappé et al., 2005; Shumway and Stoffer, 2011). Here the index k corresponds to time and we want to process some *observations* $y_{1:k}$ in a sequential manner to compute expectations with respect to the filtering distribution $\bar{\pi}_k(\mathrm{d}x_k) = \mathbb{P}(X_k \in \mathrm{d}x_k \mid y_{1:k})$. To be specific, we are interested in settings where

(i) X_k is high-dimensional, i.e. $X_k \in \mathbb{R}^d$ with $d \gg 1$, and

(ii) there are *local dependencies* among the latent variables $X_{1:k}$, both w.r.t. time k and between the individual components of the (high-dimensional) vectors X_k.

One example of the type of models we consider are the so-called spatio-temporal models (Cressie and Wikle, 2011; Rue and Held, 2005; Wikle, 2015). In Figure 1 we provide a probabilistic graphical model representation of a spatio-temporal model that we will explore further in Section 6.

Sequential Monte Carlo (SMC) methods, reviewed in Section 2.1, comprise one of the most successful methodologies for sequential Bayesian inference. However, SMC struggles in high-dimensions and these methods are rarely used for dimensions, say, $d \geq 10$ (Rebeschini and van Handel, 2015). The purpose of the NSMC methodology is to push this limit well beyond $d = 10$.

The basic strategy, described in Section 2.2, is to mimic the behaviour of a so-called *fully adapted* SMC algorithm. Full adaptation can drastically improve the efficiency of SMC in high dimensions. Unfortunately, it can rarely be implemented in practice since the fully adapted proposal distributions are typically

intractable. NSMC addresses this difficulty by requiring only approximate, *properly weighted*, samples from the proposal distribution. The proper weighting condition ensures the validity of NSMC, thus providing a generalisation of the family of SMC methods. Furthermore, NSMC will itself produce properly weighted samples. Consequently, it is possible to use one NSMC procedure within another to construct efficient high-dimensional proposal distributions. This *nesting* of the algorithm can be done to an arbitrary degree. For instance, for the model depicted in Figure 1 we could use three nested samplers, one for each dimension of the "volume".

The main methodological development is concentrated to Sections 3–4. We introduce the concept of proper weighting, approximations of the proposal distribution, and nesting of Monte Carlo algorithms. Throughout Section 3 we consider simple importance sampling and in Section 4 we extend the development to the sequential setting.

We deliberately defer the discussion of the existing body of related work until Section 5, to open up for a better understanding of the relationships to the new developments presented in Sections 3–4. We also discuss various attractive features of NSMC that are of interest in high-dimensional settings, e.g. the fact that it is easy to distribute the computation, which results in improved memory efficiency and lower communication costs. Finally, Section 6 profiles our method extensively with a state-of-the-art competing algorithm on several high-dimensional data sets. We also show the performance of inference and the modularity of the method on a $d = 1\,056$ dimensional climatological spatio-temporal model (Fu et al., 2012) structured according to Figure 1.

2 Background and Inference Strategy

2.1 Sequential Monte Carlo

Evaluating $\bar{\pi}_k(f)$ as well as the normalisation constant Z_{π_k} in (2) is typically intractable and we need to resort to approximations. SMC methods, or particle filters (PF), constitute a popular class of numerical approximations for sequential inference problems. Here we give a high-level introduction to the concepts underlying SMC methods, and postpone the details to Section 4. For a more extensive treatment we refer to Cappé et al. (2005); Doucet and Johansen (2011); Doucet et al. (2001). In particular, we will use the auxiliary SMC method as proposed by Pitt and Shephard (1999).

At iteration $k-1$, the SMC sampler approximates the target distribution $\bar{\pi}_{k-1}$ by a collection of weighted *particles* $\{(X^i_{1:k-1}, W^i_{k-1})\}_{i=1}^N$. These samples define an empirical point-mass approximation of the target distribution

$$\bar{\pi}^N_{k-1}(\mathrm{d}x_{1:k-1}) := \sum_{i=1}^{N} \frac{W^i_{k-1}}{\sum_\ell W^\ell_{k-1}} \delta_{X^i_{1:k-1}}(\mathrm{d}x_{1:k-1}), \tag{3}$$

where $\delta_X(\mathrm{d}x)$ denotes a Dirac measure at X. Each iteration of the SMC method can then conceptually be described by three steps, resampling, propagation, and weighting.

The resampling step puts emphasis on the most promising particles by discarding the unlikely ones and duplicating the likely ones. The propagation and weighting steps essentially correspond to using importance sampling when changing the target distribution from $\bar{\pi}_{k-1}$ to $\bar{\pi}_k$, i.e. simulating new particles from a *proposal distribution* and then computing corresponding importance weights.

2.2 Adapting the Proposal Distribution

The first working SMC algorithm was the bootstrap PF by Gordon et al. (1993), which propagates particles by sampling from the system dynamics and computes importance weights according to the observation likelihood (in the state space setting). However, it is well known that the bootstrap PF suffers from weight collapse in high-dimensional settings (Bickel et al., 2008), i.e. the estimate is dominated by a single particle with weight close to one. This is an effect of the mismatch between the importance sampling proposal and the target distribution, which typically gets more pronounced in high dimensions.

More efficient proposals, partially alleviating the degeneracy issue for some models, can be designed by *adapting* the proposal distribution to the target distribution (see Section 4.2). In Naesseth et al. (2014a) we make use of the *fully adapted* SMC method (Pitt and Shephard, 1999) for doing inference in a (fairly) high-dimensional *discrete* model where x_k is a 60-dimensional discrete vector. We can then make use of forward filtering and backward simulation, operating on the individual *components* of each x_k, in order to sample from the fully adapted SMC proposals. However, this method is limited to models where the latent space is either discrete or Gaussian and the optimal proposal can be identified with a tree-structured graphical model. Our development here can be seen as a non-trivial extension of this technique. Instead of coupling one SMC sampler with an *exact* forward filter/backward simulator (which in fact reduces to an instance of standard SMC), we derive a way of coupling multiple SMC samplers and SMC-based backward simulators. This allows us to construct procedures for mimicking the efficient fully adapted proposals for arbitrary latent spaces and structures in high-dimensional models.

3 Proper Weighting and Nested Importance Sampling

In this section we will lay the groundwork for the derivation of the class of NSMC algorithms. We start by considering the simpler case of importance sampling (IS), which is a fundamental component of SMC, and introduce the key concepts that we make use of. In particular, we will use a (slightly nonstandard) presentation

of an algorithm as an instance of a *class*, in the object-oriented sense, and show that these classes can be nested to an arbitrary degree.

3.1 Exact Approximation of the Proposal Distribution

Let $\bar{\pi}(x) = Z_\pi^{-1}\pi(x)$ be a target distribution of interest. IS can be used to estimate an expectation $\bar{\pi}(f) := \mathbb{E}_{\bar{\pi}}[f(X)]$ by sampling from a proposal distribution $\bar{q}(x) = Z_q^{-1}q(x)$ and computing the estimator $(\sum_{i=1}^N W^i)^{-1}\sum_{i=1}^N W^i f(X^i)$, with $W^i = \frac{Z_q\pi(X^i)}{q(X^i)}$, and where $\{(X^i, W^i)\}_{i=1}^N$ are the weighted samples. It is possible to replace the IS weight by a nonnegative unbiased estimate, and still obtain a valid (consistent, etc.) algorithm (Liu, 2001, p. 37). One way to motivate this approach is by considering the random weight to be an auxiliary variable and to extend the target distribution accordingly. Our development is in the same flavour, but we will use a more explicit condition on the relationship between the random weights and the simulated particles. Specifically, we will make use of the following key property to formally justify the proposed algorithms.

Definition 1 (Properly weighted sample). A (random) pair (X, W) is properly weighted for an *unnormalised* distribution p if $W \geq 0$ and $\mathbb{E}[f(X)W] = p(f) := \int f(x)p(x)\mathrm{d}x$ for all measurable functions f.

Note that proper weighting of $\{(X^i, W^i)\}_{i=1}^N$ implies unbiasedness of the estimate of the normalising constant of p. Indeed, taking $f(x) \equiv 1$ gives $\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^N W^i\right] = \int p(x)\mathrm{d}x =: Z_p$.

Interestingly, to construct a valid IS algorithm for our target $\bar{\pi}$ it is sufficient to generate samples that are properly weighted *w.r.t. the proposal* distribution q. To formalise this claim, assume that we are not able to simulate exactly from $\bar{q}$, but that it is possible to evaluate the unnormalised density q point-wise. Furthermore, assume we have access to a class Q, which works as follows. The constructor of Q requires the specification of an *unnormalised density function*, say, q, which will be approximated by the procedures of Q. Furthermore, to highlight the fact that we will typically use IS (and SMC) to construct Q, the constructor also takes as an argument a precision parameter M, corresponding to the number of samples used by the "internal" Monte Carlo procedure. An object is then instantiated as q = Q(q, M). The class Q is assumed to have the following properties:

(A1) Let q = Q(q, M). Assume that:

1. The construction of q results in the generation of a (possibly random) member variable, accessible as $\widehat{Z}_q$ = q.GetZ(). The variable $\widehat{Z}_q$ is a nonnegative, unbiased estimate of the normalising constant $Z_q = \int q(x)dx$.
2. Q has a member function Simulate which returns a (possibly random) variable X = q.Simulate(), such that $(X, \widehat{Z}_q)$ is properly weighted for q.

With the definition of Q in place, it is possible to generalise[1] the basic importance sampler as in Algorithm 1, which generates weighted samples $\{(X^i, W^i)\}_{i=1}^N$ targeting $\bar{\pi}$. Note that Algorithm 1 is different from a random weight IS, since it approximates the proposal distribution (and not just the importance weights).

Algorithm 1: Nested IS (steps 1–3 for $i = 1, \ldots, N$)

1. Initialise $\mathsf{q}^i = \mathsf{Q}(q, M)$.
2. Set $\widehat{Z}_q^i = \mathsf{q}^i.\mathsf{GetZ}()$ and $X^i = \mathsf{q}^i.\mathsf{Simulate}()$.
3. Set $W^i = \dfrac{\widehat{Z}_q^i \pi(X^i)}{q(X^i)}$.
4. Compute $\widehat{Z}_\pi = \frac{1}{N} \sum_{i=1}^N W^i$.

To see the validity of Algorithm 1 we can interpret the sampler as a standard IS algorithm for an extended target distribution, defined as $\bar{\Pi}(x, u) := u\, \bar{Q}(x, u)\bar{\pi}(x)q^{-1}(x)$, where $\bar{Q}(x, u)$ is the joint PDF of the random pair $(\mathsf{q.Simulate}(), \mathsf{q.GetZ}())$. Note that $\bar{\Pi}$ is indeed a PDF that admits $\bar{\pi}$ as a marginal; for any measurable subset $A \subseteq \mathsf{X}$,

$$\begin{aligned}\bar{\Pi}(A \times \mathbb{R}_+) &= \int \mathbf{1}_A(x) \frac{u\, \bar{\pi}(x)}{q(x)} \bar{Q}(x, u) \mathrm{d}x \mathrm{d}u \\ &= \mathbb{E}\left[\widehat{Z}_q \frac{\mathbf{1}_A(X)\bar{\pi}(X)}{q(X)}\right] = \bar{q}\left(\mathbf{1}_A \frac{\bar{\pi}}{q}\right) Z_q = \bar{\pi}(A),\end{aligned}$$

where the penultimate equality follows from the fact that $(X, \widehat{Z}_q)$ is properly weighted for q. Furthermore, the standard unnormalised IS weight for a sampler with target $\bar{\Pi}$ and proposal $\bar{Q}$ is given by $u\, \pi/q$, in agreement with Algorithm 1.

Algorithm 1 is an example of what is referred to as an *exact approximation*; see e.g., Andrieu and Roberts (2009); Andrieu et al. (2010). Algorithmically, the method appears to be an approximation of an IS, but samples generated by the algorithm nevertheless target the correct distribution $\bar{\pi}$.

3.2 Modularity of Nested IS

To be able to implement Algorithm 1 we need to define a class Q with the required properties (A1). The modularity of the procedure (as well as its name) comes from the fact that we can use Algorithm 1 also in this respect. Indeed, let us now view $\bar{\pi}$—the target distribution of Algorithm 1—as the *proposal distribution* for another Nested IS procedure and consider the following definition of Q:

[1] With $\mathsf{q.GetZ}() \mapsto Z$ and $\mathsf{q.Simulate}()$ returning a sample from $\bar{q}$ we obtain the standard IS method.

1. Algorithm 1 is executed at the construction of the object $\mathsf{p} = \mathsf{Q}(\pi, N)$, and $\mathsf{p.GetZ()}$ returns the normalising constant estimate $\widehat{Z}_\pi$.
2. $\mathsf{p.Simulate()}$ simulates a categorical random variable B with $\mathbb{P}(B = i) = W^i / \sum_{\ell=1}^N W^\ell$ and returns X^B.

A simple computation now yields that for any measurable f we have $\mathbb{E}[f(X^B)\widehat{Z}_\pi] = \bar{\pi}(f)Z_\pi$. This implies that $(X^B, \widehat{Z}_\pi)$ is properly weighted for π and that our definition of $\mathsf{Q}(\pi, N)$ indeed satisfies condition (A1).

The Nested IS algorithm in itself is unlikely to be of direct practical interest. However, in the next section we will, essentially, repeat the preceding derivation in the context of SMC to develop the NSMC method.

4 Nested Sequential Monte Carlo

4.1 Fully Adapted SMC Samplers

Let us return to the sequential inference problem. As before, let $\bar{\pi}_k(x_{1:k}) = Z_{\pi_k}^{-1}\pi_k(x_{1:k})$ denote the target distribution at "time" k. The unnormalised density π_k can be evaluated point-wise, but the normalising constant Z_{π_k} is typically unknown. We will use SMC to simulate sequentially from the distributions $\{\bar{\pi}_k\}_{k=1}^n$. In particular, we consider the fully adapted SMC sampler (Pitt and Shephard, 1999), which corresponds to a specific choice of resampling weights and proposal distribution, chosen in such a way that the importance weights are all equal to $1/N$. Specifically, the proposal distribution (often referred to as the *optimal proposal*) is given by $\bar{q}_k(x_k \mid x_{1:k-1}) = Z_{q_k}(x_{1:k-1})^{-1} q_k(x_k \mid x_{1:k-1})$, where

$$q_k(x_k \mid x_{1:k-1}) := \pi_k(x_{1:k})/\pi_{k-1}(x_{1:k-1}).$$

In addition, the normalising "constant" $Z_{q_k}(x_{1:k-1}) = \int q_k(x_k \mid x_{1:k-1})\mathrm{d}x_k$ is further used to define the *resampling weights*, i.e. the particles at time $k-1$ are resampled according to $Z_{q_k}(x_{1:k-1})$ before they are propagated to time k. For notational simplicity, we use the convention $x_{1:0} = \emptyset$, $q_1(x_1 \mid x_{1:0}) = \pi_1(x_1)$ and $Z_{q_1}(x_{1:0}) = Z_{\pi_1}$. The fully adapted auxiliary SMC sampler is given in Algorithm 2.

As mentioned above, at each iteration $k = 1, \ldots, n$, the method produces *unweighted* samples $\{X_k^i\}_{i=1}^N$ approximating $\bar{\pi}_k$. It also produces an unbiased estimate $\widehat{Z}_{\pi_k}$ of Z_{π_k} (Del Moral, 2004, Proposition 7.4.1). The algorithm is expressed in a slightly non-standard form; at iteration k we loop over the ancestor particles, i.e. the particles after resampling at iteration $k-1$, and let each ancestor particle j generate m_k^j offsprings. (The variable L is just for bookkeeping.) This is done to clarify the connection with the NSMC procedure below. Furthermore, we have included a (completely superfluous) resampling step at iteration $k = 1$, where the "dummy variables" $\{X_{1:0}^i\}_{i=1}^N$ are resampled according to the (all equal) weights

Algorithm 2: SMC (fully adapted)

1. Set $\widehat{Z}_{\pi_0} = 1$.
2. **for** $k = 1$ **to** n
   (a) Compute $\widehat{Z}_{\pi_k} = \widehat{Z}_{\pi_{k-1}} \times \frac{1}{N} \sum_{j=1}^{N} Z_{q_k}(X_{1:k-1}^j)$.
   (b) Draw $m_k^{1:N}$ from a multinomial distribution with probabilities $\frac{Z_{q_k}(X_{1:k-1}^j)}{\sum_{\ell=1}^{N} Z_{q_k}(X_{1:k-1}^\ell)}$, for $j = 1, \ldots, N$.
   (c) Set $L \leftarrow 0$
   (d) **for** $j = 1$ **to** N
      i. Draw $X_k^i \sim \bar{q}_k(\cdot \mid X_{1:k-1}^j)$ and let $X_{1:k}^i = (X_{1:k-1}^j, X_k^i)$ for $i = L+1, \ldots, L+m_k^j$.
      ii. Set $L \leftarrow L + m_k^j$.

$\{Z_{q_1}(X_{1:0}^i)\}_{i=1}^N = \{Z_{\pi_1}\}_{i=1}^N$. The analogue of this step is, however, used in the NSMC algorithm, where the initial normalising constant Z_{π_1} is *estimated*. We thus have to resample the corresponding initial particle systems accordingly.

4.2 Fully Adapted Nested SMC Samplers

In analogue with Section 3, assume now that we are not able to simulate exactly from $\bar{q}_k$, nor compute Z_{q_k}. Instead, we have access to a class Q which satisfies condition (A1). The proposed NSMC method is then given by Algorithm 3.

Algorithm 3: Nested SMC (fully adapted)

1. Set $\widehat{Z}_{\pi_0} = 1$.
2. **for** $k = 1$ **to** n
   (a) Initialise $\mathsf{q}^j = \mathsf{Q}(q_k(\cdot \mid X_{1:k-1}^j), M)$ for $j = 1, \ldots, N$.
   (b) Set $\widehat{Z}_{q_k}^j = \mathsf{q}^j.\mathsf{GetZ}()$ for $j = 1, \ldots, N$.
   (c) Compute $\widehat{Z}_{\pi_k} = \widehat{Z}_{\pi_{k-1}} \times \left\{ \frac{1}{N} \sum_{j=1}^{N} \widehat{Z}_{q_k}^j \right\}$.
   (d) Draw $m_k^{1:N}$ from a multinomial distribution with probabilities $\frac{\widehat{Z}_{q_k}^j}{\sum_{\ell=1}^{N} \widehat{Z}_{q_k}^\ell}$ for $j = 1, \ldots, N$.
   (e) Set $L \leftarrow 0$
   (f) **for** $j = 1$ **to** N
      i. Compute $X_k^i = \mathsf{q}^j.\mathsf{Simulate}()$ and let $X_{1:k}^i = (X_{1:k-1}^j, X_k^i)$ for $i = L+1, \ldots, L+m_k^j$.
      ii. **delete** q^j.
      iii. Set $L \leftarrow L + m_k^j$.

Algorithm 3 can be seen as an *exact approximation* of the fully adapted SMC sampler in Algorithm 2. (In Naesseth et al. (2015) we provide a formulation of NSMC with arbitrary proposals and resampling weights.) We replace the exact computation of Z_{q_k} and exact simulation from $\bar{q}_k$, by the approximate procedures available through Q. Despite this approximation, however, Algorithm 3 is a valid SMC method. This is formalised by the following theorem.

Theorem 1. *Assume that* Q *satisfies condition (A1). Then, under certain regularity conditions on the function* $f : \mathsf{X}_k \mapsto \mathbb{R}^d$ *and for an asymptotic variance* $\Sigma_k^M(f)$*, both specified in Naesseth et al. (2015), we have*

$$N^{1/2}\left(\frac{1}{N}\sum_{i=1}^{N} f(X_{1:k}^i) - \bar{\pi}_k(f)\right) \xrightarrow{D} \mathcal{N}(0, \Sigma_k^M(f)),$$

where $\{X_{1:k}^i\}_{i=1}^M$ *are generated by Algorithm 3 and* $\xrightarrow{D}$ *denotes convergence in distribution.*

Proof: See Naesseth et al. (2015). □

Remark 1. The key point with Theorem 1 is that, under certain regularity conditions, the NSMC method converges at rate $\sqrt{N}$ even for a fixed (and finite) value of the precision parameter M. The asymptotic variance $\Sigma_k^M(f)$, however, will depend on the accuracy and properties of the approximative procedures of Q. We leave it as future work to establish more informative results, relating the asymptotic variance of NSMC to that of the ideal, fully adapted SMC sampler.

4.3 Backward Simulation and Modularity of NSMC

As previously mentioned, the NSMC procedure is modular in the sense that we can make use of Algorithm 3 also to define the class Q. Thus, we now view $\bar{\pi}_n$ as the *proposal distribution* that we wish to approximately sample from using NSMC. Algorithm 3 directly generates an estimate $\widehat{Z}_{\pi_n}$ of the normalising constant of π_n (which indeed is unbiased, see Theorem 2). However, we also need to generate a sample $\widetilde{X}_{1:n}$ such that $(\widetilde{X}_{1:n}, \widehat{Z}_{\pi_n})$ is properly weighted for π_n.

The simplest approach, akin to the Nested IS procedure described in Section 3.2, is to draw B_n uniformly on $\{1, \ldots, N\}$ and return $\widetilde{X}_{1:n} = X_{1:n}^{B_n}$. This will indeed result in a valid definition of the Simulate procedure. However, this approach will suffer from the well known path degeneracy of SMC samplers. In particular, since we call q^j.Simulate() multiple times in Step 2(f)i of Algorithm 3, we risk to obtain (very) strongly correlated samples by this simple approach.

It is possible to improve the performance of the above procedure by instead making use of a *backward simulator* (Godsill et al., 2004; Lindsten and Schön, 2013) to simulate $\widetilde{X}_{1:n}$. The backward simulator, given in Algorithm 4, is a type of smoothing algorithm; it makes use of the particles generated by a forward pass

of Algorithm 3 to simulate backward in "time" a trajectory $\widetilde{X}_{1:n}$ approximately distributed according to $\bar{\pi}_n$.

Algorithm 4: Backward simulator (fully adapted)

1. Draw B_n uniformly on $\{1, \ldots, N\}$.
2. Set $\widetilde{X}_n = X_n^{B_n}$.
3. **for** $k = n - 1$ **to** 1
   (a) Compute $\widetilde{W}_k^j = \dfrac{\pi_n((X_{1:k}^j, \widetilde{X}_{k+1:n}))}{\pi_k(X_{1:k}^j)}$ for $j = 1, \ldots, N$.
   (b) Draw B_k from a categorical distribution with probabilities $\dfrac{\widetilde{W}_k^j}{\sum_{\ell=1}^N \widetilde{W}_k^\ell}$ for $j = 1, \ldots, N$.
   (c) Set $\widetilde{X}_{k:n} = (X_k^{B_k}, \widetilde{X}_{k+1:n})$.

Remark 2. Algorithm 4 assumes unweighted particles and can thus be used in conjunction with the fully adapted NSMC procedure of Algorithm 2. If, however, the forward filter is not fully adapted the weights need to be accounted for in the backward simulation; see Naesseth et al. (2015).

The modularity of NSMC is established by the following result.

Definition 2. Let $\mathsf{p} = \mathsf{Q}(\pi_n, N)$ be defined as follows:

1. The constructor executes Algorithm 3 with target distribution π_n and with N particles, and $\mathsf{p.GetZ}()$ returns the estimate of the normalising constant $\widehat{Z}_{\pi_n}$.
2. $\mathsf{p.Simulate}()$ executes Algorithm 4 and returns $\widetilde{X}_{1:n}$.

Theorem 2. *The class* Q *defined as in Definition 2 satisfies condition (A1).*

Proof: See Naesseth et al. (2015). □

A direct, and important, consequence of Theorem 2 is that NSMC can be used as a component of powerful learning algorithms, such as the particle Markov chain Monte Carlo (PMCMC) method (Andrieu et al., 2010) and many of the other methods discussed in Section 5. Since standard SMC is a special case of NSMC, Theorem 2 implies proper weighting also of SMC.

5 Practicalities and Related Work

There has been much recent interest in using SMC within SMC in various ways. The SMC^2 by Chopin et al. (2013) and the recent method by Crisan and Míguez (2013) are sequential learning algorithms for state space models, where one SMC

sampler for the parameters is coupled with another SMC sampler for the latent states. Johansen et al. (2012) and Chen et al. (2011) address the state inference problem by splitting the state variable into different components and run coupled SMC samplers for these components. These methods differ substantially from NSMC; they solve different problems and the "internal" SMC sampler(s) is constructed in a different way (for approximate marginalisation instead of for approximate simulation). Another related method is the random weights PF of Fearnhead et al. (2010a), requiring exact samples from $\bar{q}$ and where the importance weights are estimated using a nested Monte Carlo algorithm.

The method most closely related to NSMC is the space-time particle filter (ST-PF) (Beskos et al., 2014a), which has been developed independently and in parallel with our work. The ST-PF is also designed for solving inference problems in high-dimensional models. It can be seen as a island PF (Vergé et al., 2013) implementation of the method presented by Naesseth et al. (2014b). Specifically, for a spatio-temporal models they run an island PF over both spatial and temporal dimensions. However, the ST-PF does not generate an approximation of the fully adapted SMC sampler.

Another key distinction between NSMC and ST-PF is that in the latter each particle in the "outer" SMC sampler comprises a complete particle system from the "inner" SMC sampler. For NSMC, on the other hand, the particles will simply correspond to different hypotheses about the latent variables (as in standard SMC), regardless of how many samplers that are nested. This is a key feature of NSMC, since it implies that it is easily distributed over the particles. The main computational effort of Algorithm 3 is the construction of $\{\mathrm{q}^j\}_{j=1}^N$ and the calls to the Simulate procedure, which can be done independently for each particle. This leads to improved memory efficiency and lower communication costs. Furthermore, we have found (see Section 6) that NSMC can outperform ST-PF even when run on a single machine with matched computational costs.

Another strength of NSMC methods are their relative ease of implementation, which we show in Section 6.3. We use the framework to sample from what is essentially a cubic grid Markov random field (MRF) model just by implementing three nested samplers, each with a target distribution defined on a simple chain.

There are also other SMC-based methods designed for high-dimensional problems, e.g., the block PF studied by Rebeschini and van Handel (2015), the location particle smoother by Briggs et al. (2013) and the PF-based methods reviewed in Djuric and Bugallo (2013). However, these methods are all inconsistent, as they are based on various approximations that result in systematic errors.

The previously mentioned PMCMC (Andrieu et al., 2010) is a related method, where SMC is used as a component of an MCMC algorithm. We make use of a very similar extended space approach to motivate the validity of our algorithm. Note that our proposed algorithm can be used as a component in PMCMC and most of the other algorithms mentioned above, which further increases the scope

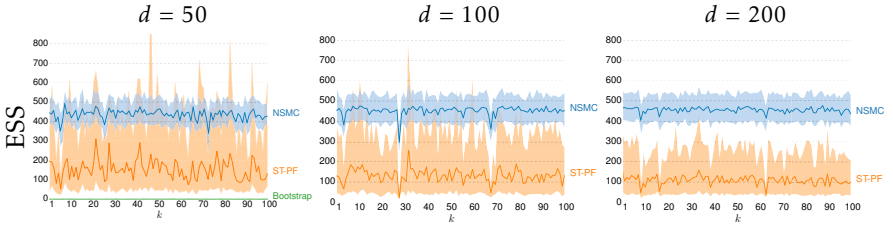


Figure 2: *Median (over dimension) ESS* (4) *and* 15–85% *percentiles (shaded region). The results are based on* 100 *independent runs for the Gaussian MRF with dimension* d.

of models it can handle.

6 Experimental Results

We illustrate NSMC on three high-dimensional examples, both with real and synthetic data. We compare NSMC with standard (bootstrap) PF and the ST-PF of Beskos et al. (2014a) with equal computational budgets on a single machine (i.e., neglecting the fact that NSMC is more easily distributed). These methods are, to the best of our knowledge, the only other available *consistent* online methods for full Bayesian inference in general sequential models. For more detailed explanations of the models and additional results, see Naesseth et al. (2015)[2].

6.1 Gaussian State Space Model

We start by considering a high-dimensional Gaussian state space model, where we have access to the true solution from belief propagation. The latent variables and measurements $\{X_{1:k}, Y_{1:k}\}$, with $\{X_k, Y_k\} = \{X_{k,l}, Y_{k,l}\}_{l=1}^{d}$, are modeled by a $d \times k$ lattice Gaussian MRF. The true data is simulated from a nearly identical state space model (see Naesseth et al. (2015)). We run a 2-level NSMC sampler. The outer level is fully adapted, i.e. the proposal distribution is $q_k = p(x_k \mid x_{k-1}, y_k)$, which thus constitute the target distribution for the inner level. To generate properly weighted samples from q_k, we use a bootstrap PF operating on the d components of the vector x_k. Note that we only use bootstrap proposals where the actual sampling takes place, and that the conditional distribution $p(x_k \mid x_{k-1}, y_k)$ is not explicitly used.

We simulate data from this model for $k = 1, \ldots, 100$ for different values of $d = \dim(x_k) \in \{50, 100, 200\}$. The exact filtering marginals are computed using belief propagation. We compare with both the ST-PF and standard (bootstrap) PF.

[2]Code available at https://github.com/can-cs/nestedsmc

The results are evaluated based on the effective sample size (ESS, see e.g. Fearnhead et al. (2010b)) defined as,

$$\mathrm{ESS}(x_{k,l}) = \left(\mathbb{E}\left[\frac{(\widehat{x}_{k,l} - \mu_{k,l})^2}{\sigma_{k,l}^2} \right] \right)^{-1}, \tag{4}$$

where $\widehat{x}_{k,l}$ denote the mean estimates and $\mu_{k,l}$ and $\sigma_{k,l}^2$ denote the true mean and variance of $x_{k,l} \mid y_{1:k}$ obtained from belief propagation. The expectation in (4) is approximated by averaging over 100 independent runs of the involved algorithms. The ESS reflects the estimator accuracy, obvious by the definition which is tightly related to the mean-squared-error. Intuitively the ESS corresponds to the equivalent number of i.i.d. samples needed for the same accuracy.

We use $N = 500$ and $M = 2 \cdot d$ for NSMC and match the computational time for ST-PF and bootstrap PF. We report the results in Figure 2. Note that the bootstrap PF is omitted from $d = 100, 200$ due to its poor performance already for $d = 50$ (which is to be expected). Each dimension $l = 1, \ldots, d$ provides us with a value of the ESS, so we present the median (lines) and 15–85% percentiles (shaded regions) in the first row of Figure 2.

We have conducted additional experiments with different model parameters and different choices for N and M (some additional results are given in Naesseth et al. (2015)). Overall the results seem to be in agreement with the ones presented here, however ST-PF seems to be more robust to the trade-off between N and M. A rule-of-thumb for NSMC is to generally try to keep N as high as possible, while still maintaining a reasonable estimate of Z_{q_k}.

6.2 Non-Gaussian State Space Model

Next, we consider an example with a non-Gaussian SSM, borrowed from Beskos et al. (2014a) where the full details of the model are given. The transition probability $p(x_k \mid x_{k-1})$ is a localised Gaussian mixture and the measurement probability $p(y_k \mid x_k)$ is t-distributed. The model dimension is $d = 1\,024$. Beskos et al. (2014a) report improvements for ST-PF over both the bootstrap PF and the block PF by Rebeschini and van Handel (2015). We use $N = M = 100$ for both ST-PF and NSMC (the special structure of this model implies that there is no significant computational overhead from running backward sampling) and the bootstrap PF is given $N = 10\,000$. In Figure 3 we report the ESS (4), estimated according to Carpenter et al. (1999). The ESS for the bootstrap PF is close to 0, for ST-PF around 1–2, and for NSMC slightly higher at 7–8. However, we note that all methods perform quite poorly on this model, and to obtain satisfactory results it would be necessary to use more particles.

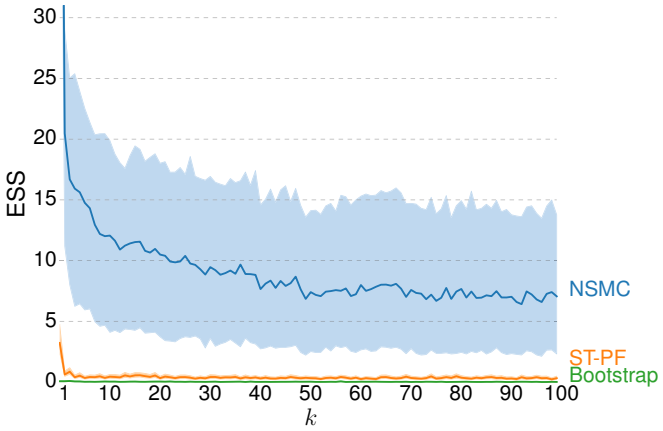


Figure 3: *Median ESS with* 15 – 85% *percentiles (shaded region) for the non-Gaussian SSM.*

6.3 Spatio-Temporal Model – Drought Detection

In this final example we study the problem of detecting droughts based on measured precipitation data (Jones and Harris, 2013) for different locations on earth. We look at the situation in North America during the years 1901–1950 and the Sahel region in Africa during the years 1950–2000, time frames including the so-called Dust Bowl in the US during the 1930s (Schubert et al., 2004) and the decades long drought in the Sahel region in Africa starting in the 1960s (Foley et al., 2003; Hoerling et al., 2006). We consider the spatio-temporal model defined by Fu et al. (2012) and compare with the results therein. Each location in a region is modelled to be in either a *normal* state 0 or in an *abnormal* state 1 (drought). Measurements are given by precipitation (in millimeters) for each loca-

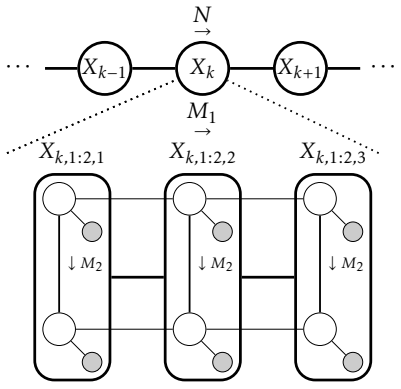


Figure 4: *Illustration of the three-level NSMC.*

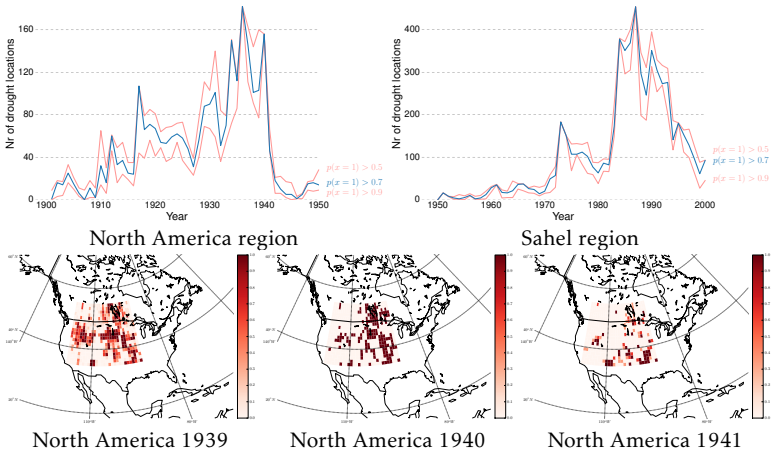


Figure 5: Top: *Number of locations with estimated* $p(x = 1) > \{0.5, 0.7, 0.9\}$ *for the two regions.* Bottom: *Estimate of* $p(x_{t,i} = 1)$ *for all sites over a span of 3 years. All results for* $N = 100, N_1 = \{30, 40\}, N_2 = 20$.

tion and year. At every time instance k our latent structure is described by a rectangular 2D grid $X_k = \{X_{k,i,j}\}_{i=1,j=1}^{I,J}$; in essence this is the model showcased in Figure 1. Fu et al. (2012) considers the problem of finding the maximum aposteriori configuration, using a linear programming relaxation. We will instead compute an approximation of the full posterior filtering distribution $\bar{\pi}_k(x_k) = p(x_k \mid y_{1:k})$. The rectangular structure is used to instantiate an NSMC method that on the first level targets the full posterior filtering distribution, second level the columns, and third level the rows of X_k. Properly weighted samples are generated using a bootstrap PF for the third level. The structure of our NSMC method applied to this particular problem is illustrated in Figure 4.

Figure 5 gives the results on the parts of North America that we consider. The first row shows the number of locations where the estimate of $p(x_{k,i,j} = 1)$ exceeds $\{0.5, 0.7, 0.9\}$, for both regions. These results seems to be in agreement with Fu et al. (2012, Figures 3, 6). However, we also receive an approximation of the full posterior and can visualise uncertainty in our estimates, as illustrated by the three different levels of posterior probability for drought. In general, we obtain a rich sample diversity from the posterior distribution. However, for some problematic years the sampler degenerates, with the result that the three credibility levels all coincide. This is also visible in the second row of Figure 5, where we show the posterior estimates $p(x_{k,i,j} \mid y_{1:k})$ for the years 1939–1941, overlayed on the regions of interest. Naturally, one way to improve the estimates is to run the sampler with a larger number of particles, which has been kept very low in this proof-of-concept.

We have shown that a straightforward NSMC implementation with fairly few

particles can attain reasonable approximations to the filtering problem for dimensions in the order of hundreds, or even thousands. This means that NSMC methods takes the SMC framework an important step closer to being viable for high-dimensional statistical inference problems. However, NSMC is not a silver bullet for solving high-dimensional inference problems, and the approximation accuracy will be highly model dependent. Hence, much work remains to be done, for instance on combining NSMC with other techniques for high-dimensional inference such as localisation (Rebeschini and van Handel, 2015) and annealing (Beskos et al., 2014b), in order to solve even more challenging problems.

Acknowledgments

This work was supported by the projects: *Learning of complex dynamical systems* (Contract number: 637-2014-466) and *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524), both funded by the Swedish Research Council.

Bibliography

C. Andrieu and G. O. Roberts. 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.

A. Beskos, D. Crisan, A. Jasra, K. Kamatani, and Y. Zhou. A stable particle filter in high-dimensions. *ArXiv:1412.3501*, December 2014a.

Alexandros Beskos, Dan Crisan, and Ajay Jasra. On the stability of sequential Monte Carlo methods in high dimensions. *Ann. Appl. Probab.*, 24(4):1396–1445, 08 2014b.

Peter Bickel, Bo Li, and Thomas Bengtsson. *Sharp failure rates for the bootstrap particle filter in high dimensions*, volume Volume 3 of *Collections*, pages 318–329. Institute of Mathematical Statistics, Beachwood, Ohio, USA, 2008.

Jonathan Briggs, Michael Dowd, and Renate Meyer. Data assimilation for large-scale spatio-temporal systems using a location particle smoother. *Environmetrics*, 24(2):81–97, 2013.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387402640.

J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings Radar, Sonar and Navigation*, 146(1):2–7, 1999.

Tianshi Chen, Thomas B. Schön, Henrik Ohlsson, and Lennart Ljung. Decentralized particle filter with arbitrary state decomposition. *IEEE Transactions on Signal Processing*, 59(2):465–478, Feb 2011.

N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. SMC2: 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.

Jacques Cohen. Bioinformatics—an introduction for computer scientists. *ACM Computing Surveys (CSUR)*, 36(2):122–158, 2004.

N. Cressie and C. K. Wikle. *Statistics for spatio-temporal data*. Wiley, 2011.

D. Crisan and J. Míguez. Nested particle filters for online parameter estimation in discrete-time state-space Markov models. *ArXiv:1308.1883*, August 2013.

P. Del Moral. *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer, 2004.

Petar M Djuric and Mónica F Bugallo. Particle filtering for high-dimensional systems. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, pages 352–355. IEEE, 2013.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *Nonlinear Filtering Handbook*. Oxford University Press, 2011.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. *An introduction to sequential Monte Carlo methods*. Springer, 2001.

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, 2010a.

Paul Fearnhead, David Wyncoll, and Jonathan Tawn. A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464, 2010b.

J. A. Foley, M. T. Coe, M. Scheffer, and G. Wang. Regime shifts in the sahara and sahel: Interactions between ecological and climatic systems in northern africa. *Ecosystems*, 6:524–539, 2003.

Qiang Fu, Arindam Banerjee, Stefan Liess, and Peter K. Snyder. Drought detection of the last century: An MRF-based approach. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 24–34, Anaheim, CA, USA, April 2012.

S. J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

N. J. Gordon, D. J. Salmond, and A. 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.

M. Hoerling, J. Hurrell, J. Eischeid, and A. Phillips. Detection and attribution of twentieth-century northern and southern african rainfall change. *Journal of Climate*, 19:3989––4008, 2006.

A. M. Johansen, N. Whiteley, and A. Doucet. Exact approximation of Rao-Blackwellised particle filters. In *Proceesings of the 16th IFAC Symposium on System Identification (SYSID)*, pages 488–493, Brussels, Belgium, 2012.

P.D. Jones and I. Harris. CRU TS3.21: Climatic research unit (CRU) time-series (ts) version 3.21 of high resolution gridded data of month-by-month variation in climate (jan. 1901- dec. 2012). NCAS British Atmospheric Data Centre, sep 2013. URL `http://dx.doi.org/10.5285/D0E1585D-3417-485F-87AE-4FCECF10A992`.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, 1960.

F. Lindsten and T. B. Schön. Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends in Machine Learning*, 6(1):1–143, 2013.

Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2001.

Claire Monteleoni, Gavin A. Schmidt, Francis Alexander, Alexandru Niculescu-Mizil, Karsten Steinhaeuser, Michael Tippett, Arindam Banerjee, M. Benno Blumenthal, Jason E. Smerdon Auroop R. Ganguly, and Marco Tedesco. Climate informatics. In Ting Yu, Nitesh Chawla, and Simeon Simoff, editors, *Computational Intelligent Data Analysis for Sustainable Development*. Chapman and Hall/CRC, London, 2013.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *The 2014 IEEE Information Theory Workshop (ITW)*, pages 431–435, Nov 2014a.

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., 2014b.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Nested sequential Monte Carlo methods. *arXiv:1502.02536*, 2015.

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

P. Rebeschini and R. van Handel. Can local particle filters beat the curse of dimensionality? *Ann. Appl. Probab. (to appear)*, 2015.

H. Rue and L. Held. *Gaussian Markov Random Fields, Theory and Applications*. CDC Press, Boca Raton, FL, USA, 2005.

S. D. Schubert, M. J. Suarez, P. J. Pegion, R. D. Koster, and J. T. Bacmeister. On the cause of the 1930s dust bowl. *Science*, 303:1855–1859, 2004.

R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications – with R examples*. Springer Texts in Statistics. Springer, New York, USA, third edition, 2011.

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*, pages 1–18, 2013.

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

C. K. Wikle. Modern perspectives on statistics for spatio-temporal data. *WIREs Computational Statistics*, 7(1):86–98, 2015.

Paper E

High-dimensional Filtering using Nested Sequential Monte Carlo

Authors: Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön

Edited version of the paper:

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. High-dimensional filtering using nested sequential Monte Carlo. *arXiv:1612.09162*, 2016.

This paper has been formatted to fit this layout.

High-dimensional Filtering using Nested Sequential Monte Carlo

Christian A. Naesseth*, Fredrik Lindsten†, and Thomas B. Schön†

*Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

†Dept. of Information Technology
Uppsala University
Uppsala, Sweden
{fredrik.lindsten,thomas.schon}@it.uu.se

Abstract

Sequential Monte Carlo (SMC) methods comprise one of the most successful approaches to approximate Bayesian filtering. However, SMC without good proposal distributions struggle in high dimensions. We propose nested sequential Monte Carlo (NSMC), a methodology that generalizes the SMC framework by requiring only approximate, properly weighted, samples from the SMC proposal distribution, while still resulting in a correct SMC algorithm. This way we can *exactly approximate* e.g. the locally optimal proposal, and extend the class of models for which we can perform efficient inference using SMC. We show improved accuracy over other state-of-the-art methods on several spatio-temporal state space models.

1 Introduction

Inference in complex and high-dimensional statistical models is a very challenging problem that is ubiquitous in applications such as climate informatics (Monteleoni et al., 2013), bioinformatics (Cohen, 2004) and machine learning (Wainwright and Jordan, 2008).

We are interested in *sequential* Bayesian inference in settings where we have a sequence of posterior distributions that we need to compute. Furthermore, we focus on settings where the model (or state variable) is high-dimensional, but where there are *local dependencies* among state variables. One example of the type of models we consider are so-called spatio-temporal models (Cressie and Wikle, 2011; Rue and Held, 2005; Wikle, 2015).

Sequential Monte Carlo (SMC) methods comprise one of the most successful methodologies for sequential Bayesian inference. However, SMC struggles in high dimensions (Snyder et al., 2008) and these methods are rarely used for dimensions, say, higher than ten (Rebeschini and van Handel, 2015a). The purpose of the NSMC methodology is to push this limit well beyond the single digits. While the methodology is applicable to a wide range of different models, we focus on the spatio-temporal setting. We propose a class of spatio-temporal models based on a combination of Markov random fields and state space models. Inference in this model class is challenging, however, we show that the NSMC method is well suited to this challenge.

The basic strategy is to mimic the behavior of a so-called *fully adapted* (or locally optimal) SMC algorithm. Full adaptation can drastically improve the efficiency of SMC in high dimensions (Snyder et al., 2015). Unfortunately, it can rarely be implemented in practice since the fully adapted proposal distributions are typically intractable. NSMC addresses this difficulty by requiring only approximate, *properly weighted*, samples from the proposal distribution. This enables us to use a second layer of SMC to simulate approximately from the proposal. The proper weighting condition ensures the validity of NSMC, thus providing a generalization of the family of SMC methods. Furthermore, the NSMC procedure itself generates properly weighted samples, meaning that the procedure can be *nested* to an arbitrary degree. This paper extends preliminary work (Naesseth et al., 2015a) with the ability to handle more expressive models, more informative central limit theorems and convergence proofs, as well as new experiments.

Related work

There has been much recent interest in using Monte Carlo methods as nested procedures of other Monte Carlo algorithms. The SMC^2 and IS^2 algorithms by Chopin et al. (2013) and Tran et al. (2013), respectively, are algorithms for learning static parameters as well as latent variables. In these methods one SMC/IS method for the parameters is coupled with another for the latent variables. Chen

et al. (2011) and Johansen et al. (2012) address the state inference problem by splitting the state vector into two components and run coupled SMC samplers for these. Compared to the present work, these methods solve different problems and the "internal" SMC samplers are constructed differently—for approximate marginalization instead of simulation.

By viewing state inference as a sequential problem in the *components* of the state vector $\mathbf{x}_t$ we can make use of the method for general graphical models by Naesseth et al. (2014b). This method is combined with the island particle filter (Vergé et al., 2015), and studied more closely by Beskos et al. (2017) under the name space-time particle filter (ST-PF). The ST-PF does not generate an approximation of the fully adapted SMC. Another key distinction is that in the ST-PF each particle in the "outer" SMC sampler corresponds to a complete particle system, whereas for NSMC it will correspond to different hypotheses about the latent state $\mathbf{x}_t$ as in standard SMC. This leads to lower communication costs and better memory efficiency in e.g. distributed implementations. We have also found that NSMC typically outperforms ST-PF, even when run on a single machine with matched computing times (see Section 4).

The method proposed by Jaoua et al. (2013) can be viewed as a special case of NSMC when the nested procedure to generate samples is given by IS with the proposal being the transition probability. Independent resampling PF (IR-PF) introduced in Lamberti et al. (2016) generates samples in the same way as NSMC with IS, instead of SMC, as the nested procedure. However, IR-PF uses a different weighting that requires both the outer and the inner number of particles to tend to infinity for consistency. On the contrary, NSMC only requires that the number of particles at the outermost level tends to infinity for consistency (Section 3.5). Furthermore, we provide results in the supplementary material showing that NSMC significantly outperforming IR-PF on an example studied in Lamberti et al. (2016).

There are other SMC-related methods that have been introduced to tackle high-dimensional problems, e.g. the so-called block PF studied by Rebeschini and van Handel (2015b), the location particle smoother by Briggs et al. (2013), and various methods discussed in Djuric and Bugallo (2013). These methods are, however, all inconsistent because they are based on approximations that result in systematic errors.

The concept of proper weighting (or random weights) is not new and has been used in the so-called random weights particle filter (Fearnhead et al., 2010). They require exact samples from a proposal q_t but use a nested Monte Carlo method to unbiasedly estimate the importance weights w_t. In Martino et al. (2016) the authors study proper weighting as a means to perform *partial resampling*, i.e. only resample a subset of the particles at each time. The authors introduce the concept of "unnormalized" proper weighting, which is essentially the same as proper weighting that was introduced and used to motivate NSMC in Naesseth et al. (2015a). Furthermore, Stern (2015) uses proper weighting and NSMC to solve an inference problem within statistical historical linguistics.

Another approach to solve the sequential inference problem is the sequential Markov chain Monte Carlo class of methods (Yang and Dunson, 2013). It was shown by Septier and Peters (2016) that the optimal sequential MCMC algorithm actually is equivalent to the fully adapted SMC.

2 Sequential probabilistic models

Before presenting the new inference methodology in Section 3 we present two classes of sequential probabilistic models that will be used to illustrate its applicability. First we review the *Markov random field* (MRF) model and show how this can be used in a spatio-temporal setting via a sequential model decomposition. We then propose a combination of the MRF and a *state space model* (SSM), resulting in a more explicit separation of the spatial and temporal dependencies of the model. These models will serve to illustrate the usefulness and wide applicability of the method we propose. Note, however, that NSMC is by no means restricted to the classes of models we illustrate in this section. It can in principle be applied to any sequence of distributions we would like to approximate. We will refer to this sequence of distributions of interest as the *target distributions*.

2.1 Markov random fields

The Markov random field is a type of undirected probabilistic graphical model (Jordan, 2004). The MRF is typically not represented as a sequence of distributions (or models), but it has previously been shown (Everitt, 2012; Hamze and de Freitas, 2005; Lindsten et al., 2017; Naesseth et al., 2014a,b, 2015a,c) that it can be very useful to artificially introduce a sequence to simplify the inference problem. Furthermore, it is also possible to postulate the model as an MRF that increases with "time", useful in e.g. climate science (Fu et al., 2012).

Consider first a standard MRF model of a multivariate random variable $\mathbf{x} = (x_1, \ldots, x_{n_x})$, where $\mathbf{x} \in \mathsf{X}$. The conditional independencies among the model variables are described by the structure of the graph $G = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \ldots, n_x\}$ is the vertex set and $\mathcal{E} = \{(i, j) : (i, j) \in \mathcal{V} \times \mathcal{V},\ \exists \text{ edge between vertex } i \text{ and } j\}$ is the edge set. Given G we can define a joint probability density function (PDF) for $\mathbf{x}$ that incorporates this structure as

$$\pi(\mathbf{x}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \phi(x_i, y_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j), \tag{1}$$

where $\mathbf{y} = (y_1, \ldots, y_{n_x})$ is the observed variable and ϕ, ψ are called observation and interaction potentials, respectively. The normalization constant ensuring

that $\pi(\cdot)$ integrates to one is given by

$$Z := \int \prod_{i \in \mathcal{V}} \phi(x_i, y_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j) \mathrm{d}\mathbf{x}.$$

Note that (1) is usually referred to as a pairwise MRF in the literature due to $\pi(\cdot)$ factorizing into potentials that only depend on pairs of components of the random variable $\mathbf{x}$. For clarity we restrict ourselves to this type, however the method we propose in this paper can be applied to more general types of graphs, see e.g. Naesseth et al. (2014b) for ideas on how to extend SMC inference to non-pairwise MRFs.

Now, a sequential MRF is obtained if we consider a sequence of random variables $\mathbf{x}_{1:t} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$, with $\mathbf{x}_{1:t} \in \mathsf{X}^t$ for $t = 1, \ldots, T$, with a PDF that factorizes according to

$$\pi_t(\mathbf{x}_{1:t}) = \frac{1}{Z_t} \gamma_t(\mathbf{x}_{1:t}) := \frac{1}{Z_t} \gamma_t(\mathbf{x}_{1:t-1}) \prod_{i \in \mathcal{V}} \phi(x_{t,i}, y_{t,i}) \rho(\mathbf{x}_{t-1}, x_{t,i}) \prod_{(i,j) \in \mathcal{E}} \psi(x_{t,i}, x_{t,j}), \tag{2}$$

where $G = \{\mathcal{V}, \mathcal{E}\}$ again encodes the structure of the graphical model and $\rho(\cdot)$ is a new type of interaction potential that links $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$. Furthermore, the normalization constant is given by $Z_t := \int \gamma_t(\mathbf{x}_{1:t}) \mathrm{d}\mathbf{x}_{1:t}$. We illustrate a typical example of a sequential MRF in Figure 1. As an example, this type of model was used by Fu et al. (2012) in a spatio-temporal application to detect drought based on annual average precipitation rates collected from various sites in North America and Africa over the last century.

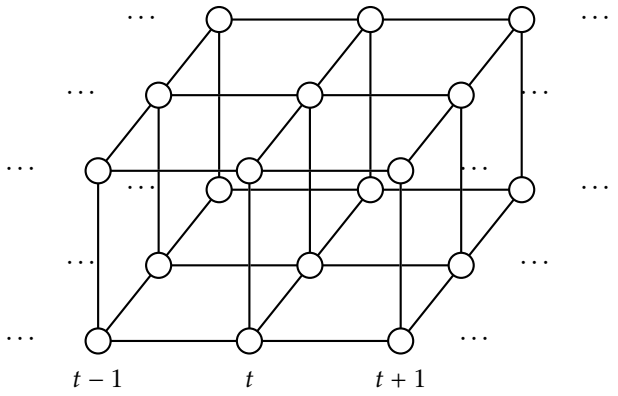


Figure 1: *Illustration of a sequential MRF where G is given by a 2×3 grid with nearest neighbor interaction.*

We would like to remark on one peculiarity that arises when the sequential MRF is used to model a spatio-temporal process. Consider $\pi_t(\cdot)$ without measure-

ments as a prior on a spatio-temporal model, i.e. the observation potentials ϕ in (2) do not depend on $\mathbf{y}_t$. In this case we get that the marginals for $t < T$ change depending on the value of T, i.e. in general $\pi_t(\mathbf{x}_{1:t}) \neq \pi_T(\mathbf{x}_{1:t}) = \int \pi_T(\mathbf{x}_{1:T}) \mathrm{d}\mathbf{x}_{t+1:T}$. Typically we would expect that *a priori* what happens for a dynamical process at time t should not be affected by the length of the time-series we consider. The next class of models we consider can introduce dependencies in both time and space without giving rise to this counter-intuitive result.

2.2 Spatio-temporal state space models

Before we move on to define the spatio-temporal state space model (ST-SSM), we will briefly review SSMs, a comprehensive and important model class commonly used for studying dynamical systems. For a more detailed account, and with pointers to the wide range of applications, we refer the readers to Cappé et al. (2005); Douc et al. (2014); Shumway and Stoffer (2010).

In state space models the sequential structure typically enters as a known, or postulated, dynamics on the unobserved latent state $\mathbf{x}_t$ that is then partially observed through the measurements $\mathbf{y}_t$. A common definition for SSMs is through its functional form

$$\mathbf{x}_t = a(\mathbf{x}_{t-1}, \mathbf{v}_t), \qquad \mathbf{v}_t \sim p_{\mathbf{v}}(\cdot), \tag{3a}$$

$$\mathbf{y}_t = c(\mathbf{x}_t, \mathbf{e}_t), \qquad \mathbf{e}_t \sim p_{\mathbf{e}}(\cdot), \tag{3b}$$

where $\mathbf{v}_t$ and $\mathbf{e}_t$—often called process and measurement noise, respectively—are random variables with some given distributions $p_{\mathbf{v}}(\cdot), p_{\mathbf{e}}(\cdot)$. Furthermore, the initial state $\mathbf{x}_1$ is a random variable with some distribution $\mu(\cdot)$. Equation (3) can equivalently be stated through conditional distributions

$$\mathbf{x}_t | \mathbf{x}_{t-1} \sim f(\mathbf{x}_t | \mathbf{x}_{t-1}), \tag{4a}$$

$$\mathbf{y}_t | \mathbf{x}_t \sim g(\mathbf{y}_t | \mathbf{x}_t), \tag{4b}$$

and we define the sequential probabilistic model (or target distribution) as follows

$$\pi_t(\mathbf{x}_{1:t}) := \frac{\gamma_t(\mathbf{x}_{1:t})}{Z_t} = \frac{1}{Z_t} \mu(\mathbf{x}_1) g(\mathbf{y}_1 | \mathbf{x}_1) \prod_{s=2}^{t} f(\mathbf{x}_s | \mathbf{x}_{s-1}) g(\mathbf{y}_s | \mathbf{x}_s). \tag{5}$$

We will assume that $g(\mathbf{y}_t | \mathbf{x}_t)$ is available and that it can be evaluated pointwise. This condition is satisfied in many practical applications.

A typical assumption when using the SSM to model spatio-temporal systems is to introduce the spatial dependency only between time steps $t-1$ and t, see e.g. Wikle and Hooten (2010). This can be achieved by defining the model such that the product of the induced distributions $f(\mathbf{x}_t | \mathbf{x}_{t-1}) g(\mathbf{y}_t | \mathbf{x}_t)$, conditionally on $\mathbf{x}_{t-1}$, completely factorize over the components of $\mathbf{x}_t$, see also (Rebeschini and van Handel, 2015b) where SMC applied to such a model is studied. However, we argue

that this approach can be limiting since any spatial dependencies are only implicitly taken into account via the temporal dynamics—indeed, conditionally on $\mathbf{x}_{t-1}$ the spatial components of $\mathbf{x}_t$ are assumed independent. Here we will instead study the case where spatial dependencies *within* each time step are introduced through the disturbance term $\mathbf{v}_t$. We define the ST-SSM as a combination of the functional and PDF representation of an SSM where the distribution for $\mathbf{v}_t$ is given by an MRF as in (1)

$$\mathbf{x}_t = \begin{pmatrix} x_{t,1} \\ \vdots \\ x_{t,n_x} \end{pmatrix} = \begin{pmatrix} a_1(\mathbf{x}_{t-1}, v_{t,1}) \\ \vdots \\ a_{n_x}(\mathbf{x}_{t-1}, v_{t,n_x}) \end{pmatrix}, \qquad \mathbf{v}_t \sim \frac{1}{Z_{\mathbf{v}}} \prod_{i \in \mathcal{V}} \phi(v_{t,i}) \prod_{(i,j) \in \mathcal{E}} \psi(v_{t,i}, v_{t,j}), \tag{6a}$$

$$\mathbf{y}_t | \mathbf{x}_t \sim g(\mathbf{y}_t | \mathbf{x}_t). \tag{6b}$$

We make no assumptions on local dependencies between $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, however, to keep it simple we will assume that the graph $G = \{\mathcal{V}, \mathcal{E}\}$ describing the distribution for $\mathbf{v}_t$ does not depend on time t. Furthermore, we will in this paper mainly consider models where dependencies between components in $\mathbf{v}_t$ are "few", i.e. the MRF is sparse with few elements in $\mathcal{E}$, and where components of $\mathbf{y}_t$ in $g(\cdot)$ only depend on subsets of $\mathbf{x}_t$. To illustrate the dependency structure in an ST-SSM we propose a combination of the traditional undirected graph for the MRF and the directed acyclic graph for the SSM, see Figure 2. This allows us to model

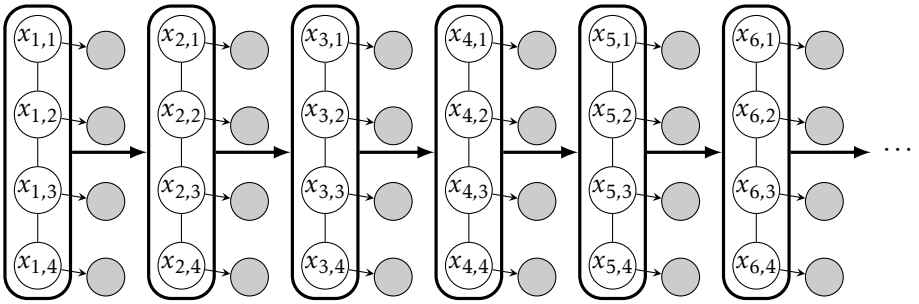


Figure 2: *Illustration of a spatio-temporal state space model with $n_x = 4$, one conditionally independent observation per component in $\mathbf{x}_t$, and the MRF for $\mathbf{v}_t$ is given by a chain. Grey circles illustrate the observations $\mathbf{y}_t = (y_{t,1}, \ldots, y_{t,4})$.*

more complex dynamical processes than Naesseth et al. (2015a) who assumed that $f(\mathbf{x}_t|\mathbf{x}_{t-1})g(\mathbf{y}_t|\mathbf{x}_t)$ factorized with only local dependencies between components of $\mathbf{x}_t$. Furthermore, we can clearly see that the peculiarity discussed in Section 2.1 is not present in this model; the marginal of the prior does not change with T as expected.

3 Methodology

Inference in sequential probabilistic models essentially boils down to computing the target distribution $\pi_t(\mathbf{x}_{1:t})$ for $t = 1, 2, \ldots$; typically an intractable problem with no analytical or simple numerical solution. This means that we have to resort to approximations. In this paper we focus on one particular successful solution to the problem, the so called sequential Monte Carlo family of algorithms first introduced in the papers by Gordon et al. (1993); Kitagawa (1993); Stewart and McCarty (1992).

The basic idea with SMC is to move a set of weighted samples $\{(\mathbf{x}^i_{1:t-1}, w^i_{t-1})\}_{i=1}^N$ (particles) approximating π_{t-1}, to a new set of particles $\{(\mathbf{x}^i_{1:t}, w^i_t)\}_{i=1}^N$ which approximate π_t. These samples define an empirical approximation of the target distribution

$$\pi_t^N(\mathrm{d}\mathbf{x}_{1:t}) := \sum_{i=1}^N \frac{w_t^i}{\sum_\ell w_t^\ell} \delta_{\mathbf{x}^i_{1:t}}(\mathrm{d}\mathbf{x}_{1:t}), \tag{7}$$

where $\delta_{\mathbf{x}}(\mathrm{d}\mathbf{x})$ is a Dirac measure at $\mathbf{x}$. In the next section we will detail a particularly efficient way of moving the particles, known as fully adapted SMC (Pitt and Shephard, 1999), ensuring that all normalized weights are equal to $\frac{1}{N}$.

3.1 Fully Adapted Sequential Monte Carlo

The procedure to move the particles and their weights from time $t-1$ to t in any SMC sampler is typically done in three stages. The first, *resampling*, stochastically chooses N particles at time $t-1$ that seem promising, discarding low-weighted ones. The second stage, *propagation*, generates new samples for time t conditioned on the resampled particles. The final stage, *weighting*, corrects for the discrepancy between the target distribution and the *proposal*, i.e. the instrumental distribution used in the propagation step.

Fully adapted SMC (Pitt and Shephard, 1999) makes specific choices on the resampling weights, ν_{t-1}, and the proposal, $q_t(\mathbf{x}_t|\mathbf{x}_{1:t-1})$, such that all the importance weights w_t are equal. By introducing *ancestor indices* $a^i_{t-1} \in \{1, \ldots, N\}$ for $i = 1, \ldots, N$, we can describe the resampling step (of the fully adapted SMC sampler) by simulating $\{a^i_{t-1}\}_{i=1}^N$ conditionally independently with

$$\mathbb{P}(a^i_{t-1} = j) = \frac{\nu^j_{t-1}}{\sum_{\ell=1}^M \nu^\ell_{t-1}}, \qquad \nu^j_{t-1} := \int \frac{\gamma_t\left((\mathbf{x}^j_{1:t-1}, \mathbf{x}_t)\right)}{\gamma_{t-1}(\mathbf{x}^j_{1:t-1})} \mathrm{d}\mathbf{x}_t. \tag{8}$$

Propagation then follows by simulating $\mathbf{x}^i_t$ conditionally on $\mathbf{x}^{a^i_{t-1}}_{1:t-1}$, for $i = 1, \ldots, N$,

according to

$$\mathbf{x}_t^i | \mathbf{x}_{1:t-1}^{a_{t-1}^i} \sim q_t(\mathbf{x}_t | \mathbf{x}_{1:t-1}^{a_{t-1}^i}) := \frac{1}{\nu_{t-1}^{a_{t-1}^i}} \frac{\gamma_t((\mathbf{x}_{1:t-1}^{a_{t-1}^i}, \mathbf{x}_t))}{\gamma_{t-1}(\mathbf{x}_{1:t-1}^{a_{t-1}^i})}, \tag{9}$$
$$\mathbf{x}_{1:t}^i = \left(\mathbf{x}_{1:t-1}^{a_{t-1}^i}, \mathbf{x}_t^i\right).$$

This proposal is sometimes referred to as the (locally) *optimal proposal* because it minimizes incremental variances in the importance weights w_t^i. Weighting is easy since all weights are equal, i.e. the unnormalized weights are all set to $w_t^i = 1$. The fully adapted SMC sampler in fact corresponds to a locally optimal choice of both resampling weights and proposal with an incremental variance in the importance weights w_t^i that is zero.

Note that in most cases it is impossible to implement this algorithm exactly, since we can not calculate ν_{t-1} and/or simulate from q_t. Nested SMC solves this by *requiring only approximate resampling weights and approximate samples from* q_t, in the sense that is formalized in Section 3.3. However, we will start by detailing some specific cases where we can efficiently implement exact fully adapted SMC. These cases are of interest in themselves, however, here we will mainly use them to build intuition for how the approximations in NSMC are constructed.

3.2 Leveraging Forward Filtering–Backward Simulation

The problems we need to solve are those of efficiently computing $\{\nu_{t-1}^i\}_{i=1}^N$ and simulating from q_t, defined in (8) and (9), respectively. There are at least two important special cases where we can use fully adapted SMC. The first is if the state space X is discrete and finite, i.e. $\mathbf{x}_t \in \{1, \ldots, S\}^{\otimes n_x}, \forall t$. Even though exact algorithms are known in this case (Cappé et al., 2005) the computational complexity scales quadratically with the cardinality of $\mathbf{x}_t$ (which is S^{n_x}). SMC methods can thus still be of interest (Fearnhead and Clifford, 2003; Naesseth et al., 2014a, 2015a). The second case is if $\frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$ is an unnormalized Gaussian distribution, e.g. in the ST-SSM this would correspond to

$$\mathbf{x}_t = a(\mathbf{x}_{t-1}) + \mathbf{v}_t, \qquad \mathbf{v}_t \sim \text{Gaussian MRF},$$
$$\mathbf{y}_t | \mathbf{x}_t \sim \mathcal{N}(\mathbf{y}_t; C\mathbf{x}_t, R),$$

for some matrix C, covariance matrix R, and an MRF in the components of $\mathbf{v}_t$ where all pair-wise potentials are Gaussian.

Now, even though in principle the fully adapted SMC is available in these special cases, the computational complexity can be prohibitive—it is of order $\mathcal{O}(S^{n_x})$ and $\mathcal{O}(n_x^3)$ for the finite state space and the Gaussian case, respectively. However, when there are local dependencies among state variables adhering to an underlying chain (or tree) structure it is possible to make use of efficient implementations

with only $\mathcal{O}(S^2 n_x)$ and $\mathcal{O}(n_x)$ complexity, respectively, as proposed by Naesseth et al. (2014a) for the finite state space case. This approach makes use of forward filtering–backward simulation (sampling), from Carter and Kohn (1994); Frühwirth-Schnatter (1994), on the *components* of $\mathbf{x}_t$ to compute $\{\nu_{t-1}^i\}_{i=1}^N$ and sample q_t exactly. As an example, let us consider the above ST-SSM with $C = I$ and $R = I$ and a Gaussian MRF given by

$$p_{\mathbf{v}}(\mathbf{v}_t) = \frac{1}{Z_{\mathbf{v}}} \exp\left(-\frac{\tau}{2}\sum_{d=1}^{n_x} v_{t,d}^2 - \frac{\lambda}{2}\sum_{d=2}^{n_x}(v_{t,d} - v_{t,d-1})^2\right),$$

for some positive constants τ and λ. For a state space model it is well known that the fully adapted SMC sampler corresponds to $q_t(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = \frac{f(\mathbf{x}_t|\mathbf{x}_{t-1})g(\mathbf{y}_t|\mathbf{x}_t)}{p(\mathbf{y}_t|\mathbf{x}_{t-1})}$ and $\nu_{t-1}^i = p(\mathbf{y}_t|\mathbf{x}_{t-1}^i)$. However, an efficient way of computing $\{\nu_{t-1}^i\}_{i=1}^N$ and simulating from $q_t(\,\cdot\,|\mathbf{x}_{1:t-1}^i)$ is

$$\mathbf{x}_t = a(\mathbf{x}_{t-1}^i) + \mathbf{v}_t', \qquad \mathbf{v}_t' \sim \frac{1}{\nu_{t-1}^i} g(\mathbf{y}_t|a(\mathbf{x}_{t-1}^i) + \mathbf{v}_t) p_{\mathbf{v}}(\mathbf{v}_t),$$
$$\nu_{t-1}^i = \int g(\mathbf{y}_t|a(\mathbf{x}_{t-1}^i) + \mathbf{v}_t) p_{\mathbf{v}}(\mathbf{v}_t) \mathrm{d}\mathbf{v}_t.$$

Due to the structure in $p_{\mathbf{v}}(\,\cdot\,)$ and $g(\mathbf{y}_t|\mathbf{x}_t)$ the distribution to sample from corresponds to a Gaussian MRF with a chain-structure in the $v_{t,d}$'s (cf. Figure 2)

$$p(\mathbf{v}_t|\mathbf{y}_t, \mathbf{x}_{t-1}^i) = \frac{g(\mathbf{y}_t|a(\mathbf{x}_{t-1}^i) + \mathbf{v}_t) p_{\mathbf{v}}(\mathbf{v}_t)}{\prod_{d=1}^{n_x} p(y_{t,d}|y_{t,1:d-1}, \mathbf{x}_{t-1}^i)}$$
$$\propto \exp\left(-\frac{1}{2}\sum_{d=1}^{n_x}\left[(y_{t,d} - a_d(\mathbf{x}_{t-1}^i) - v_{t,d})^2 + \tau v_{t,d}^2\right] - \frac{\lambda}{2}\sum_{d=2}^{n_x}(v_{t,d} - v_{t,d-1})^2\right). \quad (10)$$

Because of this structure we can efficiently compute the normalization constant of (10) by means of "forward" filtering over the components of the vector $\mathbf{v}_t$, keeping track of the incremental contributions to ν_{t-1}^i, $p(y_{t,d}|y_{t,1:d-1}, \mathbf{x}_{t-1}^i)$, $d = 1, \ldots, n_x$. Sampling the distribution is then done by an explicit "backward" pass, simulating $v_{t,d}' \sim p(v_{t,d}|v_{t,d+1:n_x}', y_{t,1:d}, \mathbf{x}_{t-1}^i)$, $d = n_x, n_x - 1, \ldots, 1$. We provide an illustration of the process in Figure 3. See also Naesseth et al. (2014a) for an example of how this is done in practice for a finite state space.

3.3 Nested Sequential Monte Carlo

The idea behind nested SMC is to emulate the forward-backward-based implementation of the fully adapted SMC sampler detailed in the previous section for arbitrary sequential probabilistic models. Because computing $\{\nu_{t-1}^i\}_{i=1}^N$ and simulating from q_t exactly is intractable in general we propose to run an SMC-based forward filtering–backward simulation method (Godsill et al., 2004; Lindsten and Schön, 2013) on the components of $\mathbf{x}_t$ (or $\mathbf{v}_t$) to approximate $\{\nu_{t-1}^i\}_{i=1}^N$

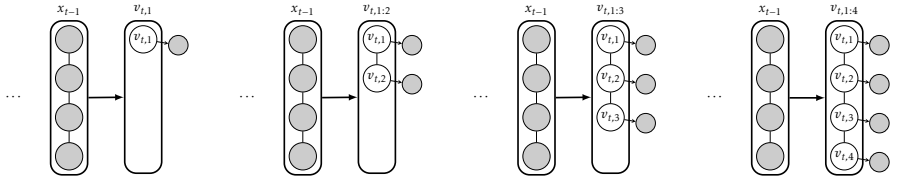


Figure 3: *Illustration of forward filtering for $\mathbf{v}_t$, we then obtain a sample from $\mathbf{v}'_t \sim p(\mathbf{v}_t|\mathbf{y}_t, \mathbf{x}^i_{t-1})$ using backward sampling for $v_{t,4}, \ldots, v_{t,1}$. Setting $\mathbf{x}_t = a(\mathbf{x}^i_{t-1}) + \mathbf{v}'_t$ yields a draw from the locally optimal proposal $q_t(\mathbf{x}_t|\mathbf{x}^i_{1:t-1})$.*

and generate approximate draws from q_t. As we shall see below, this can be viewed as an *exact approximation* (Andrieu et al., 2010) of a fully adapted SMC algorithm.

To describe the NSMC method we will start from a formal presentation of the procedure and the basic conditions needed for its validity. The description is based on the introduction of a generic auxiliary variable, here denoted by $\mathbf{u}_{t-1} \in \mathsf{U}_{t-1}$, for each time $t \geq 1$. Recall that $\{\pi_t(\mathbf{x}_{1:t}), t \geq 1\}$ with $\pi_t(\mathbf{x}_{1:t}) \propto \gamma_t(\mathbf{x}_{1:t})$ is a sequence of target distributions for the sampler. Let $\{\mathbf{x}^i_{1:t-1}\}_{i=1}^N$ denote an unweighted particle set ($w_{t-1} \equiv 1$) approximating π_{t-1}. Then, one step of the NSMC method, going from iteration $t-1$ to t, proceeds as follows: first, we sample conditionally independently the auxiliary variables $\mathbf{u}^i_{t-1} \sim \eta^M_{t-1}(\mathbf{u}_{t-1}|\mathbf{x}^i_{1:t-1})$, where η^M_{t-1} is some distribution parameterized by M. In Section 3.4 we will discuss how to make use of an internal SMC sampler to define this distribution, in which case the parameter M will denote the number of particles in the internal sampler. More precisely, in this case, sampling the auxiliary variables corresponds to the forward filtering step (cf. Section 3.2) where $\mathbf{u}_{t-1}$ denotes all the random variables (particles and weights) generated by the internal SMC sampler (with M particles) and η^M_{t-1} denotes the joint distribution of these variables (implicitly defined by the sampling procedure).

Next, we compute the resampling weights based on the auxiliary variables, $\nu^i_{t-1} = \tau_t(\mathbf{u}^i_{t-1})$, where τ_t is some real-valued function satisfying the requirements

$$\int \tau_t(\mathbf{u}_{t-1})\eta^M_{t-1}(\mathbf{u}_{t-1}|\mathbf{x}_{1:t-1})\mathrm{d}\mathbf{u}_{t-1} = \frac{\int \gamma_t((\mathbf{x}_{1:t-1}, \mathbf{x}_t))\mathrm{d}\mathbf{x}_t}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}, \quad \tau(\mathbf{u}_{t-1}) \geq 0 \text{ a.s.} \tag{11}$$

Below we will define τ_t as the normalization constant estimate at the final step of the internal SMC samplers and then the unbiasedness condition (11) is satisfied by known properties of SMC (Del Moral, 2004, Proposition 7.4.1). Next, we resample the particles $\{\mathbf{x}^i_{1:t-1}\}_{i=1}^N$ jointly with the auxiliary variables $\{\mathbf{u}^i_{t-1}\}_{i=1}^N$, by simulating ancestor variables $\{a^i_t\}_{i=1}^N$ with

$$\mathbb{P}(a^i_t = j) = \frac{\nu^j_{t-1}}{\sum_\ell \nu^\ell_{t-1}}, \qquad j = 1, \ldots, N. \tag{12}$$

Note that this means that we formally resample the complete state of the internal SMC samplers, captured by the auxiliary variables $\{\mathbf{u}_{t-1}^i\}_{i=1}^N$. In practical implementations there is no need to save multiple copies of the same internal state.

Next, for propagation we generate samples $\mathbf{x}_t^i \sim \kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1}^{a_t^i})$ from some distribution κ_t^M satisfying

$$\int \tau_t(\mathbf{u}_{t-1})\kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1})\eta_{t-1}^M(\mathbf{u}_{t-1}|\mathbf{x}_{1:t-1})\mathrm{d}\mathbf{u}_{t-1} = \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}. \tag{13}$$

The distribution κ_t^M can be realized by running (SMC-based) backward simulation, analogously to the procedure described in Section 3.2. However, a simple straightforward alternative that also satisfies (13) is to sample from the corresponding empirical distribution induced by the internal SMC sampler. We discuss the construction of η_{t-1}^M, κ_t^M and τ_t further in the next section.

Finally, we set $\mathbf{x}_{1:t}^i = (\mathbf{x}_{1:t-1}^{a_t^i}, \mathbf{x}_t^i)$, $i = 1, \ldots, N$, and have thus obtained a new set of unweighted particles approximating π_t, i.e.

$$\pi_t^N(\mathrm{d}\mathbf{x}_{1:t}) := \frac{1}{N}\sum_{i=1}^N \delta_{\mathbf{x}_{1:t}^i}(\mathrm{d}\mathbf{x}_{1:t}). \tag{14}$$

The auxiliary variables $\{\mathbf{u}_{t-1}^i\}_{i=1}^N$ can now be discarded.

Algorithm 1: Nested Sequential Monte Carlo (all for $i = 1, \ldots, N$)

Require: $\eta_{t-1}^M, \kappa_t^M, \tau_t$ that generate samples properly weighted for $\frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$

1: **for** $t = 1$ to T **do**
2: Simulate $\mathbf{u}_{t-1}^i \sim \eta_{t-1}^M(\mathbf{u}_{t-1}|\mathbf{x}_{1:t-1}^i)$
3: Draw a_t^i with probability $\mathbb{P}(a_t^i = j) = \frac{\tau_t(\mathbf{u}_{t-1}^j)}{\sum_\ell \tau_t(\mathbf{u}_{t-1}^\ell)}$
4: Simulate $\mathbf{x}_t^i \sim \kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1}^{a_t^i})$
5: Set $\mathbf{x}_{1:t}^i = (\mathbf{x}_{1:t-1}^{a_t^i}, \mathbf{x}_t^i)$
6: **end for**

The two conditions on $\eta_{t-1}^M, \tau_t, \kappa_t^M$, i.e. (11) and (13), can in fact be replaced by the single condition that $(\mathbf{x}_t^i, \tau_t(\mathbf{u}_{t-1}^i))$ are *properly weighted* for $\frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$.

Definition 1. Let γ_{t-1} and γ_t be unnormalized densities on X^{t-1} and X^t, respectively, and let $\mathbf{x}_{1:t-1}$ be in the support of γ_{t-1}. Let $(\mathbf{x}_t, \mathbf{u}_{t-1})$ be a random pair with distribution possibly depending on $\mathbf{x}_{1:t-1}$ and let $\tau_t : \mathsf{U}_{t-1} \to \mathbb{R}_+$. We say that $(\mathbf{x}_t, \tau_t(\mathbf{u}_{t-1}))$ are properly weighted for the (unnormalized) distribution $\frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$ if for all (suitably measurable) functions $h : \mathsf{X} \to \mathbb{R}$

$$\mathbb{E}[h(\mathbf{x}_t)\tau_t(\mathbf{u}_{t-1}) \mid \mathbf{x}_{1:t-1}] = C\int h(\mathbf{x}_t)\frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}\mathrm{d}\mathbf{x}_t, \tag{15}$$

for some positive constant $C > 0$ that is independent of the $\mathbf{x}$'s and $\mathbf{u}$'s.

We provide a summary of the proposed method in Algorithm 1. Although we focus on approximating the fully adapted SMC sampler, the extension to arbitrary resampling weights and proposal is straightforward, see the appendix. Next we will illustrate how we can make use of *nested* or internal SMC samplers to construct $\eta_{t-1}^M, \tau_t, \kappa_t^M$ that generate properly weighted samples.

3.4 Constructing η_{t-1}^M, τ_t and κ_t^M

To construct η_{t-1}^M we propose to run an SMC sampler targeting the components of $\mathbf{x}_t$ (or $\mathbf{v}_t$) one-by-one. Note that the internal SMC sampler is run with $\mathbf{x}_{1:t-1}$ fixed (to one of the N particles in the outer SMC sampler), and for notational simplicity we drop the dependence on $\mathbf{x}_{1:t-1}$ throughout this section. The internal SMC sampler is based on some sequence of (unnormalized) targets $p_d(x_{t,1:d})$ and proposals $r_d(x_{t,d}|x_{t,1:d-1})$, $d = 1, \ldots, n_x$ such that $p_{n_x}(x_{t,1:n_x}) \propto \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$. Note that $\mathbf{x}_t = x_{t,1:n_x}$. We provide a summary in Algorithm 2. In this case, the auxiliary variable $\mathbf{u}_{t-1}$ corresponds to all the random variables generated by the internal SMC sampler, i.e. $\mathbf{u}_{t-1} := \{x_{t,d}^{1:M}\}_{d=1}^{n_x} \bigcup \{a_{t,d}^{1:M}\}_{d=2}^{n_x}$. The function τ_t is naturally defined as the normalizing constant estimate for the internal SMC procedure, $\tau_t(\mathbf{u}_{t-1}) = \prod_{d=1}^{n_x} \frac{1}{M} \sum_{i=1}^{M} w_{t,d}^i$.

Algorithm 2: Internal Sequential Monte Carlo (all for $i = 1, \ldots, M$)

Require: Unnormalized target distributions $p_d(x_{t,1:d})$, proposals $r_d(x_{t,d}|x_{t,1:d-1})$, and M

1: $x_{t,1}^i \sim r_1(x_{t,1})$
2: Set $w_{t,1}^i = \frac{p_1(x_{t,1}^i)}{r_1(x_{t,1}^i)}$
3: **for** $d = 2$ to n_x **do**
4: Draw $a_{t,d}^i$ with probability $\mathbb{P}(a_{t,d}^i = j) = \frac{w_{t,d-1}^j}{\sum_\ell w_{t,d-1}^\ell}$
5: Simulate $x_{t,d}^i \sim r_d(x_{t,d}|x_{t,1:d-1}^{a_{t,d}^i})$
6: Set $x_{t,1:d}^i = (x_{t,1:d-1}^{a_{t,d}^i}, x_{t,d}^i)$
7: Set $w_{t,d}^i = \frac{p_d(x_{t,1:d}^i)}{p_{d-1}(x_{t,1:d-1}^{a_{t,d}^i}) r_d(x_{t,d}^i|x_{t,1:d-1}^{a_{t,d}^i})}$
8: **end for**

It remains to define the distribution κ_t^M, used to generate new particles $\mathbf{x}_t^i$ conditionally on the auxiliary variables. A first simple alternative is to simulate directly from the empirical measure defined by the approximation in Algorithm 2, leading to the following procedure.

Definition 2 (Internal SMC-based procedure). Let η_{t-1}^M, τ_t, and κ_t^M be defined as follows for some sequence $\{p_d(\cdot)\}_{d=1}^{n_x}$ such that $p_{n_x}(x_{t,1:n_x}) \propto \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$:

1. Simulate $\mathbf{u}_{t-1} \sim \eta_{t-1}^M(\mathbf{u}_{t-1}|\mathbf{x}_{1:t-1})$ by running Algorithm 2.
2. Set $\tau_t(\mathbf{u}_{t-1}) = \prod_{d=1}^{n_x} \frac{1}{M} \sum_{i=1}^M w_{t,d}^i$
3. Simulate $\mathbf{x}_t \sim \kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1}) := \sum_{i=1}^M \frac{w_{t,d}^i}{\sum_{j=1}^M w_{t,d}^j} \delta_{x_{t,1:n_x}^i}(\mathrm{d}\mathbf{x}_t)$.

However, although this approach will result in properly weighted samples (Proposition 1 below) it can introduce significant correlation between the samples. To mitigate this we propose to instead make use of backward simulation (Godsill et al., 2004; Lindsten and Schön, 2013) to construct a more efficient κ_t^M, see Algorithm 3. This mimics the exact backward-simulation-based implementation of the fully adapted SMC sampler discussed in Section 3.2. Note that we only need to draw once from κ_t^M for each outer-level particle, implying that for models with local spatial dependencies there is a small constant ($\lesssim 2$) computational overhead in using backward sampling in place of simply sampling from the empirical measure of the forward filter. The NSMC procedure using internal backward simulation is defined as follows.

Definition 3 (Internal SMC-based procedure with backward simulation). Let η_{t-1}^M and τ_t be defined as in Definition 2, but define κ_t^M as:

3'. Simulate $\mathbf{x}_t \sim \kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1})$ by running Algorithm 3

Algorithm 3: Internal Backward Simulation

Require: $\{(x_{t,1:d}^i, w_{t,d}^i)\}_{i=1}^M$, $d = 1, \ldots, n_x$ approximating $p_d(x_{t,1:d})$

1: Draw b_{n_x} with probability $\mathbb{P}(b_{n_x} = j) = \frac{w_{t,n_x}^j}{\sum_\ell w_{t,n_x}^\ell}$
2: Set $x_{t,n_x} = x_{t,n_x}^{b_{n_x}}$
3: **for** $d = n_x - 1$ to 1 **do**
4: Draw b_d with probability $\mathbb{P}(b_d = j) \propto w_{t,d}^j \frac{p_{n_x}\left((x_{t,1:d}^j, x_{t,d+1:n_x})\right)}{p_d(x_{t,1:d}^j)}$
5: Set $x_{t,d:n_x} = (x_{t,d}^{b_d}, x_{t,d+1:n_x})$
6: **end for**

Proposition 1 (Proper weighting). *The procedure in either Definition 2 or 3 generates* $(\mathbf{x}_t, \tau_t(\mathbf{u}_{t-1}))$ *that are properly weighted for* $\frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$.

Proof: The result follows from Theorem 2 in Naesseth et al. (2015a). □

Compare with the example in Section 3.2 and Figure 3 where we used forward filtering–backward sampling by considering the components of $v_{t,1:d}$ as our target. Instead of exact forward filtering we can use Algorithm 2, and instead of exact backward sampling we can use Algorithm 3, to generate properly weighted samples.

3.5 Theoretical Justification

In this section we will provide a central limit theorem that further motivates NSMC, and show how the asymptotic variance depends on the internal approximation of the exact fully adapted SMC. Furthermore, we provide a result that shows how this asymptotic variance converges to that of the corresponding asymptotic variance of the exact fully adapted SMC method as $M \to \infty$. We define the shorthand $\pi(f) := \int f(x)\pi(\mathrm{d}x)$ for a measure π and function f.

Theorem 1 (Central Limit Theorem). *Assume that η_{t-1}^M, τ_t, κ_t^M, generate properly weighted samples for $\gamma_t(\mathbf{x}_{1:t})/\gamma_{t-1}(\mathbf{x}_{1:t-1})$. Under certain (standard) regularity conditions, specified in the appendix, we have the following central limit theorem for any fixed M:*

$$\sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N}\varphi(\mathbf{x}_{1:t}^i) - \pi_t(\varphi)\right) \xrightarrow{d} \mathcal{N}\left(0, \Sigma_t^M(\varphi)\right),$$

where $\varphi : \mathsf{X}_t \to \mathbb{R}$, and the $\{\mathbf{x}_{1:t}^i\}_{i=1}^N$ are generated by Algorithm 1. The asymptotic variance is given by

$$\Sigma_t^M(\varphi) = \sum_{s=0}^{t}\sigma_{s,t}^M(\varphi),$$

for $\sigma_{s,t}^M(\varphi)$ defined by

$$\sigma_{t,t}^M(\varphi) = \pi_t\left((\varphi - \pi_t(\varphi))^2\right),$$

$$\sigma_{s,t}^M(\varphi) = \int \Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi)\pi_s(\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{1:s}, \quad \text{for } 0 < s < t,$$

$$\sigma_{0,t}^M(\varphi) = \int \frac{\tau_1(\mathbf{u}_0)^2}{Z_1^2}\left(\int (\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi))\,\frac{\pi_t(\mathbf{x}_{1:t})}{\pi_1(\mathbf{x}_1)}\kappa_1^M(\mathbf{x}_1|\mathbf{u}_0)\mathrm{d}\mathbf{x}_{1:t}\right)^2 \eta_0^M(\mathbf{u}_0)\mathrm{d}\mathbf{u}_0.$$

with

$$\Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi) := \mathbb{E}_{\eta_s^M(\mathbf{u}_s|\mathbf{x}_{1:s})}\left[\frac{Z_s^2}{Z_{s+1}^2}\tau_{s+1}(\mathbf{u}_s)^2\left(\int (\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi))\,\frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})}\kappa_{s+1}^M(\mathbf{x}_{s+1}|\mathbf{u}_s)\mathrm{d}\mathbf{x}_{s+1:t}\right)^2\right] \tag{16}$$

Proof: See the appendix. □

This theorem shows that, even for a fixed and finite value of M, the NSMC method obtains the standard $\sqrt{N}$ convergence rate. We can see how the asymptotic variance depends on how well we approximate q_t and its normalization constant with κ_t^M and τ_t. Furthermore, this lets us study convergence of the variance in M (and also analytic expressions for a high-dimensional state space model; see the subsequent section).

To show the convergence to fully adapted SMC as the approximation improves with increasing M we make some further assumptions detailed below.

Assumption E.1 (Uniform integrability). The sequence (in M) of random variables $\{\Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi)\}$, where $\mathbf{x}_{1:s} \sim \pi_s$, is uniformly integrable.

Note that a sufficient condition for Assumption E.1 to hold is that for some $\delta > 0$ and for all $s, M \geq 1$ the following holds

$$\int \Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi)^{1+\delta}\pi_s(\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{1:s} < \infty.$$

Assumption E.2 (Strong mixing). For all s, t, there exist constants $0 < \lambda_{s+1,t}^- < \infty$, $0 < \lambda_{s+1,t}^+ < \infty$ such that

$$\lambda_{s+1,t}^- \cdot \pi_t(\mathbf{x}_{s+2:t}|\mathbf{x}_{1:s+1}) \leq \frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})} \leq \lambda_{s+1,t}^+ \cdot \pi_t(\mathbf{x}_{s+2:t}|\mathbf{x}_{1:s+1}).$$

In the appendix we detail a weaker assumption for which Proposition 2 still holds.

Proposition 2. *Assume that the NSMC method uses the internal sampling procedure specified in Definition 2. Under the assumptions of Theorem 1, Assumption E.1 and E.2 the following limit holds:*

$$\lim_{M\to\infty} \Sigma_t^M(\varphi) =$$

$$= \pi_t\left((\varphi - \pi_t(\varphi))^2\right) + \sum_{s=1}^{t-1} \int \frac{\pi_t(\mathbf{x}_{1:s})^2}{\pi_s(\mathbf{x}_{1:s})} \left(\int \varphi(\mathbf{x}_{1:t})\pi_t(\mathbf{x}_{s+1:t}|\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{s+1:t} - \pi_t(\varphi) \right)^2 \mathrm{d}\mathbf{x}_{1:s}.$$

Proof: See the appendix. □

For simplicity we state the result for the case without backward sampling in the internal sampling procedure (Definition 2). However, the same result is expected to hold with backward sampling as well (Definition 3). Note that the limiting asymptotic variance is exactly the one derived for the fully adapted SMC asymptotic variance by Johansen and Doucet (2008).

To study the finite M behavior of NSMC we consider a fairly simple model and test function that leads to analytical expressions for the asymptotic variance in the CLT above. Specifically, we study a high-dimensional SSM, given in Definition 4, obtained by making n_x independent copies of an SSM. A similar model is considered by Beskos et al. (2014) for analyzing the stability of SMC samplers in high dimensions.

Definition 4. Define the spatially independent state space model as

$$\pi_t(\mathbf{x}_{1:t}) \propto \prod_{d=1}^{n_x} \left[\mu(x_{1,d}) \prod_{s=1}^{t} g(y_{s,d}|x_{s,d}) \prod_{s=2}^{t} f(x_{s,d}|x_{s-1,d}) \right].$$

For simplicity we also assume that $y_{s,d} = y_{s,e}, \forall d, e$ and that $\mathbb{E}_{\pi_t}[\mathbf{x}_t] = 0$.

For this model we can obtain explicit expressions for the asymptotic variances for the exact fully adapted SMC sampler as well as for the NSMC sampler.

Proposition 3. *For the model in Definition 4 and $\varphi(\mathbf{x}_{1:t}) = \sum_{d=1}^{n_x} x_{t,d}$, we have that the asymptotic variance of fully adapted SMC is given by*

$$\Sigma_t^{FA}(\varphi) = n_x A_t + \sum_{s=1}^{t-1} n_x B_s^{n_x-1} A_s + n_x(n_x-1) B_s^{n_x-2} C_s^2.$$

Furthermore, using $r(x_{s,d}|x_{s-1,d})$ as proposal in the inner SMC (Algorithm 2) of NSMC implemented according to Definition 3, we get that the asymptotic variance of NSMC is

$$\Sigma_t^M(\varphi) = n_x A_t + \sum_{s=0}^{t-1} \left[n_x B_s^{n_x-1} \left(A_s + M^{-1} \left(\tilde{A}_s - A_s \right) \right) \left(1 - \frac{1}{M} \right)^{n_x-1} \left(1 + \frac{\tilde{B}_s}{B_s(M-1)} \right)^{n_x-1} \right.$$
$$\left. + n_x(n_x-1) B_s^{n_x-2} \left(C_s + M^{-1} \left(\tilde{C}_s - C_s \right) \right)^2 \left(1 - \frac{1}{M} \right)^{n_x-2} \left(1 + \frac{\tilde{B}_s}{B_s(M-1)} \right)^{n_x-2} \right],$$

for the (finite) positive constants $A_t, A_s, \tilde{A}_s, B_s, \tilde{B}_s, C_s$, and $\tilde{C}_s$ defined in the appendix.

Proof: See the appendix. □

As expected the asymptotic variance of fully adapted SMC grows exponentially in the dimension n_x of the state. (Note that this result is asymptotic only in N, and not in n_x, in contrast to the result obtained by Snyder et al. (2015).) However, to control the additional approximation introduced by NSMC, i.e. not evaluating ν_{t-1} and sampling q_t exactly, we only need to scale $M \propto n_x$, even as $n_x \to \infty$.

We expect that intuition and rules-of-thumb from running standard SMC also apply to the internal approximation targeting $\gamma_t(\mathbf{x}_{1:t})/\gamma_{t-1}(\mathbf{x}_{1:t-1})$, rather than $\gamma_t(\mathbf{x}_{1:t})$. In Section 4.1 we empirically study how the choice of M affects the accuracy.

3.6 Modularity and implementation aspects

The procedure described by Algorithms 1–3 describes a nested SMC procedure with two levels that can be used to sample from high-dimensional models such as the sequential MRF or the ST-SSM described in Section 2. Since this procedure is based on using SMC on the components of each $\mathbf{x}_t$-vector it will, intuitively, work best when the dependencies among these components have a chain (or chain-like) structure, even though this is not a formal requirement. However, the methodology described in this section can be generalized to an arbitrary number of nested SMC procedures, which could prove useful for models with more than one spatial dimension. Indeed, the nested SMC procedure itself produces properly weighted samples (Naesseth et al., 2015a). For instance, in an ST-SSM (6) where the MRF describing the noise distribution is a 2-dimensional lattice, we may consider a three-level nested SMC procedure: at the first level the sampler operates on the temporal dimension, the second level simulates complete "rows" of states of the 2d lattice, and at the third level we sample the individual components of each "row". We investigate this three-level procedure numerically in Section 4.2.

An important aspect of this nesting of the proposed method is that it is completely modular, in the sense that the first level SMC does not need to be aware of the number or specific implementations of the consecutive levels. Indeed, as long as it has access to some procedure of generating properly weighted samples for $\gamma_t(\mathbf{x}_{1:t})/\gamma_{t-1}(\mathbf{x}_{1:t-1})$ it is possible to run Algorithm 1 without caring about *how* these samples are produced (whether we use one or several internal SMC samplers, e.g.).

Related to the modularity of the method it is worth noting that while Algorithm 1 describes the NSMC procedure in mathematical terms, this is not typically how one would like to implement the method in practice. Specifically, at line 2 of Algorithm 1 we simulate the auxiliary variables $\{\mathbf{u}_{t-1}^i\}_{i=1}^N$, which correspond to running N internal SMC samplers. This can be done in separate processes or, indeed, in a distributed environment on separate machines. Importantly, there is no need to return the auxiliary variables $\{\mathbf{u}_{t-1}^i\}_{i=1}^N$ to the "master process", which would incur a high communication cost. Instead, we simply return the estimates of the normalizing constants $\{\nu_{t-1}^i\}_{i=1}^N$, which is sufficient to carry out the resampling on line 3. Next, for the propagation step on line 4 we request samples from the internal procedures; the backward simulation is run internally in these processes and the resulting samples are returned.

4 Numerical Results

4.1 Gaussian Model

We start by considering a Gaussian spatio-temporal state space model where the exact solution is available via the Kalman filter (Kalman, 1960), and we can implement exact fully adapted SMC as explained in Section 3.2. The model is given by

$$\mathbf{x}_t = 0.5\mathbf{x}_{t-1} + \mathbf{v}_t, \quad \mathbf{v}_t \sim \frac{1}{Z_\mathbf{v}} \exp\left(-\frac{\tau}{2}\sum_{d=1}^{n_x} v_{t,d}^2 - \frac{\lambda}{2}\sum_{d=2}^{n_x}(v_{t,d} - v_{t,d-1})^2\right) \tag{17a}$$

$$\mathbf{y}_t|\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t, \sigma_y^2 I). \tag{17b}$$

The results for $N = 100$, $T = 10$, $\tau = \lambda = 1$ and $\sigma_y^2 = 0.25^2$, i.e. with fairly high signal-to-noise ratio, are given in Figure 4. We compare NSMC with (and without) backward simulation to the bootstrap particle filter (BPF) that uses the transition probability as proposal. We give all methods equivalent computational budget as the number of internal particles M grow, i.e. BPF gets $N_{\text{BPF}} = 100 \cdot M$ particles. Furthermore, for illustrative purposes we include fully adapted SMC (FAPF), the method that NSMC approximates, for a fixed number of particles $N_{\text{FAPF}} = 100$. The experiments are run ten times independently and we show the mean squared error (MSE) as well as one standard deviation error bars, for estimates of the log-likelihood, $\mathbb{E}[x_{T,1}]$ and $\mathbb{E}[x_{T,n_x}]$ with $n_x \in \{10, 100\}$. The expectations are with respect to the posterior distribution.

We can see that NSMC is significantly better than BPF and that it converges quickly towards the fully adapted SMC. Backward simulation also clearly helps with estimates of $\mathbb{E}[x_{T,d}]$ for $d = 1$, alleviating the correlation between generated samples. It is worthwhile to point out that for small M the NSMC seems to improve much more quickly than the standard *asymptotic* rate M^{-1}. For the likelihood estimate the rate almost exceeds M^{-4}. We provide results for different settings of σ_y^2 in the supplementary material. In general we see less striking improvement of NSMC over BPF when the signal to noise ratio is low, i.e. σ_y^2 is high compared to τ^{-1}.

4.2 Soil Carbon Cycles

We move on to study the performance of NSMC and compare it to ST-PF (Beskos et al., 2017) on a spatio-temporal model inspired by the soil carbon cycle model of (Clifford et al., 2014; Murray, 2016). The simplified model that we use to profile

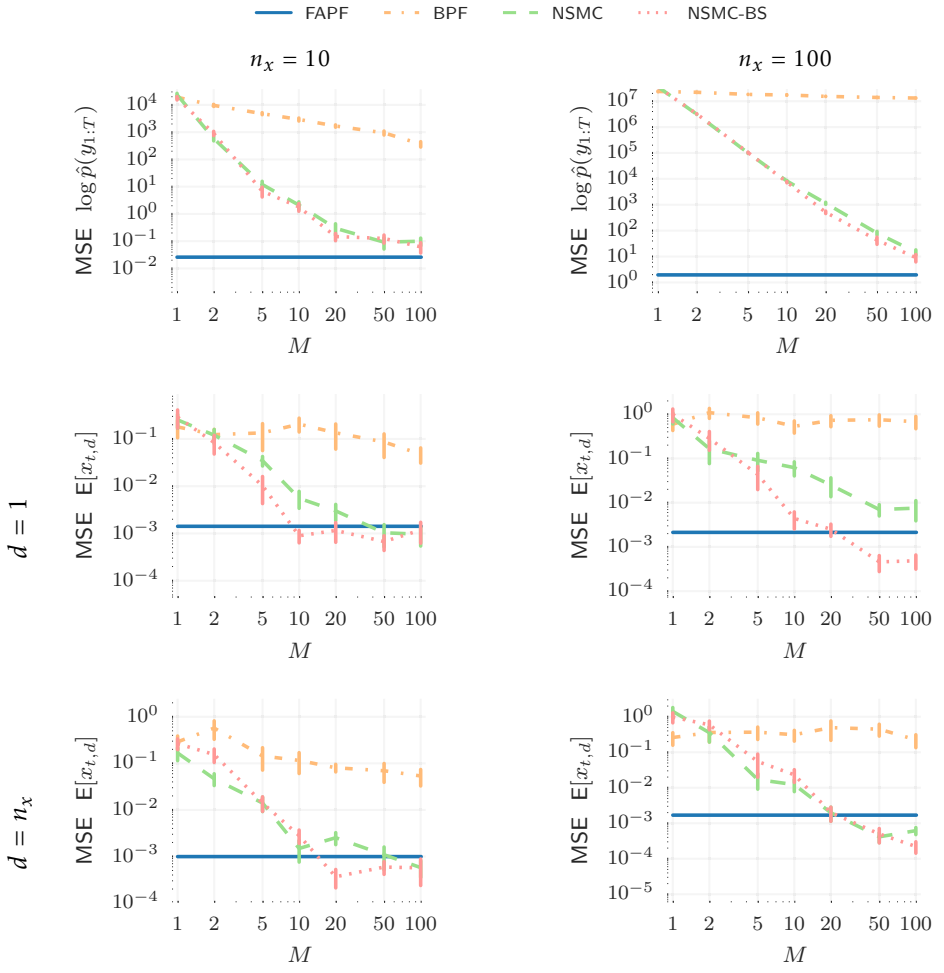


Figure 4: *MSE and error bars of Monte Carlo estimates of* $\log p(\mathbf{y}_{1:T})$, $\mathbb{E}[x_{T,1}]$, $\mathbb{E}[x_{T,n_x}]$ *for BPF, FAPF and two variants of NSMC.* $N = 100$ *for FAPF and NSMC and BPF has equivalent computational budget.*

the two state-of-the-art methods is defined by

$$\mathbf{x}_t = 0.5(\mathbf{x}_{t-1} + e^{\xi_t})e^{\mathbf{v}_t}, \qquad \mathbf{v}_t \sim \frac{1}{Z_{\mathbf{v}}} \exp\Big(-\frac{\tau}{2} \sum_{i \in \mathcal{V}} v_{t,i}^2 - \frac{\lambda}{2} \sum_{(i,j) \in \mathcal{E}} (v_{t,i} - v_{t,j})^2 \Big), \tag{18a}$$

$$\mathbf{y}_t | \mathbf{x}_t \sim \text{TruncatedNormal}\big(\mathbf{x}_t, \sigma^2 I, 0, \infty\big), \tag{18b}$$

where ξ_t is a known input signal and $(\mathcal{V}, \mathcal{E})$ is a square lattice, $\sqrt{n_x} \times \sqrt{n_x}$, with nearest neigbour interaction, i.e. $(i, j) \in \mathcal{E}$ if i and j are neighbors on the lattice. The latent variables $\mathbf{x}_t$ are positive and it is not possible to implement the exact fully adapted SMC method. We set $T = 25$, $n_x = 36$ (6×6), $\sigma = 0.2$, $\tau = 2$, and $\lambda = 1.0$ and run NSMC and ST-PF with matched computational complexity. Figure 5 displays the mean, over the n_x dimensions, mean squared error for each time-point t estimated by running the algorithms 10 times independently. We

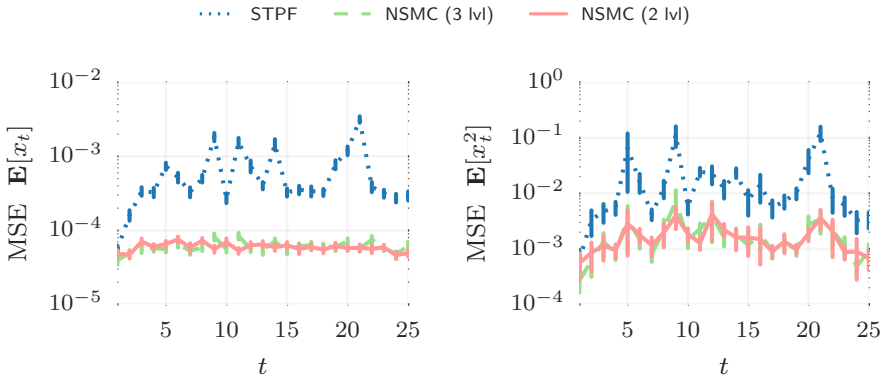


Figure 5: *Mean, over components d, MSE of $x_{t,d}, x_{t,d}^2$ estimated by ST-PF, and NSMC with two or three levels of inner SMC.*

can see that the different NSMC versions perform better than ST-PF in terms of MSE. This is without taking into account that NSMC simplifies distribution of the computation and is more memory efficient, only N rather than NM samples need to be retained at each step.

4.3 Mixture Model

Finally, we consider an example with a non-Gaussian ST-SSM, borrowed from Beskos et al. (2017) where the full details of the model are given. The transition probability $f(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a spatially localized Gaussian mixture and the measurement probability $g(\mathbf{y}_t|\mathbf{x}_t)$ is Student's t-distributed. The model dimension is $n_x = 1\,024$. Beskos et al. (2017) report improvements for ST-PF over both the BPF and the block PF by Rebeschini and van Handel (2015a). Following Beskos

et al. (2017) we use $N = M = 100$ for both ST-PF and NSMC and the BPF is given

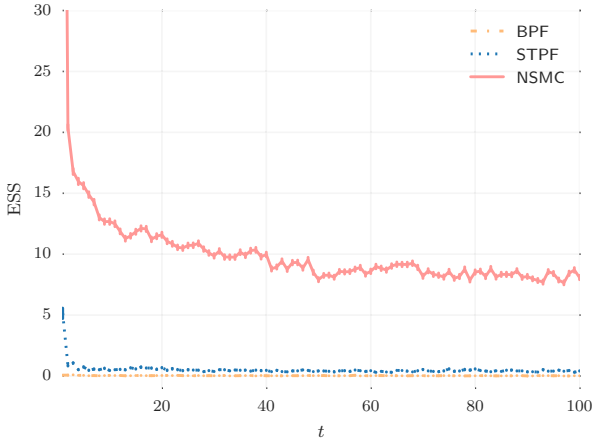


Figure 6: *Mean ESS, with one standard deviation error bars, for the non-Gaussian SSM.*

$N = 10\,000$. In Figure 6 we report the effective sample size (ESS, higher is better), estimated according to Carpenter et al. (1999). The ESS for the BPF is close to 0, for ST-PF around 1–2, and for NSMC slightly higher at 7–8. However, we note that all methods perform quite poorly on this model, and to obtain satisfactory results it would be necessary to use more particles.

Appendix

This appendix contains supplementary material for *High-dimensional Filtering using Nested Sequential Monte Carlo*.

A General Nested Sequential Monte Carlo

Assume that we are interested in approximating an arbitrary auxiliary SMC sampler with proposal $q_t(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = \frac{r_t(\mathbf{x}_t|\mathbf{x}_{1:t-1})}{\int r_t(\mathbf{x}_t|\mathbf{x}_{1:t-1})\mathrm{d}\mathbf{x}_t}$ and *adjustment multipliers* $\nu_{t-1}(\mathbf{x}_{1:t-1})$. The fully adapted SMC that we focus on in this paper is then attained as a special case when $q_t(\mathbf{x}_t|\mathbf{x}_{1:t-1}) \propto \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$ and $\nu_{t-1}(\mathbf{x}_{1:t-1}) = \int \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}\mathrm{d}\mathbf{x}_t$.

We can just as easily use a nested Monte Carlo method that produces properly weighted samples with respect to an arbitrary proposal q_t and multipliers ν_{t-1}, see Algorithm 4. Here $\hat{\nu}_{t-1}(\mathbf{x}_{1:t-1}, \mathbf{u}_{t-1})$ is an approximation to $\nu_{t-1}(\mathbf{x}_{1:t-1})$.

Algorithm 4: Nested Sequential Monte Carlo (all for $i = 1, \ldots, N$)

Require: $\hat{\nu}_{t-1}$ and η_{t-1}^M, κ_t^M, τ_t that generate samples properly weighted for $q_t(\mathbf{x}_t|\mathbf{x}_{1:t-1})$

1: **for** $t = 1$ to T **do**
2: Simulate $\mathbf{u}_{t-1}^i \sim \eta_{t-1}^M(\mathbf{u}_{t-1}|\mathbf{x}_{1:t-1}^i)$
3: Draw a_t^i with probability $\mathbb{P}(a_t^i = j) = \frac{\hat{\nu}_{t-1}(\mathbf{x}_{1:t-1}^j, \mathbf{u}_{t-1}^j) w_{t-1}^j}{\sum_\ell \hat{\nu}_{t-1}(\mathbf{x}_{1:t-1}^\ell, \mathbf{u}_{t-1}^\ell) w_{t-1}^\ell}$
4: Simulate $\mathbf{x}_t^i \sim \kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1}^{a_t^i})$
5: Set $\mathbf{x}_{1:t}^i = (\mathbf{x}_{1:t-1}^{a_t^i}, \mathbf{x}_t^i)$
6: Set $w_t^i = \frac{\gamma_t(\mathbf{x}_{1:t}^i)}{\gamma_{t-1}(\mathbf{x}_{1:t-1}^{a_t^i})} \frac{\tau_t(\mathbf{u}_{t-1}^{a_t^i})}{\hat{\nu}_{t-1}(\mathbf{x}_{1:t-1}^{a_t^i}, \mathbf{u}_{t-1}^{a_t^i}) r_t(\mathbf{x}_t^i|\mathbf{x}_{1:t-1}^{a_t^i})}$
7: **end for**

Note that if the adjustment multipliers $\hat{\nu}_{t-1}$ do not depend on $\mathbf{u}_{t-1}$, simulating from η_{t-1} can be done after resampling (simulating a_t). This ensures that the new samples are conditionally independent, thus decreasing correlation between samples.

Generating Properly Weighted Samples using IS

There are many ways of generating properly weighted samples with respect to a distribution, one example is using sequential Monte Carlo with or without backward simulation as explained in the main manuscript. However, perhaps one of the most straightforward and simple approaches is to use standard importance sampling. This means we would define η_{t-1}^M, κ_t^M, τ_t as follows:

$\eta_{t-1}^M(\mathbf{u}_{t-1}|\mathbf{x}_{1:t-1})$: Set $\mathbf{u}_{t-1} = \{\tilde{\mathbf{x}}_t^i\}_{i=1}^M$, where $\tilde{\mathbf{x}}_t^i \sim p_t(\mathbf{x}_t|\mathbf{x}_{1:t-1})$ for some proposal p_t,

$\kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1})$: Set $\mathbf{x}_t = \tilde{\mathbf{x}}_t^B$, where B is simulated with probability $\mathbb{P}(B = j) = \frac{\tilde{w}_t^j}{\sum_\ell \tilde{w}_t^\ell}$ and $w_t^j = \frac{r_t(\tilde{\mathbf{x}}_t^j|\mathbf{x}_{1:t-1})}{p_t(\tilde{\mathbf{x}}_t^j|\mathbf{x}_{1:t-1})}$,

$\tau_t(\mathbf{u}_{t-1})$: Set $\tau_t(\mathbf{u}_{t-1}) = \frac{1}{M} \sum_{i=1}^M \tilde{w}_t^i$.

It is straightforward to show that the above procedure generates properly weighted samples for q_t as long as $p_t > 0$ whenever q_t is. Now, if we want to use the above to approximate fully adapted SMC we simply let $r_t = \gamma_t / \gamma_{t-1}$ and $\hat{\nu}_{t-1} = \tau_t$.

B Theoretical Results

B.1 Proof of Theorem 1

We reproduce the central limit theorem of Naesseth et al. (2015a) here for clarity, see the Appendix of the extended version Naesseth et al. (2015b) for details.

Notation and Definitions

To explicitly state the general theorem we need some notation defined below. First, let

$$\Gamma_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) = \frac{\tau_t(\mathbf{u}_{t-1})\eta_t^M(\mathbf{u}_t|\mathbf{x}_{1:t})\kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1})}{r_t(\mathbf{x}_t|\mathbf{x}_{1:t-1})} \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})} \Gamma_{t-1}(\mathbf{x}_{1:t-1}, \mathbf{u}_{0:t-1}),$$
$$\Pi_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) = \frac{\Gamma_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t})}{Z_t},$$

with $\Pi_0(\mathbf{u}_0) = \eta_0^M(\mathbf{u}_0)$. If holds that Π_t is a probability density function on an extended space containing both the state variables $\mathbf{x}_{1:t}$ and the auxiliary variables $\mathbf{u}_{0:t}$. The NSMC method is in fact a standard SMC sampler targeting the sequence of distributions $\{\Pi_t\}_{t\geq 0}$ (Naesseth et al., 2015b). The proposal distribution for this sampler is given by

$$Q_t^M(\mathbf{x}_t, \mathbf{u}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_{t-1}) = \eta_t^M(\mathbf{u}_t|\mathbf{x}_{1:t})\kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1}),$$

and the weight function is given by

$$w_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) \propto \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})} \frac{\tau_t(\mathbf{u}_{t-1})}{\nu_{t-1}(\mathbf{x}_{1:t-1}, \mathbf{u}_{t-1})r_t(\mathbf{x}_t|\mathbf{x}_{1:t-1})}.$$

Furthermore, the central limit theorem of Naesseth et al. (2015b) is stated using the following "twisted" target distributions:

$$\Gamma_t'(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) = \nu_t(\mathbf{x}_{1:t}, \mathbf{u}_t)\Gamma_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}),$$
$$\Pi_t'(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) = \frac{\Gamma_t'(\mathbf{x}_{1:t}, \mathbf{u}_{0:t})}{\int \Gamma_t'(\mathbf{x}_{1:t}, \mathbf{u}_{0:t})\mathrm{d}\mathbf{x}_{1:t}\mathrm{d}\mathbf{u}_{0:t}}$$

and the ancillary (weight) functions

$$w_t'(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) = \frac{\Pi_t'(\mathbf{x}_{1:t}, \mathbf{u}_{0:t})}{Q_t^M(\mathbf{x}_t, \mathbf{u}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_{t-1})\Pi_{t-1}'(\mathbf{x}_{1:t-1}, \mathbf{u}_{0:t-1})} \propto \nu_t(\mathbf{x}_{1:t}, \mathbf{u}_t)w_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}),$$
$$\omega_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) = \frac{\Pi_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t})}{Q_t^M(\mathbf{x}_t, \mathbf{u}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_{t-1})\Pi_{t-1}'(\mathbf{x}_{1:t-1}, \mathbf{u}_{0:t-1})} \propto w_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}).$$

The domain of $\Pi_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t})$ is denoted by $\Theta_t = \mathsf{X}_t \times \mathsf{U}_t$. For a function $h : \mathsf{X}_t \to \mathbb{R}$, we define the extension of h to Θ_t by $h^e(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) := h(\mathbf{x}_{1:t})$. Let Φ_t be defined recursively to be the set of measurable functions $h : \Theta_t \to \mathbb{R}$ such that there exists a $\delta > 0$ with $\mathbb{E}_{Q_t^M \Pi'_{t-1}}[\|w'_t h\|^{2+\delta}] < \infty$, and such that $(\mathbf{x}_{1:t-1}, \mathbf{u}_{0:t-1}) \to \mathbb{E}_{Q_t^M}[w'_t h]$ is in Φ_{t-1}. We are now ready to state the more general central limit theorem of Naesseth et al. (2015a).

Theorem 2 (Central Limit Theorem). *Assume that $\varphi : \mathsf{X}_t \to \mathbb{R}$ is a function such that $\mathbb{E}_{Q_t^M \Pi'_{t-1}}[\|w'_t \varphi^e\|^{2+\delta}] < \infty$ for some $\delta > 0$, and that $(\mathbf{x}_{1:t-1}, \mathbf{u}_{0:t-1}) \to \mathbb{E}_{Q_t^M}[\omega_t \varphi^e]$ is in Φ_{t-1}. Then we have the following central limit theorem*

$$\sqrt{N}\left(\sum_{i=1}^{N} \frac{w_t^i}{\sum_{\ell=1}^{N} w_t^\ell} \varphi(\mathbf{x}_{1:t}^i) - \pi_t(\varphi)\right) \xrightarrow{d} \mathcal{N}\left(0, \Sigma_t^M(\varphi)\right),$$

where $\{(w_t^i, \mathbf{x}_{1:t}^i)\}_{i=1}^N$ are generated by Algorithm 4 and the asymptotic variance is given by

$$\Sigma_t^M(\varphi) = \tilde{V}_t^M(\omega_t(\varphi^e - \mathbb{E}_{\Pi_t}[\varphi^e])),$$

where $\tilde{V}_t^M$ is defined by the following set of recursions for measurable functions $h : \Theta_t \to \mathbb{R}$

$$\begin{aligned}
\tilde{V}_t^M(h) &= \hat{V}_{t-1}^M\left(\mathbb{E}_{Q_t^M}[h]\right) + \mathbb{E}_{\Pi'_{t-1}}\left[\mathrm{Var}_{Q_t^M}(h)\right], && t > 0,\\
V_t^M(h) &= \tilde{V}_t^M\left(w'_t(h - \mathbb{E}_{\Pi'_t}[h])\right), && t \geq 0,\\
\hat{V}_t^M(h) &= V_t^M(h) + \mathrm{Var}_{\Pi'_t}(h), && t \geq 0.
\end{aligned}$$

initialized by $\tilde{V}_0^M(h) = \mathrm{Var}_{\eta_0^M}(h)$ for $h : \Theta_0 \to \mathbb{R}$.

Approximating the Fully Adapted SMC

When we are approximating the fully adapted SMC, i.e. when $q_t(\mathbf{x}_t|\mathbf{x}_{1:t-1}) \propto \frac{\gamma_t(\mathbf{x}_{1:t})}{\gamma_{t-1}(\mathbf{x}_{1:t-1})}$ and $\nu_t(\mathbf{x}_{1:t}, \mathbf{u}_t) = \tau_{t+1}(\mathbf{u}_t)$, we can make significant simplifications of the expressions in the general central limit theorem above. We get that

$$\begin{aligned}
\Pi'_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) &= \frac{\tau_{t+1}(\mathbf{u}_t)}{Z_{t+1}} \Gamma_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}),\\
w'_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) &= \frac{Z_t}{Z_{t+1}} \tau_{t+1}(\mathbf{u}_t),\\
\omega_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}) &= 1.
\end{aligned}$$

Lemma 1. *The asymptotic variance $\Sigma_t^M(\varphi)$ in Theorem 2 when approximating the fully adapted SMC is given by*

$$\Sigma_t^M(\varphi) = \mathrm{Var}_{\eta_0^M}(h_0) + \sum_{s=1}^{t} \mathrm{Var}_{\Pi'_{s-1}, Q_s^M}(h_s), \tag{19}$$

for h_s defined by

$$
\begin{aligned}
h_t &= \varphi^e - \mathbb{E}_{\Pi_t}[\varphi^e], \\
h_s &= \frac{Z_s}{Z_{s+1}} \tau_{s+1}(\mathbf{u}_s) \left(\mathbb{E}_{Q_{s+1}^M}[h_{s+1}] - \mathbb{E}_{\Pi_s'} \left[\mathbb{E}_{Q_{s+1}^M}[h_{s+1}] \right] \right), \qquad 1 \leq s \leq t-1, \\
h_0 &= \frac{1}{Z_1} \tau_1(\mathbf{u}_0) \left(\mathbb{E}_{Q_1^M}[h_1] - \mathbb{E}_{\Pi_0'} \left[\mathbb{E}_{Q_1^M}[h_1] \right] \right),
\end{aligned}
$$

where $\Pi_0'(\mathbf{u}_0) = \frac{\tau_1(\mathbf{u}_0)}{Z_1} \eta_0^M(\mathbf{u}_0)$.

Proof: For a function $h_t : \Theta_t \to \mathbb{R}$ we have by Theorem 2 that

$$
\begin{aligned}
&\tilde{V}_t^M(h_t) = \hat{V}_{t-1}^M \left(\mathbb{E}_{Q_t^M}[h_t] \right) + \mathbb{E}_{\Pi_{t-1}'} \left[\mathrm{Var}_{Q_t^M}(h_t) \right] \\
&= V_{t-1}^M \left(\mathbb{E}_{Q_t^M}[h_t] \right) + \mathrm{Var}_{\Pi_{t-1}'} \left(\mathbb{E}_{Q_t^M}[h_t] \right) + \mathbb{E}_{\Pi_{t-1}'} \left[\mathrm{Var}_{Q_t^M}(h_t) \right] \\
&= \tilde{V}_{t-1}^M \left(w_{t-1}'(\mathbb{E}_{Q_t^M}[h_t] - \mathbb{E}_{\Pi_{t-1}'}[\mathbb{E}_{Q_t^M}[h_t]]) \right) + \mathrm{Var}_{\Pi_{t-1}'} \left(\mathbb{E}_{Q_t^M}[h_t] \right) + \mathbb{E}_{\Pi_{t-1}'} \left[\mathrm{Var}_{Q_t^M}(h_t) \right] \\
&= \ldots = \tilde{V}_{t-1}^M \left(\frac{Z_{t-1}}{Z_t} \tau_t(\mathbf{u}_{t-1}) \left(\mathbb{E}_{Q_t^M}[h_t] - \mathbb{E}_{\Pi_{t-1}'}[\mathbb{E}_{Q_t^M}[h_t]] \right) \right) + \mathrm{Var}_{\Pi_{t-1}', Q_t^M}(h_t)
\end{aligned}
$$

Recursion with $h_{t-1} := \frac{Z_{t-1}}{Z_t} \tau_t(\mathbf{u}_{t-1}) \left(\mathbb{E}_{Q_t^M}[h_t] - \mathbb{E}_{\Pi_{t-1}'}[\mathbb{E}_{Q_t^M}[h_t]] \right)$ gives the result. □

Next, we further simplify the terms of the asymptotic variance expression.

Lemma 2. *It holds that*

$$
h_t = \varphi - \pi_t(\varphi), \tag{20}
$$

$$
h_s = \frac{Z_s}{Z_{s+1}} \tau_{s+1}(\mathbf{u}_s) \int (\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi)) \frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})} \kappa_{s+1}^M(\mathbf{x}_{s+1}|\mathbf{u}_s) \mathrm{d}\mathbf{x}_{s+1:t}, \ 1 \leq s \leq t-1, \tag{21}
$$

$$
h_0 = \frac{1}{Z_1} \tau_1(\mathbf{u}_0) \int (\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi)) \frac{\pi_t(\mathbf{x}_{1:t})}{\pi_1(\mathbf{x}_1)} \kappa_1^M(\mathbf{x}_1|\mathbf{u}_0) \mathrm{d}\mathbf{x}_{1:t}, \tag{22}
$$

Proof: The first, h_t, follows straightforwardly by the definition of φ^e and Π_t. The remaining will be proved by induction. Assume that (21) holds for some $s \leq t-1$. We will now show that this in fact holds for both h_{t-1} and h_{s-1}; thus the result follows. Start by considering h_{t-1} using the definition in Lemma 1

$$
\begin{aligned}
&h_{t-1} = \frac{Z_{t-1}}{Z_t} \tau_t(\mathbf{u}_{t-1}) \left(\mathbb{E}_{Q_t^M}[h_t] - \mathbb{E}_{\Pi_{t-1}'} \left[\mathbb{E}_{Q_t^M}[h_t] \right] \right) = \\
&\frac{Z_{t-1}}{Z_t} \tau_t(\mathbf{u}_{t-1}) \left(\mathbb{E}_{Q_t^M}[\varphi - \pi_t(\varphi)] - 0 \right) = \frac{Z_{t-1}}{Z_t} \tau_t(\mathbf{u}_{t-1}) \left(\int \varphi(\mathbf{x}_{1:t}) \kappa_t^M(\mathbf{x}_t|\mathbf{u}_{t-1}) \mathrm{d}\mathbf{x}_t - \pi_t(\varphi) \right).
\end{aligned}
$$

Now, for h_{s-1} let us start by studying $\mathbb{E}_{Q_s^M}[h_s]$ and $\mathbb{E}_{\Pi'_{s-1}}\left[\mathbb{E}_{Q_s^M}[h_s]\right]$

$$\mathbb{E}_{Q_s^M}[h_s] = \mathbb{E}_{Q_s^M}\left[\frac{Z_s}{Z_{s+1}}\tau_{s+1}(\mathbf{u}_s)\int(\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi))\frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})}\kappa_{s+1}^M(\mathbf{x}_{s+1}|\mathbf{u}_s)\mathrm{d}\mathbf{x}_{s+1:t}\right]$$
$$= \ldots = \int(\varphi(\mathbf{x}_{1:t} - \pi_t(\varphi))\frac{\pi_t(\mathbf{x}_{1:t})}{\pi_s(\mathbf{x}_{1:s})}\kappa_s^M(\mathbf{x}_s|\mathbf{u}_{s-1})\mathrm{d}\mathbf{x}_{s:t},$$
$$\mathbb{E}_{\Pi'_{s-1}}\left[\mathbb{E}_{Q_s^M}[h_s]\right] = \ldots = 0.$$

This gives us that

$$\begin{aligned} h_{s-1} &= \frac{Z_{s-1}}{Z_s}\tau_s(\mathbf{u}_{s-1})\left(\mathbb{E}_{Q_s^M}[h_s] - \mathbb{E}_{\Pi'_{s-1}}\left[\mathbb{E}_{Q_s^M}[h_s]\right]\right) \\ &= \frac{Z_{s-1}}{Z_s}\tau_s(\mathbf{u}_{s-1})\int(\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi))\frac{\pi_t(\mathbf{x}_{1:t})}{\pi_s(\mathbf{x}_{1:s})}\kappa_s^M(\mathbf{x}_s|\mathbf{u}_{s-1})\mathrm{d}\mathbf{x}_{s:t}. \end{aligned}$$

The results follows by noting that the procedure is the same for h_0 taking into account edge effects, i.e. $Z_0 = 1$. □

Lemma 3. *It holds that*

$$\mathrm{Var}_{\Pi'_{t-1},Q_t^M}(h_t) = \pi_t\left((\varphi - \pi_t(\varphi))^2\right),$$
$$\mathrm{Var}_{\Pi'_{s-1},Q_s^M}(h_s) =$$
$$\int\left[\frac{Z_s^2\tau_{s+1}(\mathbf{u}_s)^2}{Z_{s+1}^2}\left(\int(\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi))\frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})}\kappa_{s+1}^M(\mathbf{x}_{s+1}|\mathbf{u}_s)\mathrm{d}\mathbf{x}_{s+1:t}\right)^2\right.$$
$$\left.\eta_s^M(\mathbf{u}_s|\mathbf{x}_{1:s-1})\pi_s(\mathbf{x}_{1:s})\right]\mathrm{d}\mathbf{u}_s\mathrm{d}\mathbf{x}_{1:s}, \qquad 1 \leq s \leq t-1,$$
$$\mathrm{Var}_{\eta_0^M}(h_0) = \int\frac{\tau_1(\mathbf{u}_0)^2}{Z_1^2}\left(\int(\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi))\frac{\pi_t(\mathbf{x}_{1:t})}{\pi_1(\mathbf{x}_1)}\kappa_1^M(\mathbf{x}_1|\mathbf{u}_0)\mathrm{d}\mathbf{x}_{1:t}\right)^2\eta_0^M(\mathbf{u}_0)\mathrm{d}\mathbf{u}_0$$

Proof: We get the first equality

$$\begin{aligned} \mathrm{Var}_{\Pi'_{t-1},Q_t^M}(h_t) &= \mathbb{E}_{\Pi'_{t-1},Q_t^M}\left[(\varphi - \pi_t(\varphi))^2\right] - \left(\mathbb{E}_{\Pi'_{t-1},Q_t^M}\left[\varphi - \pi_t(\varphi)\right]\right)^2 \\ &= \pi_t\left((\varphi - \pi_t(\varphi)^2\right), \end{aligned}$$

due to Lemma 2 and because

$$\Pi'_{t-1}(\mathbf{x}_{1:t-1}, \mathbf{u}_{0:t-1})Q_t^M(\mathbf{x}_t, \mathbf{u}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_{0:t-1}) = \Pi_t(\mathbf{x}_{1:t}, \mathbf{u}_{0:t}).$$

$$\mathrm{Var}_{\Pi'_{s-1},Q_s^M}(h_s) = \mathbb{E}_{\Pi'_{s-1},Q_s^M}\left[h_s^2\right] - \left(\mathbb{E}_{\Pi'_{s-1},Q_s^M}\left[h_s\right]\right)^2 = \mathbb{E}_{\Pi'_{s-1},Q_s^M}\left[h_s^2\right]$$
$$= \int h_s(\mathbf{x}_{1:s},\mathbf{u}_s)^2 \Pi_s(\mathbf{x}_{1:s},\mathbf{u}_{0:s})\mathrm{d}\mathbf{u}_{0:s}\mathrm{d}\mathbf{x}_{1:s} = \int h_s(\mathbf{x}_{1:s},\mathbf{u}_s)^2 \eta_s^M(\mathbf{u}_s|\mathbf{x}_{1:s})\pi_s(\mathbf{x}_{1:s})\mathrm{d}\mathbf{u}_s\mathrm{d}\mathbf{x}_{1:s}$$
$$= \int \left[\frac{Z_s^2 \tau_{s+1}(\mathbf{u}_s)^2}{Z_{s+1}^2} \left(\int \left(\varphi(\mathbf{x}_{1:t}) - \pi_t(\varphi)\right) \frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})} \kappa_{s+1}^M(\mathbf{x}_{s+1}|\mathbf{u}_s)\mathrm{d}\mathbf{x}_{s+1:t} \right)^2 \right.$$
$$\left. \eta_s^M(\mathbf{u}_s|\mathbf{x}_{1:s-1})\pi_s(\mathbf{x}_{1:s}) \right] \mathrm{d}\mathbf{u}_s\mathrm{d}\mathbf{x}_{1:s}, \qquad 1 \leq s \leq t-1,$$

where the second equality follows by noting that $\mathbb{E}_{\Pi'_{s-1},Q_s^M}[h_s] = 0$. Analogously to Lemma 2 the expression for $s = 0$ follows by taking into account the edge effects. □

Finally, with Lemmas 1, 2, and 3 together the result, i.e. Theorem 1, follows.

B.2 Proof of Proposition 2

Assumption E.3 (Approximation property). The approximations, based on η_s^M, κ_{s+1}^M and τ_s, of $q_s(\mathbf{x}_s|\mathbf{x}_{1:s-1}) \propto \frac{\pi_s(\mathbf{x}_{1:s})}{\pi_{s-1}(\mathbf{x}_{1:s-1})}$ and $\nu_{s-1}(\mathbf{x}_{1:s-1}) = \int \frac{\pi_s(\mathbf{x}_{1:s})}{\pi_{s-1}(\mathbf{x}_{1:s-1})}\mathrm{d}\mathbf{x}_s$ are such that

$$\Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi) \xrightarrow{\mathrm{d}} \frac{\pi_t(\mathbf{x}_{1:s})^2}{\pi_s(\mathbf{x}_{1:s})^2} \left(\int \varphi(\mathbf{x}_{1:t})\pi_t(\mathbf{x}_{s+1:t}|\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{s+1:t} - \pi_t(\varphi) \right)^2, \text{ as } M \to \infty, \tag{23}$$

where $\Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi)$ is defined in Theorem 1. Furthermore, assume that $\sigma_{0,t}^M(\varphi) \xrightarrow{\mathrm{d}} 0$ as $M \to \infty$.

Lemma 4. *The strong mixing assumption,*

$$\lambda_{s+1,t}^- \cdot \pi_t(\mathbf{x}_{s+2:t}|\mathbf{x}_{1:s+1}) \leq \frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})} \leq \lambda_{s+1,t}^+ \cdot \pi_t(\mathbf{x}_{s+2:t}|\mathbf{x}_{1:s+1}),$$

where $0 < \lambda_{s+1,t}^-, \lambda_{s+1,t}^+ < \infty$*, implies that*

$$\Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi) \xrightarrow{d} \frac{\pi_t(\mathbf{x}_{1:s})^2}{\pi_s(\mathbf{x}_{1:s})^2} \left(\int \varphi(\mathbf{x}_{1:t})\pi_t(\mathbf{x}_{s+1:t}|\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{s+1:t} - \pi_t(\varphi) \right)^2, \text{ as } M \to \infty. \tag{24}$$

Proof: Under the strong mixing assumption and given that we use a SMC method to generate properly weighted samples the result follows from standard SMC results (Del Moral, 2004). □

Theorem 3 (Vitali Convergence Theorem). *If* $\{\Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi)\}$ *is uniformly integrable and if* $\Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi) \xrightarrow{d} \Psi_{s,t}(\mathbf{x}_{1:s};\varphi)$*, then*

$$\lim_{M\to\infty} \int \Psi_{s,t}^M(\mathbf{x}_{1:s};\varphi)\pi_s(\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{1:s} = \int \Psi_{s,t}(\mathbf{x}_{1:s};\varphi)\pi_s(\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{1:s}.$$

Proof: See Folland (1999, Chapter 6). □

Under assumptions of uniform integrability and strong mixing (or Assumption E.3), the result now follows by using the Vitali convergence theorem 3 and noting that

$$\begin{aligned}
&\int \Psi_{s,t}(\mathbf{x}_{1:s};\varphi)\pi_s(\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{1:s} \\
&= \int \frac{\pi_t(\mathbf{x}_{1:s})^2}{\pi_s(\mathbf{x}_{1:s})}\left(\int \varphi(\mathbf{x}_{1:t})\pi_t(\mathbf{x}_{s+1:t}|\mathbf{x}_{1:s})\mathrm{d}\mathbf{x}_{s+1:t} - \pi_t(\varphi)\right)^2 \mathrm{d}\mathbf{x}_{1:s}.
\end{aligned}$$

B.3 Proposition 3

The constants in Proposition 3 are defined as follows

$$\begin{aligned}
A_t &= \int x_{t,d}^2 \pi_t(x_{1:t,d})\mathrm{d}x_{1:t,d}, \\
A_s &= \int \frac{\pi_t(x_{1:s,d})^2}{\pi_s(x_{1:s,d})}\left(\int x_{t,d}\pi_t(x_{t,d}|x_{s,d})\mathrm{d}x_{t,d}\right)^2 \mathrm{d}x_{1:s,d}, \\
\tilde{A}_s &= \int \frac{\pi_t(x_{1:s+1,d})^2}{\pi_s(x_{1:s,d})r(x_{s+1,d}|x_{s,d})}\left(\int x_{t,d}\pi_t(x_{t,d}|x_{s+1,d})\mathrm{d}x_{t,d}\right)^2 \mathrm{d}x_{1:s+1,d}, \\
B_s &= \int \frac{\pi_t(x_{1:s,d})^2}{\pi_s(x_{1:s,d})}\mathrm{d}x_{1:s,d}, \qquad \tilde{B}_s = \int \frac{\pi_t(x_{1:s+1,d})^2}{\pi_s(x_{1:s,d})r(x_{s+1,d}|x_{s,d})}\mathrm{d}x_{1:s+1,d}, \\
C_s &= \int \frac{\pi_t(x_{1:s,d})^2}{\pi_s(x_{1:s,d})}\int x_{t,d}\pi_t(x_{t,d}|x_{s,d})\mathrm{d}x_{t,d}\mathrm{d}x_{1:s,d}, \\
\tilde{C}_s &= \int \frac{\pi_t(x_{1:s+1,d})^2}{\pi_s(x_{1:s,d})r(x_{s+1,d}|x_{s,d})}\int x_{t,d}\pi_t(x_{t,d}|x_{s+1,d})\mathrm{d}x_{t,d}\mathrm{d}x_{1:s+1,d},
\end{aligned}$$

with $A_0 = 0$, $B_0 = 1$, $C_0 = 0$,

$$\begin{aligned}
\tilde{A}_0 &= \int \frac{\pi_t(x_{1,d})^2}{r(x_{1,d})}\left(\int x_{t,d}\pi_t(x_{t,d}|x_{1,d})\mathrm{d}x_{t,d}\right)^2 \mathrm{d}x_{1,d}, \\
\tilde{B}_0 &= \int \frac{\pi_t(x_{1,d})^2}{r(x_{1,d})}\mathrm{d}x_{1,d}, \\
\tilde{C}_0 &= \int \frac{\pi_t(x_{1,d})^2}{r(x_{1,d})}\int x_{t,d}\pi_t(x_{t,d}|x_{1,d})\mathrm{d}x_{t,d}\mathrm{d}x_{1,d},
\end{aligned}$$

and

$$\tilde{A}_{t-1} = \int \frac{\pi_t(x_{1:t,d})^2}{\pi_s(x_{1:t-1,d})r(x_{t,d}|x_{t-1,d})} x_{t,d}^2 \mathrm{d}x_{1:t,d},$$
$$\tilde{B}_{t-1} = \int \frac{\pi_t(x_{1:t,d})^2}{\pi_s(x_{1:t-1,d})r(x_{t,d}|x_{t-1,d})} \mathrm{d}x_{1:t,d},$$
$$\tilde{C}_{t-1} = \int \frac{\pi_t(x_{1:t,d})^2}{\pi_s(x_{1:t-1,d})r(x_{t,d}|x_{t-1,d})} x_{t,d} \mathrm{d}x_{1:t,d}.$$

B.3.1 Proof of Proposition 3

For fully adapted SMC we have from the result in Johansen and Doucet (2008) (see also our convergence result in the previous section) and for the model defined in the main manuscript

$$\pi_t((\varphi - \pi_t(\varphi))^2) = n_x A_t$$
$$\int \frac{\pi_t(\mathbf{x}_{1:s})^2}{\pi_s(\mathbf{x}_{1:s})} \left(\sum_{d=1}^{n_x} \int x_{t,d} \pi_t(\mathbf{x}_{s+1:t}|\mathbf{x}_{1:s}) \mathrm{d}\mathbf{x}_{s+1:t} \right)^2 \mathrm{d}\mathbf{x}_{1:s} =$$
$$= \sum_{e=1}^{n_x} \sum_{f=1}^{n_x} \int \left[\frac{\pi_t(\mathbf{x}_{1:s})^2}{\pi_s(\mathbf{x}_{1:s})} \int x_{t,e} \pi_t(\mathbf{x}_{s+1:t}|\mathbf{x}_{1:s}) \mathrm{d}\mathbf{x}_{s+1:t} \cdot \int x_{t,f} \pi_t(\mathbf{x}_{s+1:t}|\mathbf{x}_{1:s}) \mathrm{d}\mathbf{x}_{s+1:t} \right] \mathrm{d}\mathbf{x}_{1:s}$$
$$= \sum_{e=1}^{n_x} \sum_{f=1}^{n_x} \int \left[\frac{\pi_t(\mathbf{x}_{1:s})^2}{\pi_s(\mathbf{x}_{1:s})} \int x_{t,e} \pi_t(x_{t,e}|x_{s,e}) \mathrm{d}x_{t,e} \cdot \int x_{t,f} \pi_t(x_{t,f}|x_{s,f}) \mathrm{d}x_{t,f} \right] \mathrm{d}\mathbf{x}_{1:s}$$
$$= n_x B_s^{n_x - 1} A_s + n_x(n_x - 1) B_s^{n_x - 2} C_s^2,$$

with constants as defined above.

For nested SMC we have $r(\mathbf{x}_s|\mathbf{x}_{s-1}) = \prod_{d=1}^{n_x} r(x_{s,d}|x_{s-1,d})$ and due to the independence between dimensions we will have no dependence on internal ancestor variables in η_s, κ_{s+1}, τ_{s+1}, i.e.

$$\eta_s(\mathbf{u}_s|\mathbf{x}_{1:s}) = \prod_{d=1}^{n_x} \prod_{j=1}^{M} r(x_{s+1,d}^j|x_{s,d}),$$
$$\kappa_{s+1}(\mathbf{x}_{s+1}|\mathbf{u}_s) = \prod_{d=1}^{n_x} \sum_{j=1}^{M} \frac{w_d^j}{\sum_\ell w_d^\ell} \delta_{x_{s+1,d}^j}(\mathrm{d}x_{s+1,d}),$$
$$\tau_{s+1}(\mathbf{u}_s) = \prod_{d=1}^{n_x} \frac{1}{M} \sum_{j=1}^{M} w_d^j,$$
$$w_d^j = \frac{f(x_{s+1,d}^j|x_{s,d}) g(y_{s+1,d}|x_{s+1,d}^j)}{r(x_{s+1,d}^j|x_{s,d})}.$$

For the variance contribution of the final step we obtain $\pi_t((\varphi - \pi_t(\varphi))^2) = n_x \sigma_x^2$, the same result as fully adapted SMC. The remaining can be calculated as follows

$$\begin{aligned}
&\int \left[\frac{Z_s^2 \tau_{s+1}(\mathbf{u}_s)^2}{Z_{s+1}^2} \left(\int \varphi(\mathbf{x}_{1:t}) \frac{\pi_t(\mathbf{x}_{1:t})}{\pi_{s+1}(\mathbf{x}_{1:s+1})} \kappa_{s+1}^M(\mathbf{x}_{s+1}|\mathbf{u}_s) \mathrm{d}\mathbf{x}_{s+1:t} \right)^2 \right. \\
&\quad \left. \eta_s^M(\mathbf{u}_s|\mathbf{x}_{1:s-1}) \pi_s(\mathbf{x}_{1:s}) \right] \mathrm{d}\mathbf{u}_s \mathrm{d}\mathbf{x}_{1:s} \\
&= \frac{1}{p(\mathbf{y}_{s+1}|\mathbf{y}_{1:s})^2} \int \left[\tau_{s+1}(\mathbf{u}_s)^2 \frac{1}{M^{2n_x} \tau_{s+1}(\mathbf{u}_s)^2} \eta_s^M(\mathbf{u}_s|\mathbf{x}_{1:s-1}) \pi_s(\mathbf{x}_{1:s}) \right. \\
&\left(\sum_{e=1}^{n_x} \left[\sum_{j=1}^{M} w_e^j \frac{\int x_{t,e} \pi_t(x_{1:s,e}, x_{s+1,e}^j, x_{t,e}) \mathrm{d}x_{t,e}}{\pi_{s+1}(x_{1:s,e}, x_{s+1,e}^j)} \cdot \prod_{d \neq e} \sum_{j=1}^{M} w_d^j \frac{\pi_t(x_{1:s,d}, x_{s+1,d}^j)}{\pi_{s+1}(x_{1:s,d}, x_{s+1,d}^j)} \right] \right)^2 \left. \right] \mathrm{d}\mathbf{u}_s \mathrm{d}\mathbf{x}_{1:s} \\
&= \frac{1}{p(\mathbf{y}_{s+1}|\mathbf{y}_{1:s})^2 M^{2n_x}} \\
&\quad \sum_{e=1}^{n_x} \sum_{e'=1}^{n_x} \int \tilde{h}_e(\mathbf{x}_{1:s}, \mathbf{u}_s) \tilde{h}_{e'}(\mathbf{x}_{1:s}, \mathbf{u}_s) \prod_{d=1}^{n_x} \left[\pi_s(x_{1:s,d}) \prod_{j=1}^{M} r(x_{s+1,d}^j | x_{s,d}) \right] \mathrm{d}\mathbf{u}_s \mathrm{d}\mathbf{x}_{1:s}, \quad (25)
\end{aligned}$$

for $\tilde{h}_e$ defined by

$$\tilde{h}_e(\mathbf{x}_{1:s}, \mathbf{u}_s) = \sum_{j=1}^{M} w_e^j \frac{\int x_{t,e} \pi_t(x_{1:s,e}, x_{s+1,e}^j, x_{t,e}) \mathrm{d}x_{t,e}}{\pi_{s+1}(x_{1:s,e}, x_{s+1,e}^j)} \cdot \prod_{d \neq e} \sum_{j=1}^{M} w_d^j \frac{\pi_t(x_{1:s,d}, x_{s+1,d}^j)}{\pi_{s+1}(x_{1:s,d}, x_{s+1,d}^j)}.$$

Now, note that

$$\begin{aligned}
\tilde{h}_e(\mathbf{x}_{1:s}, \mathbf{u}_s)^2 &= \sum_{i_{1:n_x}, j_{1:n_x}} \left[\prod_{d=1}^{n_x} w_d^{i_d} w_d^{j_d} \cdot \prod_{d \neq e} \frac{\pi_t(x_{1:s,d}, x_{s+1,d}^{i_d})}{\pi_{s+1}(x_{1:s,d}, x_{s+1,d}^{i_d})} \frac{\pi_t(x_{1:s,d}, x_{s+1,d}^{j_d})}{\pi_{s+1}(x_{1:s,d}, x_{s+1,d}^{j_d})} \right. \\
&\quad \left. \cdot \frac{\int x_{t,e} \pi_t(x_{1:s,e}, x_{s+1,e}^{i_e}, x_{t,e}) \mathrm{d}x_{t,e}}{\pi_{s+1}(x_{1:s,e}, x_{s+1,e}^{i_e})} \frac{\int x_{t,e} \pi_t(x_{1:s,e}, x_{s+1,e}^{j_e}, x_{t,e}) \mathrm{d}x_{t,e}}{\pi_{s+1}(x_{1:s,e}, x_{s+1,e}^{j_e})} \right], \\
\tilde{h}_e(\mathbf{x}_{1:s}, \mathbf{u}_s) \tilde{h}_{e'}(\mathbf{x}_{1:s}, \mathbf{u}_s) &= \\
&\sum_{i_{1:n_x}, j_{1:n_x}} \left[\prod_{d=1}^{n_x} w_d^{i_d} w_d^{j_d} \cdot \prod_{d \neq e} \frac{\pi_t(x_{1:s,d}, x_{s+1,d}^{i_d})}{\pi_{s+1}(x_{1:s,d}, x_{s+1,d}^{i_d})} \prod_{d \neq e'} \frac{\pi_t(x_{1:s,d}, x_{s+1,d}^{j_d})}{\pi_{s+1}(x_{1:s,d}, x_{s+1,d}^{j_d})} \right. \\
&\quad \left. \cdot \frac{\int x_{t,e} \pi_t(x_{1:s,e}, x_{s+1,e}^{i_e}, x_{t,e}) \mathrm{d}x_{t,e}}{\pi_{s+1}(x_{1:s,e}, x_{s+1,e}^{i_e})} \frac{\int x_{t,e'} \pi_t(x_{1:s,e'}, x_{s+1,e'}^{j_{e'}}, x_{t,e'}) \mathrm{d}x_{t,e'}}{\pi_{s+1}(x_{1:s,e'}, x_{s+1,e'}^{j_{e'}})} \right],
\end{aligned}$$

with $e \neq e'$ and all $i_d, j_d \in \{1, \ldots, M\}$.

We will in the sequel also make use of the following observation

$$\frac{w_d^i}{\pi_{s+1}(x_{1:s,d}, x_{s+1,d}^i)} = \frac{p(y_{s+1,d}|y_{1:s,d})}{r(x_{s+1,d}^i|x_{s,d})\pi_s(x_{1:s,d})}. \tag{26}$$

Now, we consider the case in (25) when $e = e'$:

$$\frac{1}{p(\mathbf{y}_{s+1}|\mathbf{y}_{1:s})^2 M^{2n_x}} \int \tilde{h}_e(\mathbf{x}_{1:s}, \mathbf{u}_s)^2 \prod_{d=1}^{n_x} \left[\pi_s(x_{1:s,d}) \prod_{j=1}^{M} r(x_{s+1,d}^j|x_{s,d}) \right] \mathrm{d}\mathbf{u}_s \mathrm{d}\mathbf{x}_{1:s} =$$

$$\frac{1}{M^{2n_x}} \sum_{i_{1:n_x}, j_{1:n_x}} \left[\prod_{d \neq e} \int \frac{\prod_{j=1}^{M} r(x_{s+1,d}^j|x_{s,d})\pi_t(x_{1:s,d}, x_{s+1,d}^{i_d})\pi_t(x_{1:s,d}, x_{s+1,d}^{j_d})}{r(x_{s+1,d}^{i_d}|x_{s,d}) r(x_{s+1,d}^{j_d}|x_{s,d})\pi_s(x_{1:s,d})} \mathrm{d}\mathbf{u}_{s,d} \mathrm{d}x_{1:s,d} \right.$$

$$\cdot \int \left[\frac{\prod_{j=1}^{M} r(x_{s+1,e}^j|x_{s,e})\pi_t(x_{1:s,e})^2}{r(x_{s+1,e}^{i_e}|x_{s,e}) r(x_{s+1,e}^{j_e}|x_{s,e})\pi_s(x_{1:s,e})} \right.$$

$$\left. \left. \int x_{t,e}\pi_t(x_{s+1,e}^{i_e}, x_{t,e}|x_{s,e})\mathrm{d}x_{t,e} \int x_{t,e}\pi_t(x_{s+1,e}^{j_e}, x_{t,e}|x_{s,e})\mathrm{d}x_{t,e} \right] \mathrm{d}\mathbf{u}_{s,e} \mathrm{d}x_{1:s,e} \right]$$

$$= B_s^{n_x - 1} \left(A_s + M^{-1} \left(\tilde{A}_s - A_s \right) \right) \left(1 - \frac{1}{M} \right)^{n_x - 1} \left(1 + \frac{\tilde{B}_s}{B_s(M-1)} \right)^{n_x - 1},$$

where in the first equality we have used (26) and independency over dimensions. The second equality follows by straightforward (but tedious) calculations using combinatorial identities and noting that by definition of the model the constants do not depend on the dimension d.

Let us now consider the case in (25) when $e \neq e'$:

$$\frac{\int \tilde{h}_e(\mathbf{x}_{1:s}, \mathbf{u}_s)\tilde{h}_{e'}(\mathbf{x}_{1:s}, \mathbf{u}_s) \prod_{d=1}^{n_x} \left[\pi_s(x_{1:s,d}) \prod_{j=1}^{M} r(x_{s+1,d}^j|x_{s,d}) \right] \mathrm{d}\mathbf{u}_s \mathrm{d}\mathbf{x}_{1:s}}{p(\mathbf{y}_{s+1}|\mathbf{y}_{1:s})^2 M^{2n_x}} =$$

$$\frac{1}{M^{2n_x}} \sum_{i_{1:n_x}, j_{1:n_x}} \left[\prod_{d \neq e, e'} \int \frac{\prod_{j=1}^{M} r(x_{s+1,d}^j|x_{s,d})\pi_t(x_{1:s,d}, x_{s+1,d}^{i_d})\pi_t(x_{1:s,d}, x_{s+1,d}^{j_d})}{r(x_{s+1,d}^{i_d}|x_{s,d}) r(x_{s+1,d}^{j_d}|x_{s,d})\pi_s(x_{1:s,d})} \mathrm{d}\mathbf{u}_{s,d} \mathrm{d}x_{1:s,d} \right.$$

$$\int \frac{\pi_t(x_{1:s,e}, x_{s+1,e}^{j_e}) \prod_{j=1}^{M} r(x_{s+1,e}^j|x_{s,e})}{r(x_{s+1,e}^{i_e}|x_{s,e}) r(x_{s+1,e}^{j_e}|x_{s,e})\pi_s(x_{1:s,e})} \int x_{t,e}\pi_t(x_{1:s,e}, x_{s+1,e}^{i_e}, x_{t,e})\mathrm{d}x_{t,e} \mathrm{d}\mathbf{u}_{s,e} \mathrm{d}x_{1:s,e}$$

$$\left. \int \frac{\pi_t(x_{1:s,e'}, x_{s+1,e'}^{i_{e'}}) \prod_{j=1}^{M} r(x_{s+1,e'}^j|x_{s,e'})}{r(x_{s+1,e'}^{i_{e'}}|x_{s,e'}) r(x_{s+1,e'}^{j_{e'}}|x_{s,e'})\pi_s(x_{1:s,e'})} \int x_{t,e'}\pi_t(x_{1:s,e'}, x_{s+1,e'}^{j_e}, x_{t,e'})\mathrm{d}x_{t,e'} \mathrm{d}\mathbf{u}_{s,e'} \mathrm{d}x_{1:s,e'} \right]$$

$$= \left(C_s + M^{-1} \left(\tilde{C}_s - C_s \right) \right)^2 \left(1 - \frac{1}{M} \right)^{n_x - 2} \left(1 + \frac{\tilde{B}_s}{B_s(M-1)} \right)^{n_x - 2},$$

where again we have made use of independence over dimensions d and (26). The last equality follows again by straightforward manipulations and we can see that

product $\prod_{d\neq e,e'} \cdot$ is more or less equal to the one above, hence we obtain $B_s^{n_x-2}$ instead of $B_s^{n_x-1}$.

Putting all this together we get that

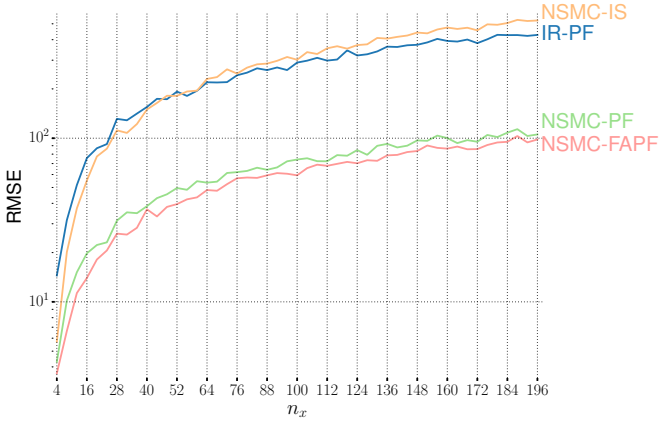
$$\Sigma_t^M(\varphi) = n_x A_t + \sum_{s=0}^{t-1}\Bigg[n_x B_s^{n_x-1}\left(A_s + M^{-1}\left(\tilde{A}_s - A_s\right)\right)\left(1-\frac{1}{M}\right)^{n_x-1}\left(1+\frac{\tilde{B}_s}{B_s(M-1)}\right)^{n_x-1} + n_x(n_x-1)B_s^{n_x-2}\left(C_s + M^{-1}\left(\tilde{C}_s - C_s\right)\right)^2\left(1-\frac{1}{M}\right)^{n_x-2}\left(1+\frac{\tilde{B}_s}{B_s(M-1)}\right)^{n_x-2}\Bigg],$$

equality follows by noting that $\sum_{e,e'} = \sum_e \sum_{e'=e} + \sum_e \sum_{e'\neq e}$ and that the constants do not depend on e/e'.

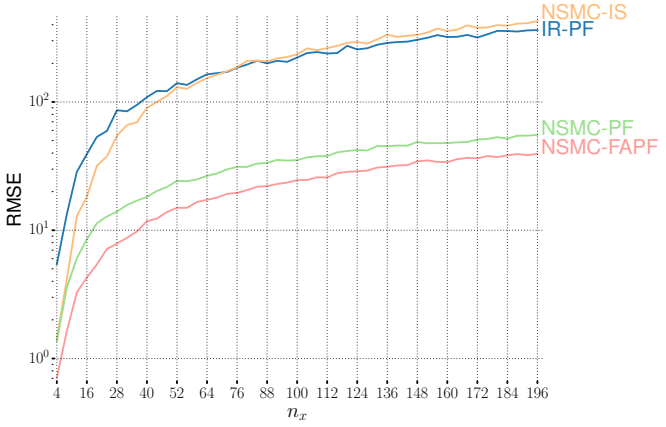
C Experiments

C.1 Comparison with Independent Resampling Particle Filter

We compare several variants of NSMC to Independent Resampling Particle Filter (IR-PF) on the same setup studied in Lamberti et al. (2016, High dimensional problems), for more information on the model and setup we refer to that paper. Figure 7 illustrates the results for $N = M \in \{10, 100\}$ and as we can see NSMC outperforms IR-PF significantly in root mean square error (RMSE). NSMC-IS and NSMC-PF both approximate the optimal proposal SMC and as such generate conditionally independent samples (see supplementary methods section above for how to use IS as a nested procedure). NSMC-FAPF, clearly the best of all of them, on the other hand, approximates the fully adapted SMC and generates conditionally *dependent* samples.



(a) $N = M = 10$



(b) $N = M = 100$

Figure 7: *RMSE of the IR-PF and three types of NSMC methods, approximation of optimal proposal SMC using IS (orange) and PF (green), approximation of fully adapted SMC using PF with BS (red).*

Bibliography

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.

Alexandros Beskos, Dan Crisan, and Ajay Jasra. On the stability of sequential monte carlo methods in high dimensions. *The Annals of Applied Probability*, 24(4):1396–1445, 08 2014.

Alexandros Beskos, Dan Crisan, Ajay Jasra, Kengo Kamatani, and Yan Zhou. A stable particle filter for a class of high-dimensional state-space models. *Advances in Applied Probability*, 49(1):24–48, 2017.

Jonathan Briggs, Michael Dowd, and Renate Meyer. Data assimilation for large-scale spatio-temporal systems using a location particle smoother. *Environmetrics*, 24(2):81–97, 2013.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer-Verlag New York, 2005.

J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings Radar, Sonar and Navigation*, 146(1):2–7, 1999.

Chris K Carter and Robert Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

Tianshi Chen, Thomas B. Schön, Henrik Ohlsson, and Lennart Ljung. Decentralized particle filter with arbitrary state decomposition. *IEEE Transactions on Signal Processing*, 59(2):465–478, Feb 2011.

Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos. SMC2: 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.

D. Clifford, D. Pagendam, J. Baldock, N. Cressie, R. Farquharson, M. Farrell, L. Macdonald, and L. Murray. Rethinking soil carbon modelling: a stochastic approach to quantify uncertainties. *Environmetrics*, 25(4):265–278, 2014.

Jacques Cohen. Bioinformatics—an introduction for computer scientists. *ACM Computing Surveys (CSUR)*, 36(2):122–158, 2004.

Noel Cressie and Christopher K. Wikle. *Statistics for spatio-temporal data*. Wiley, 2011.

Pierre Del Moral. *Feynman-Kac Formulae - Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer-Verlag New York, 2004.

Petar M Djuric and Mónica F Bugallo. Particle filtering for high-dimensional systems. In *Proceedings of the IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 352–355, 2013.

Randal Douc, Eric Moulines, and David Stoffer. *Nonlinear time series: Theory, methods and applications with R examples*. CRC Press, 2014.

Richard G. Everitt. Bayesian parameter estimation for latent Markov random fields and social networks. *Journal of Computational and Graphical Statistics*, 21(4):940–960, 2012.

Paul Fearnhead and Peter Clifford. On-line inference for hidden Markov models via particle filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(4):887–899, 2003.

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.

Gerald B. Folland. *Real analysis*. Pure and Applied Mathematics (New York). John Wiley & Sons, Inc., New York, second edition, 1999. Modern techniques and their applications.

Sylvia Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, 15(2):183–202, 1994.

Qiang Fu, Arindam Banerjee, Stefan Liess, and Peter K. Snyder. Drought detection of the last century: An MRF-based approach. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 24–34, Anaheim, CA, USA, April 2012.

Simon J. Godsill, Arnaud Doucet, and Mike West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465): 156–168, March 2004.

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.

Firas Hamze and Nando de Freitas. Hot coupling: a particle approach to inference and normalization on pairwise undirected graphs of arbitrary topology. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Nouha Jaoua, Emmanuel Duflos, Philippe Vanheeghe, and Francois Septier. Bayesian nonparametric state and impulsive measurement noise density estimation in nonlinear dynamic systems. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP*, pages 5755–5759, May 2013.

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

Adam M. Johansen, Nick Whiteley, and Arnaud Doucet. Exact approximation of Rao-Blackwellised particle filters. In *Proceesings of the 16th IFAC Symposium on System Identification (SYSID)*, pages 488–493, Brussels, Belgium, 2012.

Michael I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, 1960.

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.

Roland Lamberti, Yohan Petetin, Francois Desbouvries, and Francois Septier. Independent Resampling Sequential Monte Carlo Algorithms. *ArXiv e-prints*, July 2016.

F. Lindsten, A. M. Johansen, A. C. 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 (JCGS)*, 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.

Luca Martino, Victor Elvira, and Francisco Louzada. Weighting a resampled particles in sequential monte carlo (extended preprint). *viXra e-prints*, February 2016.

Claire Monteleoni, Gavin A. Schmidt, Francis Alexander, Alexandru Niculescu-Mizil, Karsten Steinhaeuser, Michael Tippett, Arindam Banerjee, M. Benno Blumenthal, Jason E. Smerdon Auroop R. Ganguly, and Marco Tedesco. Climate informatics. In *Computational Intelligent Data Analysis for Sustainable Development*. Chapman and Hall/CRC, London, 2013.

L. Murray. Personal communication, 2016.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Capacity estimation of two-dimensional channels using sequential Monte Carlo. In *Proceedings of the IEEE Information Theory Workshop (ITW)*, Hobart, Tasmania, Australia, November 2014a.

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, 2014b.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Nested sequential Monte Carlo methods. In *The 32nd International Conference on Machine Learning*, volume 37 of *JMLR W&CP*, pages 1292–1301, Lille, France, jul 2015a.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Nested sequential Monte Carlo methods. *Arxiv pre-print, arXiv:1502.02536v3*, 2015b.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Towards automated sequential Monte Carlo for probabilistic graphical models. In *NIPS Workshop on Black Box Inference and Learning*. Montreal, Canada, 2015c.

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

P. Rebeschini and R. van Handel. Can local particle filters beat the curse of dimensionality? *Annals of Applied Probability*, 25(5):2809–2866, 2015a.

Patrick Rebeschini and Ramon van Handel. Can local particle filters beat the curse of dimensionality? *Ann. Appl. Probab.*, 25(5):2809–2866, 10 2015b.

H. Rue and L. Held. *Gaussian Markov Random Fields, Theory and Applications*. CDC Press, Boca Raton, FL, USA, 2005.

F. Septier and G. W. Peters. Langevin and hamiltonian based sequential mcmc for efficient bayesian filtering in high-dimensional spaces. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):312–327, March 2016.

Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications: with R examples*. Springer Science & Business Media, 2010.

Chris Snyder, Thomas Bengtsson, Peter Bickel, and Jeff Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

Chris Snyder, Thomas Bengtsson, and Mathias Morzfeld. Performance bounds for particle filters using the optimal proposal. *Monthly Weather Review*, 143 (11):4750–4761, 2015.

Rafael Stern. *A statistical contribution to historical linguistics*. PhD thesis, Carnegie Mellon University, Department of Statistics, Carnegie Mellon University, Pittsburgh PA 15213, 5 2015.

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.

Minh-Ngoc Tran, Marcel Scharth, Michael K. Pitt, and Robert Kohn. Importance sampling squared for Bayesian inference in latent variable models. *ArXiv:1309.3339*, sep 2013.

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.

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

C. K. Wikle. Modern perspectives on statistics for spatio-temporal data. *WIREs Computational Statistics*, 7(1):86–98, 2015.

Christopher K. Wikle and Mevin B. Hooten. A general science-based framework for dynamical spatio-temporal models. *Test*, 19(3):417–451, 2010.

Y. Yang and D. B. Dunson. Sequential Markov Chain Monte Carlo. *arXiv:1308.3861*, August 2013.

Paper F

Interacting Particle Markov Chain Monte Carlo

Authors: Tom Rainforth, Christian A. Naesseth, Fredrik Lindsten, Brooks Paige, Jan-Willem van de Meent, Arnaud Doucet and Frank Wood

The two first authors contributed equally.

Edited version of the paper:

> Tom Rainforth, Christian A. Naesseth, Fredrik Lindsten, Brooks Paige, Jan-Willem Vandemeent, Arnaud Doucet, and Frank Wood. Interacting particle Markov chain Monte Carlo. In *International Conference on Machine Learning (ICML)*, pages 2616–2625, 2016.

This paper has been formatted to fit this layout.

Interacting Particle Markov Chain Monte Carlo

Tom Rainforth[†], Christian A. Naesseth[⋆], Fredrik Lindsten[‡], Brooks Paige[†], Jan-Willem van de Meent[†], Arnaud Doucet[†] and Frank Wood[†]

[†]The University of Oxford
Oxford, United Kingdom
{twgr,brooks,jwvdm,fwood}@robots.ox.ac.uk, doucet@stats.ox.ac.uk

[⋆]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

[‡]Department of Information Technology
Uppsala University
Uppsala, Sweden
fredrik.lindsten@it.uu.se

Abstract

We introduce *interacting particle Markov chain Monte Carlo* (iPMCMC), a PMCMC method based on an interacting pool of standard and conditional sequential Monte Carlo samplers. Like related methods, iPMCMC is a Markov chain Monte Carlo sampler on an extended space. We present empirical results that show significant improvements in mixing rates relative to both non-interacting PMCMC samplers, and a single PMCMC sampler with an equivalent memory and computational budget. An additional advantage of the iPMCMC method is that it is suitable for distributed and multi-core architectures.

1 Introduction

MCMC methods are a fundamental tool for generating samples from a posterior density in Bayesian data analysis (see e.g., Robert and Casella (2013)). Particle Markov chain Monte Carlo (PMCMC) methods, introduced by Andrieu et al. (2010), make use of sequential Monte Carlo (SMC) algorithms (Doucet et al., 2001; Gordon et al., 1993) to construct efficient proposals for the MCMC sampler.

One particularly widely used PMCMC algorithm is particle Gibbs (PG). The PG algorithm modifies the SMC step in the PMCMC algorithm to sample the latent variables conditioned on an existing particle trajectory, resulting in what is called a conditional sequential Monte Carlo (CSMC) step. The PG method was first introduced as an efficient Gibbs sampler for latent variable models with static

parameters (Andrieu et al., 2010). Since then, the PG algorithm and the extension by Lindsten et al. (2014) have found numerous applications in e.g. Bayesian non-parametrics (Tripuraneni et al., 2015; Valera et al., 2015), probabilistic programming (van de Meent et al., 2015; Wood et al., 2014) and graphical models (Everitt, 2012; Naesseth et al., 2014, 2015).

A drawback of PG is that it can be particularly adversely affected by *path degeneracy* in the CSMC step. Conditioning on an existing trajectory means that whenever resampling of the trajectories results in a common ancestor, this ancestor must correspond to this trajectory. Consequently, the mixing of the Markov chain for the early steps in the state sequence can become very slow when the particle set typically coalesces to a single ancestor during the CSMC sweep.

In this paper we propose the interacting particle Markov chain Monte Carlo (iPMCMC) sampler. In iPMCMC we run a pool of CSMC and unconditional SMC algorithms as parallel processes that we refer to as nodes. After each run of this pool, we apply successive Gibbs updates to the indexes of the CSMC nodes, such that the indices of the CSMC nodes changes. Hence, the nodes from which retained particles are sampled can change from one MCMC iteration to the next. This lets us trade off exploration (SMC) and exploitation (CSMC) to achieve improved mixing of the Markov chains. Crucially, the pool provides numerous candidate indices at each Gibbs update, giving a significantly higher probability that an entirely new retained particle will be "switched in" than in non-interacting alternatives.

This interaction requires only minimal communication; each node must report an estimate of the marginal likelihood and receive a new role (SMC or CSMC) for the next sweep. This means that iPMCMC is embarrassingly parallel and can be run in a distributed manner on multiple computers.

We prove that iPMCMC is a partially collapsed Gibbs sampler on the extended space containing the particle sets for all nodes. In the special case where iPMCMC uses only *one* CSMC node, it can in fact be seen as a non-trivial and unstudied instance of the α-SMC-based (Whiteley et al., 2016) PMCMC method introduced by Huggins and Roy (2015). However, with iPMCMC we extend this further to allow for an arbitrary number of CSMC and standard SMC algorithms with interaction. Our experimental evaluation shows that iPMCMC outperforms both independent PG samplers as well as a single PG sampler with the same number of particles run longer to give a matching computational budget.

An implementation of iPMCMC is provided in the probabilistic programming system *Anglican*[1] Wood et al. (2014), whilst illustrative MATLAB code, similar to that used for the experiments, is also provided[2].

[1] http://www.robots.ox.ac.uk/~fwood/anglican
[2] https://bitbucket.org/twgr/ipmcmc

2 Background

We start by briefly reviewing sequential Monte Carlo (Doucet et al., 2001; Gordon et al., 1993) and the particle Gibbs algorithm (Andrieu et al., 2010). Let us consider a non-Markovian latent variable model of the following form

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

where $x_t \in \mathsf{X}$ is the latent variable and $y_t \in \mathsf{Y}$ the observation at time step t, respectively, with transition densities f_t and observation densities g_t; x_1 is drawn from some initial distribution $\mu(\cdot)$. The method we propose is not restricted to the above model, it can in fact be applied to an arbitrary sequence of targets.

We are interested in calculating expectation values with respect to the posterior distribution $p(x_{1:T}|y_{1:T})$ on latent variables $x_{1:T} := (x_1, \ldots, x_T)$ conditioned on observations $y_{1:T} := (y_1, \ldots, y_T)$, which is proportional to the joint distribution $p(x_{1:T}, y_{1:T})$,

$$p(x_{1:T}|y_{1:T}) \propto \mu(x_1) \prod_{t=2}^{T} f_t(x_t|x_{1:t-1}) \prod_{t=1}^{T} g_t(y_t|x_{1:t}).$$

In general, computing the posterior $p(x_{1:T}|y_{1:T})$ is intractable and we have to resort to approximations. We will in this paper focus on, and extend, the family of particle Markov chain Monte Carlo algorithms originally proposed by Andrieu et al. (2010). The key idea in PMCMC is to use SMC to construct efficient proposals of the latent variables $x_{1:T}$ for an MCMC sampler.

2.1 Sequential Monte Carlo

The SMC method is a widely used technique for approximating a sequence of target distributions: in our case $p(x_{1:t}|y_{1:t}) = p(y_{1:t})^{-1} p(x_{1:t}, y_{1:t})$, $t = 1, \ldots, T$. At each time step t we generate a *particle system* $\{(x^i_{1:t}, w^i_t)\}_{i=1}^N$ which provides a weighted approximation to $p(x_{1:t}|y_{1:t})$. Given such a weighted particle system at time $t-1$, this is propagated forward in time to t by first drawing an ancestor variable a^i_{t-1} for each particle from its corresponding distribution:

$$\mathbb{P}(a^i_{t-1} = \ell) = \bar{w}^\ell_{t-1}. \qquad \ell = 1, \ldots, N, \tag{2}$$

where $\bar{w}^\ell_{t-1} = w^\ell_{t-1} / \sum_i w^i_{t-1}$. This is commonly known as the resampling step in the literature. We introduce the ancestor variables $\{a^i_{t-1}\}_{i=1}^N$ explicitly to simplify the exposition of the theoretical justification given in Section 3.1.

We continue by simulating from some given proposal density $x^i_t \sim q_t(x_t|x^{a^i_{t-1}}_{1:t-1})$

Algorithm 1: Sequential Monte Carlo (all for $i = 1, \ldots, N$)

1: **Input:** data $y_{1:T}$, number of particles N, proposals q_t
2: $x_1^i \sim q_1(x_1)$
3: $w_1^i = \frac{g_1(y_1|x_1^i)\mu(x_1^i)}{q_1(x_1^i)}$
4: **for** $t = 2$ **to** T **do**
5: $\quad a_{t-1}^i \sim \text{Discrete}\left(\left\{\bar{w}_{t-1}^\ell\right\}_{\ell=1}^N\right)$
6: $\quad x_t^i \sim q_t(x_t|x_{1:t-1}^{a_{t-1}^i})$
7: $\quad \text{Set } x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i)$
8: $\quad w_t^i = \frac{g_t(y_t|x_{1:t}^i) f_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}{q_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}$
9: **end for**

and re-weight the system of particles as follows:

$$w_t^i = \frac{g_t(y_t|x_{1:t}^i) f_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}{q_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}, \tag{3}$$

where $x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i)$. This results in a new particle system $\{(x_{1:t}^i, w_t^i)\}_{i=1}^N$ that approximates $p(x_{1:t}|y_{1:t})$. A summary is given in Algorithm 1.

2.2 Particle Gibbs

The PG algorithm (Andrieu et al., 2010) is a Gibbs sampler on the extended space composed of all random variables generated at one iteration, which still retains the original target distribution as a marginal. Though PG allows for inference over both latent variables and static parameters, we will in this paper focus on sampling of the former. The core idea of PG is to iteratively run *conditional* sequential Monte Carlo (CSMC) sweeps as shown in Algorithm 2, whereby each conditional trajectory is sampled from the surviving trajectories of the previous sweep. This *retained particle* index, b, is sampled with probability proportional to the final particle weights $\bar{w}_T^i$.

3 Interacting Particle Markov Chain Monte Carlo

The main goal of iPMCMC is to increase the efficiency of PMCMC, in particular particle Gibbs. The basic PG algorithm is especially susceptible to the *path degeneracy* effect of SMC samplers, i.e. sample impoverishment due to frequent resampling. Whenever the ancestral lineage collapses at the early stages of the

Algorithm 2: Conditional sequential Monte Carlo

Input: data $y_{1:T}$, number of particles N, proposals q_t, conditional trajectory $x'_{1:T}$
$x_1^i \sim q_1(x_1),\ i = 1,\dots,N-1$ and set $x_1^N = x'_1$
$w_1^i = \frac{g_1(y_1|x_1^i)\mu(x_1^i)}{q_1(x_1^i)},\ i = 1,\dots,N$
for $t = 2$ **to** T **do**
  $a_{t-1}^i \sim \text{Discrete}\left(\left\{\bar{w}_{t-1}^\ell\right\}_{\ell=1}^N\right),\ i = 1,\dots,N-1$
  $x_t^i \sim q_t(x_t|x_{1:t-1}^{a_{t-1}^i}),\ i = 1,\dots,N-1$
  Set $a_{t-1}^N = N$ and $x_t^N = x'_t$
  Set $x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i),\ i = 1,\dots,N$
  $w_t^i = \frac{g_t(y_t|x_{1:t}^i) f_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}{q_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})},\ i = 1,\dots,N$
end for

state sequence, the common ancestor is, by construction, guaranteed to be equal to the retained particle. This results in high correlation between the samples, and poor mixing of the Markov chain. To counteract this we might need a very high number of particles to get good mixing for all latent variables $x_{1:T}$, which can be infeasible due to e.g. limited available memory. iPMCMC can alleviate this issue by, from time to time, switching out a CSMC particle system with a completely independent SMC one, resulting in improved mixing.

iPMCMC, summarized in Algorithm 3, consists of M interacting separate CSMC and SMC algorithms, exchanging only very limited information at each iteration to draw new MCMC samples. We will refer to these internal CSMC and SMC algorithms as nodes, and assign an index $m = 1,\dots,M$. At every iteration, we have P nodes running local CSMC algorithms, with the remaining $M-P$ nodes running independent SMC. The CSMC nodes are given an identifier $c_j \in \{1,\dots,M\},\ j = 1,\dots,P$ with $c_j \neq c_k,\ k \neq j$ and we write $c_{1:P} = \{c_1,\dots,c_P\}$. Let $\mathbf{x}_m^i = x_{1:T,m}^i$ be the internal particle trajectories of node m.

Suppose we have access to P trajectories $\mathbf{x}'_{1:P}[0] = (\mathbf{x}'_1[0],\dots,\mathbf{x}'_P[0])$ corresponding to the initial retained particles, where the index $[\,\cdot\,]$ denotes MCMC iteration. At each iteration r, the nodes $c_{1:P}$ run CSMC (Algorithm 2) with the previous MCMC sample $\mathbf{x}'_j[r-1]$ as the retained particle. The remaining $M-P$ nodes run standard (unconditional) SMC, i.e. Algorithm 1. Each node m returns an estimate of the marginal likelihood for the internal particle system defined as

$$\hat{Z}_m = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} w_{t,m}^i. \tag{4}$$

The new conditional nodes are then set using a single loop $j = 1 : P$ of Gibbs

Algorithm 3: iPMCMC sampler

Input: number of nodes M, conditional nodes P and MCMC steps R, initial $\mathbf{x}'_{1:P}[0]$
for $r = 1$ **to** R **do**
  Workers $1 : M \backslash c_{1:P}$ run Algorithm 1 (SMC)
  Workers $c_{1:P}$ run Algorithm 2 (CSMC), conditional on $\mathbf{x}'_{1:P}[r-1]$ respectively.
  for $j = 1$ **to** P **do**
    Select a new conditional node by simulating c_j according to (5).
    Set new MCMC sample $\mathbf{x}'_j[r] = \mathbf{x}_{c_j}^{b_j}$ by simulating b_j according to (7)
  end for
end for

updates, sampling new indices c_j where

$$\mathbb{P}(c_j = m | c_{1:P\backslash j}) = \hat{\zeta}_m^j \tag{5}$$

$$\text{and} \quad \hat{\zeta}_m^j = \frac{\hat{Z}_m \mathbf{1}_{m \notin c_{1:P\backslash j}}}{\sum_{n=1}^M \hat{Z}_n \mathbf{1}_{n \notin c_{1:P\backslash j}}}, \tag{6}$$

defining $c_{1:P\backslash j} = \{c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_P\}$. We thus loop once through the conditional node indices and resample them from the union of the current node index and the unconditional node indices[3], in proportion to their marginal likelihood estimates. This is the key step that lets us switch completely the nodes from which the retained particles are drawn.

One MCMC iteration r is concluded by setting the new samples $\mathbf{x}'_{1:P}[r]$ by simulating from the corresponding conditional node's, c_j, internal particle system

$$\begin{aligned} \mathbb{P}(b_j = i | c_j) &= \bar{w}_{T,c_j}^i, \\ \mathbf{x}'_j[r] &= \mathbf{x}_{c_j}^{b_j}. \end{aligned} \tag{7}$$

The potential to pick from updated nodes c_j, having run independent SMC algorithms, decreases correlation and improves mixing of the MCMC sampler. Furthermore, as each Gibbs update corresponds to a one-to-many comparison for maintaining the same conditional index, the probability of switching is much higher than in an analogous non-interacting system.

The theoretical justification for iPMCMC is independent of how the initial trajectories $\mathbf{x}'_{1:P}[0]$ are generated. One simple and effective method (that we use in our experiments) is to run standard SMC sweeps for the "conditional" nodes at the first iteration.

[3]Unconditional node indices here refers to all $m \notin c_{1:P}$ at that point in the loop. It may thus include nodes who just ran a CSMC sweep, but have been "switched out" earlier in the loop.

The iPMCMC samples $\mathbf{x}'_{1:P}[r]$ can be used to estimate expectations for test functions $f : \mathsf{X}^T \mapsto \mathbb{R}$ in the standard Monte Carlo sense, with

$$\mathbb{E}[f(\mathbf{x})] \approx \frac{1}{RP} \sum_{r=1}^{R} \sum_{j=1}^{P} f(\mathbf{x}'_j[r]). \tag{8}$$

However, we can improve upon this if we have access to all particles generated by the algorithm, see Section 3.2.

We note that iPMCMC is suited to distributed and multi-core architectures. In practise, the particle to be retained, should the node be a conditional node at the next iteration, can be sampled upfront and discarded if unused. Therefore, at each iteration, only a single particle trajectory and normalisation constant estimate need be communicated between the nodes, whilst the time taken for calculation of the updates of $c_{1:P}$ is negligible. Further, iPMCMC should be amenable to an asynchronous adaptation under the assumption of a random execution time, independent of $\mathbf{x}'_j[r-1]$ in Algorithm 3. We leave this asynchronous variant to future work.

3.1 Theoretical Justification

In this section we will give some crucial results to justify the proposed iPMCMC sampler. This section is due to space constraints fairly brief and it is helpful to be familiar with the proof of PG in Andrieu et al. (2010). We start by defining some additional notation. Let $\xi := \{x_t^i\}_{\substack{i=1:N \\ t=1:T}} \bigcup \{a_t^i\}_{\substack{i=1:N \\ t=1:T-1}}$ denote all generated particles and ancestor variables of a (C)SMC sampler. We write ξ_m when referring to the variables of the sampler local to node m. Let the conditional particle trajectory and corresponding ancestor variables for node c_j be denoted by $\{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\}$, with $\mathbf{b}_{c_j} = (\beta_{1,c_j}, \ldots, \beta_{T,c_j})$, $\beta_{T,c_j} = b_j$ and $\beta_{t,c_j} = a_{t,c_j}^{\beta_{t+1,c_j}}$. Let the posterior distribution of the latent variables be denoted by $\pi_T(\mathbf{x}) := p(x_{1:T}|y_{1:T})$ with normalisation constant $Z := p(y_{1:T})$. Finally we note that the SMC and CSMC algorithms induce the respective distributions over the random variables generated by the procedures:

$$q_{\text{SMC}}(\xi) = \prod_{i=1}^{N} q_1(x_1^i) \cdot \prod_{t=2}^{T} \prod_{i=1}^{N} \left[\bar{w}_{t-1}^{a_{t-1}^i} q_t(x_t^i | x_{1:t-1}^{a_{t-1}^i}) \right],$$

$$q_{\text{CSMC}}(\xi \backslash \{\mathbf{x}', \mathbf{b}\} \mid \mathbf{x}', \mathbf{b}) = \prod_{\substack{i=1 \\ i \neq b_1}}^{N} q_1(x_1^i) \cdot \prod_{t=2}^{T} \prod_{\substack{i=1 \\ i \neq b_t}}^{N} \left[\bar{w}_{t-1}^{a_{t-1}^i} q_t(x_t^i | x_{1:t-1}^{a_{t-1}^i}) \right].$$

Note that running Algorithm 2 corresponds to simulating from q_{CSMC} using a fixed choice for the index variables $\mathbf{b} = (N \ldots, N)$. While these indices are used

to facilitate the proof of validity of the proposed method, they have no practical relevance and can thus be set to arbitrary values, as is done in Algorithm 2, in a practical implementation.

Now we are ready to state the main theoretical result.

Theorem 1. *The interacting particle Markov chain Monte Carlo sampler of Algorithm 3 is a partially collapsed Gibbs sampler (Van Dyk and Park, 2008) for the target distribution*

$$\tilde{\pi}(\xi_{1:M}, c_{1:P}, b_{1:P}) = \frac{1}{N^P T\binom{M}{P}} \prod_{\substack{m=1 \\ m \notin c_{1:P}}}^{M} q_{SMC}(\xi_m) \cdot \prod_{j=1}^{P} \pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \mathbf{1}_{c_j \notin c_{1:j-1}} \cdot \prod_{j=1}^{P} q_{CSMC}\left(\xi_{c_j} \backslash \{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\} \mid \mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\right). \tag{9}$$

Proof: See Appendix A at the end of the paper. □

Remark 1. The marginal distribution of $(\mathbf{x}_{c_{1:P}}^{b_{1:P}}, c_{1:P}, b_{1:P})$, with $\mathbf{x}_{c_{1:P}}^{b_{1:P}} = (\mathbf{x}_{c_1}^{b_1}, \ldots, \mathbf{x}_{c_P}^{b_P})$, under (9) is given by

$$\tilde{\pi}\left(\mathbf{x}_{c_{1:P}}^{b_{1:P}}, c_{1:P}, b_{1:P}\right) = \frac{\prod_{j=1}^{P} \pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \mathbf{1}_{c_j \notin c_{1:j-1}}}{N^P T\binom{M}{P}}. \tag{10}$$

This means that each trajectory $\mathbf{x}_{c_j}^{b_j}$ is marginally distributed according to the posterior distribution of interest, π_T. Indeed, the P retained trajectories of iPMCMC will in the limit $R \to \infty$ be independent draws from π_T.

Note that adding a backward or ancestor simulation step can drastically increase mixing when sampling the conditional trajectories $\mathbf{x}'_j[r]$ (Lindsten and Schön, 2013). In the iPMCMC sampler we can replace simulating from the final weights on line 7 by a backward simulation step. Another option for the CSMC nodes is to replace this step by internal ancestor sampling (Lindsten et al., 2014) steps and simulate from the final weights as normal.

3.2 Using All Particles

At each MCMC iteration r, we generate MN full particle trajectories. Using only P of these as in (8) might seem a bit wasteful. We can however make use of all particles to estimate expectations of interest by, for each Gibbs update j, averaging over the possible new values for the conditional node index c_j and corresponding particle index b_j. We can do this by replacing $f(\mathbf{x}'_j[r])$ in (8) by

$$\mathbb{E}_{c_j | c_{1:P \backslash j}}\left[\mathbb{E}_{b_j | c_j}\left[f(\mathbf{x}'_j[r])\right]\right] = \sum_{m=1}^{M} \hat{\zeta}_m^j \sum_{i=1}^{N} \bar{w}_{T,m}^i f(\mathbf{x}_m^i).$$

This procedure is referred to as a Rao-Blackwellization of a statistical estimator and is (in terms of variance) never worse than the original one. We highlight that each $\hat{\zeta}_m^j$, as defined in (6), depends on which indices are sampled earlier in the index reassignment loop. Further details, along with a derivation, are provided in the supplementary material.

3.3 Choosing P

Before jumping into the full details of our experimentation, we quickly consider the choice of P. Intuitively we can think of the independent SMC's as particularly useful if they are selected as the next conditional node. The probability of the event that at least one conditional node switches with an unconditional, is given by

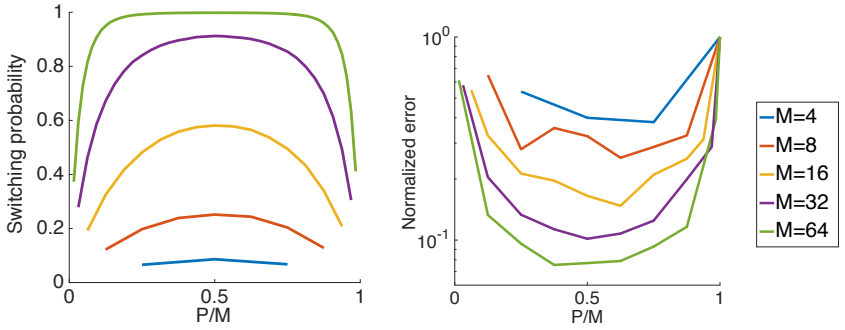
$$\mathbb{P}(\{\text{switch}\}) = 1 - \mathbb{E}\Big[\prod_{j=1}^{P} \frac{\hat{Z}_{c_j}}{\hat{Z}_{c_j} + \sum_{m \notin c_{1:P}}^{M} \hat{Z}_m}\Big]. \tag{11}$$

There exist theoretical and experimental results (Bérard et al., 2014; Doucet et al., 2015; Pitt et al., 2012) that show that the distributions of the normalisation constants are well-approximated by their log-Normal limiting distributions. Now, with σ^2 ($\propto \frac{1}{N}$) being the variance of the (C)SMC estimate, it means we have $\log\left(Z^{-1}\hat{Z}_{c_j}\right) \sim \mathcal{N}(\frac{\sigma^2}{2}, \sigma^2)$ and $\log\left(Z^{-1}\hat{Z}_m\right) \sim \mathcal{N}(-\frac{\sigma^2}{2}, \sigma^2)$, $m \notin c_{1:P}$ at stationarity, where Z is the true normalization constant. Under this assumption, we can accurately estimate the probability (11) for different choices of P an example of which is shown in Figure 1a along with additional analysis in the supplementary material. These provide strong empirical evidence that the switching probability is maximised for $P = M/2$.

In practice we also see that best results are achieved when P makes up roughly half of the nodes, see Figure 1b for performance on the state space model introduced in (12). Note also that the accuracy seems to be fairly robust with respect to the choice of P. Based on these results, we set the value of $P = \frac{M}{2}$ for the rest of our experiments.

4 Experiments

To demonstrate the empirical performance of iPMCMC we report experiments on two state space models. Although both the models considered are Markovian, we emphasise that iPMCMC goes far beyond this and can be applied to arbitrary graphical models. We will focus our comparison on the trivially distributed alternatives, whereby M independent PMCMC samplers are run in parallel–these are PG, particle independent Metropolis-Hastings (PIMH) Andrieu et al. (2010) and the alternate move PG sampler (APG) Holenstein (2009). Comparisons to

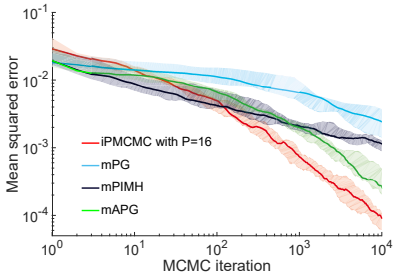


(a) *Limiting log-Normal* ***(b)*** *Gaussian state space model*

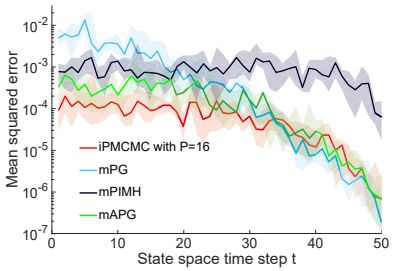
Figure 1: *a) Estimation of switching probability for different choices of P and M assuming the log-Normal limiting distribution for $\hat{Z}_m$ with $\sigma = 3$. b) Median error in mean estimate for different choices of P and M over 10 different synthetic datasets of the linear Gaussian state space model given in* (12) *after 1000 MCMC iterations. Here errors are normalized by the error of a multi-start PG sampler which is a special case of iPMCMC for which $P = M$ (see Section 6).*

other alternatives, including independent SMC, serialized implementations of PG and PIMH, and running a mixture of independent PG and PIMH samplers, are provided in the supplementary material. None outperformed the methods considered here, with the exception of running a serialized PG implementation with an increased number of particles, requiring significant additional memory ($O(MN)$ as opposed to $O(M + N)$).

In PIMH a new particle set is proposed at each MCMC step using an independent SMC sweep, which is then either accepted or rejected using the standard Metropolis-Hastings acceptance ratio. APG interleaves PG steps with PIMH steps in an attempt to overcome the issues caused by path degeneracy in PG. We refer to the trivially distributed versions of these algorithms as multi-start PG, PIMH and APG respectively (mPG, mPIMH and mAPG). We use Rao-Blackwellization, as described in 3.2, to average over all the generated particles for all methods, weighting the independent Markov chains equally for mPG, mPIMH and mAPG. We note that mPG is a special case of iPMCMC for which $P = M$. For simplicity, multinomial resampling was used in the experiments, with the prior transition distribution of the latent variables taken for the proposal. $M = 32$ nodes and $N = 100$ particles were used unless otherwise stated. Initialization of the retained particles for iPMCMC and mPG was done by using standard SMC sweeps.



(a) Convergence in mean for full sequence



(b) Final error in mean for latent marginals

Figure 2: *Mean squared error averaged over all dimensions and steps in the state sequence as a function of MCMC iterations (left) and mean squared error after 10^4 iterations averaged over dimensions as function of position in the state sequence (right) for (12) with 50 time sequences. The solid line shows the median error across the 10 tested synthetic datasets, while the shading shows the upper and lower quartiles. Ground truth was calculated using the Rauch–Tung–Striebel smoother algorithm Rauch et al. (1965).*

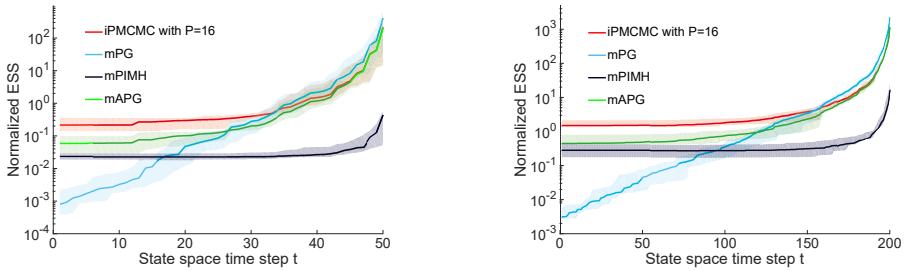
4.1 Linear Gaussian State Space Model

We first consider a linear Gaussian state space model (LGSSM) with 3 dimensional latent states $x_{1:T}$, 20 dimensional observations $y_{1:T}$ and dynamics given by

$$x_1 \sim \mathcal{N}(\mu, V) \tag{12a}$$
$$x_t = \alpha x_{t-1} + \delta_{t-1} \qquad \delta_{t-1} \sim \mathcal{N}(0, \Omega) \tag{12b}$$
$$y_t = \beta x_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \Sigma). \tag{12c}$$

We set $\mu = [0, 1, 1]^T$, $V = 0.1\ \mathbf{I}$, $\Omega = \mathbf{I}$ and $\Sigma = 0.1\ \mathbf{I}$ where $\mathbf{I}$ represents the identity matrix. The constant transition matrix, α, corresponds to successively applying rotations of $\frac{7\pi}{10}$, $\frac{3\pi}{10}$ and $\frac{\pi}{20}$ about the first, second and third dimensions of x_{t-1} respectively followed by a scaling of 0.99 to ensure that the dynamics remain stable. A total of 10 different synthetic datasets of length $T = 50$ were generated by simulating from (12a)–(12c), each with a different emission matrix β generated by sampling each column independently from a symmetric Dirichlet distribution with concentration parameter 0.2.

Figure 2a shows convergence in the estimate of the latent variable means to the ground-truth solution for iPMCMC and the benchmark algorithms as a function of MCMC iterations. It shows that iPMCMC comfortably outperforms the alternatives from around 200 iterations onwards, with only iPMCMC and mAPG demonstrating behaviour consistent with the Monte Carlo convergence rate, suggesting that mPG and mPIMH are still far from the ergodic regime. Figure 2b shows the same errors after 10^4 MCMC iterations as a function of position in state sequence.



(a) *LGSSM* ***(b)*** *NLSSM*

Figure 3: *Normalized effective sample size (NESS) for LGSSM (left) and NLSSM (right).*

This demonstrates that iPMCMC outperformed all the other algorithms for the early stages of the state sequence, for which mPG performed particularly poorly. Toward the end of state sequence, iPMCMC, mPG and mAPG all gave similar performance, whilst that of mPIMH was significantly worse.

4.2 Nonlinear State Space Model

We next consider the one dimensional nonlinear state space model (NLSSM) considered by, among others, Andrieu et al. (2010); Gordon et al. (1993)

$$x_1 \sim \mathcal{N}\left(\mu, v^2\right) \tag{13a}$$

$$x_t = \frac{x_{t-1}}{2} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos\left(1.2t\right) + \delta_{t-1} \tag{13b}$$

$$y_t = \frac{{x_t}^2}{20} + \varepsilon_t \tag{13c}$$

where $\delta_{t-1} \sim \mathcal{N}\left(0, \omega^2\right)$ and $\varepsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$. We set the parameters as $\mu = 0$, $v = \sqrt{5}$, $\omega = \sqrt{10}$ and $\sigma = \sqrt{10}$. Unlike the LGSSM, this model does not have an analytic solution and therefore one must resort to approximate inference methods. Further, the multi-modal nature of the latent space makes full posterior inference over $x_{1:T}$ challenging for long state sequences.

To examine the relative mixing of iPMCMC we calculate an effective sample size (ESS) for different steps in the state sequence. In order to calculate the ESS, we condensed identical samples as done in for example van de Meent et al. (2015). Let

$$u_t^k \quad \in \quad \{x_{t,m}^i[r]\}_{m=1:M}^{i=1:N,r=1:R}, \quad \forall k \quad \in \quad 1\ldots K, \quad t \quad \in \quad 1\ldots T$$

denote the unique samples of x_t generated by all the nodes and sweeps of particular algorithm after R iterations, where K is the total number of unique samples

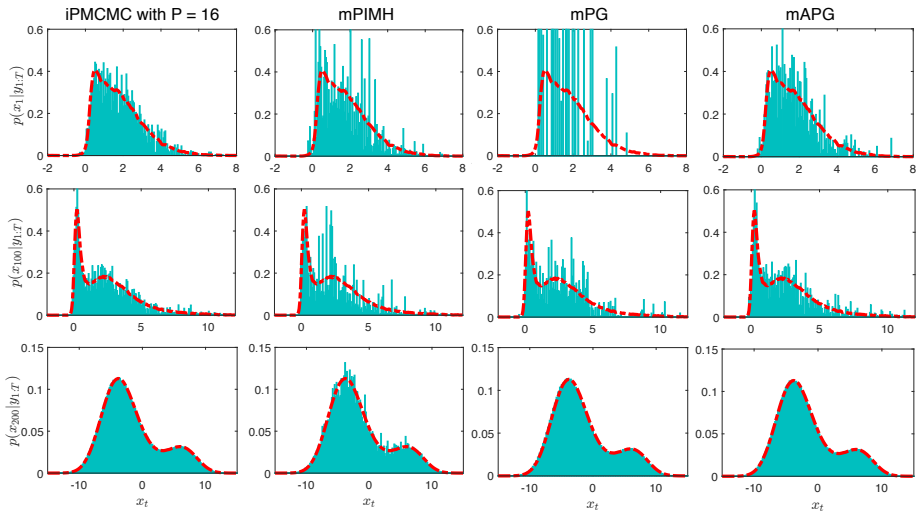


Figure 4: *Histograms of generated samples at* $t = 1, 100,$ *and* 200 *for a single dataset generated from* (13) *with* $T = 200$. *Dashed red line shows an approximate estimate of the ground truth, found by running a kernel density estimator on the combined samples from a small number of independent SMC sweeps, each with* 10^7 *particles.*

generated. The weight assigned to these unique samples, v_t^k, is given by the combined weights of all particles for which x_t takes the value u_t^k:

$$v_t^k = \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{i=1}^{N} \bar{w}_{t,m}^{i,r} \eta_m^r \delta_{x_{t,m}^i[r]}(u_t^k) \tag{14}$$

where $\delta_{x_{t,m}^i[r]}(u_t^k)$ is the Kronecker delta function and η_m^r is a node weight. For iPMCMC the node weight is given by as per the Rao-Blackwellized estimator described in Section 3.2. For mPG and mPIMH, η_m^r is simply $\frac{1}{RM}$, as samples from the different nodes are weighted equally in the absence of interaction. Finally we define the effective sample size as $\mathrm{ESS}_t = \left(\sum_{k=1}^{K} \left(v_t^k\right)^2\right)^{-1}$.

Figure 3 shows the ESS for the LGSSM and NLSSM as a function of position in the state sequence. For this, we omit the samples generated by the initialization step as this SMC sweep is common to all the tested algorithms. We further normalize by the number of MCMC iterations so as to give an idea of the rate at which unique samples are generated. These show that for both models the ESS of iPMCMC, mPG and mAPG is similar towards the end of the space sequence, but that iPMCMC outperforms all the other methods at the early stages. The ESS of mPG was particularly poor at early iterations. PIMH performed poorly throughout, reflecting the very low observed acceptance ratio of around 7.3% on average.

It should be noted that the ESS is not a direct measure of performance for these models. For example, the equal weighting of nodes is likely to make the ESS artificially high for mPG, mPIMH and mAPG, when compared with methods such as iPMCMC that assign a weighting to the nodes at each iteration. To acknowledge this, we also plot histograms for the marginal distributions of a number of different position in the state sequence as shown in Figure 4. These confirm that iPMCMC and mPG have similar performance at the latter state sequence steps, whilst iPMCMC is superior at the earlier stages, with mPG producing almost no more new samples than those from the initialization sweep due to the degeneracy. The performance of PIMH was consistently worse than iPMCMC throughout the state sequence, with even the final step exhibiting noticeable noise.

5 Discussion and Future Work

The iPMCMC sampler overcomes degeneracy issues in PG by allowing the newly sampled particles from SMC nodes to replace the retained particles in CSMC nodes. Our experimental results demonstrate that, for the models considered, this switching in rate is far higher than the rate at which PG generates fully independent samples. Moreover, the results in Figure 1b suggest that the degree of improvement over an mPG sampler with the same total number of nodes increases with the total number of nodes in the pool.

The mAPG sampler performs an accept reject step that compares the marginal likelihood estimate of a single CSMC sweep to that of a single SMC sweep. In the iPMCMC sampler the CSMC estimate of the marginal likelihood is compared to a population sample of SMC estimates, resulting in a higher probability that at least one of the SMC nodes will become a CSMC node.

Since the original PMCMC paper in 2010 there have been several papers studying (Chopin and Singh, 2015; Lindsten et al., 2015) and improving upon the basic PG algorithm. Key contributions to combat the path degeneracy effect are backward simulation (Lindsten and Schön, 2013; Whiteley et al., 2010) and ancestor sampling (Lindsten et al., 2014). These can also be used to improve the iPMCMC method ever further.

Appendix

A Proof of Theorem 1

The proof follows similar ideas as Andrieu et al. (2010). We prove that the interacting particle Markov chain Monte Carlo sampler is in fact a standard partially collapsed Gibbs sampler (Van Dyk and Park, 2008) on an extended space $\Upsilon := \mathsf{X}^{\otimes MTN} \times [N]^{\otimes M(T-1)N} \times [M]^{\otimes P} \times [N]^{\otimes P}$.

Proof: Assume the setup of Section 3. With $\tilde{\pi}(\cdot)$ with as per (9), we will show that the Gibbs sampler on the extended space, Υ, defined as follows

$$\xi_{1:M} \backslash \{\mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}\} \sim \tilde{\pi}(\ \cdot\ | \mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}, c_{1:P}, b_{1:P}), \tag{15a}$$

$$c_j \sim \tilde{\pi}(\ \cdot\ | \xi_{1:M}, c_{1:P\backslash j}), \quad j = 1, \ldots, P, \tag{15b}$$

$$b_j \sim \tilde{\pi}(\ \cdot\ | \xi_{1:M}, c_{1:P}), \quad j = 1, \ldots, P, \tag{15c}$$

is equivivalent to the iPMCMC method in Algorithm 3.

First, the initial step (15a) corresponds to sampling from

$$\tilde{\pi}(\xi_{1:M} \backslash \{\mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}\} | \mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}, c_{1:P}, b_{1:P}) =$$
$$\prod_{\substack{m=1 \\ m \notin c_{1:P}}}^{M} q_{\mathrm{SMC}}(\xi_m) \times \prod_{j=1}^{P} q_{\mathrm{CSMC}}\left(\xi_{c_j} \backslash \{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\} \mid \mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}, c_j, b_j\right).$$

This, excluding the conditional trajectories, just corresponds to steps 3–4 in Algorithm 3, i.e. running P CSMC and $M - P$ SMC algorithms independently.

We continue with a reformulation of (9) which will be usefuly to prove correctness for the other two steps

$$\begin{aligned}
\tilde{\pi}(\xi_{1:M}, c_{1:P}, b_{1:P}) &= \frac{1}{\binom{M}{P}} \prod_{m=1}^{M} q_{\mathrm{SMC}}(\xi_m) \\
&\times \prod_{j=1}^{P} \left[\mathbf{1}_{c_j \notin c_{1:j-1}} \bar{w}_{T,c_j}^{b_j} \pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \times \frac{q_{\mathrm{CSMC}}\left(\xi_{c_j} \backslash \{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\} \mid \mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}, c_j, b_j\right)}{N^T \bar{w}_{T,c_j}^{b_j} q_{\mathrm{SMC}}\left(\xi_{c_j}\right)} \right] \\
&= \frac{1}{\binom{M}{P}} \prod_{m=1}^{M} q_{\mathrm{SMC}}(\xi_m) \prod_{j=1}^{P} \frac{\hat{Z}_{c_j}}{Z} \mathbf{1}_{c_j \notin c_{1:j-1}} \bar{w}_{T,c_j}^{b_j}.
\end{aligned} \tag{16}$$

Furthermore, we note that by marginalising (collapsing) the above reformulation, i.e. (16), over $b_{1:P}$ we get

$$\tilde{\pi}(\xi_{1:M}, c_{1:P}) = \frac{1}{\binom{M}{P}} \prod_{m=1}^{M} q_{\mathrm{SMC}}(\xi_m) \prod_{j=1}^{P} \frac{\hat{Z}_{c_j}}{Z} \mathbf{1}_{c_j \notin c_{1:j-1}}.$$

From this it is easy to see that $\tilde{\pi}(c_j | \xi_{1:M}, c_{1:P\backslash j}) = \hat{\zeta}_{c_j}^{j}$, which corresponds to sampling the conditional node indices, i.e. step 6 in Algorithm 3. Finally, from (16) we can see that simulating $b_{1:P}$ can be done independently as follows

$$\tilde{\pi}(b_{1:P} | \xi_{1:M}, c_{1:P}) = \frac{\tilde{\pi}(b_{1:P}, \xi_{1:M}, c_{1:P})}{\tilde{\pi}(\xi_{1:M}, c_{1:P})} = \prod_{j=1}^{P} \bar{w}_{T,c_j}^{b_j}.$$

This corresponds to step 7 in the iPMCMC sampler, Algorithm 3. So the procedure defined by (15) is a partially collapsed Gibbs sampler, derived from (9), and we have shown that it is exactly equal to the iPMCMC sampler described in Algorithm 3. □

Acknowledgments

Tom Rainforth is supported by a BP industrial grant. Christian A. Naesseth is supported by CADICS, a Linnaeus Center, funded by the Swedish Research Council (VR). Fredrik Lindsten is supported by the project *Learning of complex dynamical systems* (Contract number: 637-2014-466) also funded by the Swedish Research Council. Frank Wood is supported under DARPA PPAML through the U.S. AFRL under Cooperative Agreement number FA8750-14-2-0006, Sub Award number 61160290-111668.

Bibliography

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. ISSN 1467-9868.

Jean Bérard, Pierre Del Moral, and Arnaud Doucet. A lognormal central limit theorem for particle approximations of normalizing constants. *Electronic Journal of Probability*, 19(94):1–28, 2014.

Nicolas Chopin and Sumeetpal S. Singh. On particle Gibbs sampling. *Bernoulli*, 21(3):1855–1883, 08 2015. doi: 10.3150/14-BEJ629.

Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2001.

Arnaud Doucet, Michael Pitt, George Deligiannidis, and Robert Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, page asu075, 2015.

Richard G. Everitt. Bayesian parameter estimation for latent Markov random fields and social networks. *Journal of Computational and Graphical Statistics*, 21(4):940–960, 2012.

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

Roman Holenstein. *Particle Markov chain Monte Carlo*. PhD thesis, The University Of British Columbia (Vancouver, 2009.

Jonathan H. Huggins and Daniel M. Roy. Convergence of sequential Monte Carlo-based sampling methods. *ArXiv e-prints, arXiv:1503.00966v1*, March 2015.

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. *Journal of Machine Learning Research*, 15:2145–2184, june 2014.

Fredrik Lindsten, Randal Douc, and Eric Moulines. Uniform ergodicity of the particle Gibbs sampler. *Scandinavian Journal of Statistics*, 42(3):775–797, 2015.

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., 2014.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B Schön. Nested sequential Monte Carlo methods. In *The 32nd International Conference on Machine Learning*, volume 37 of *JMLR W&CP*, pages 1292–1301, Lille, France, jul 2015.

Michael K Pitt, Ralph dos Santos Silva, Paolo Giordani, and Robert Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

Herbert E Rauch, CT Striebel, and F Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Nilesh Tripuraneni, Shixiang Gu, Hong Ge, and Zoubin Ghahramani. Particle Gibbs for infinite hidden Markov Models. In *Advances in Neural Information Processing Systems 28*, pages 2386–2394. Curran Associates, Inc., 2015.

Isabel Valera, Fran Francisco, Lennart Svensson, and Fernando Perez-Cruz. Infinite factorial dynamical model. In *Advances in Neural Information Processing Systems 28*, pages 1657–1665. Curran Associates, Inc., 2015.

Jan-Willem van de Meent, Hongseok Yang, Vikash Mansinghka, and Frank Wood. Particle Gibbs with ancestor sampling for probabilistic programs. In *Proceedings of the 18th International conference on Artificial Intelligence and Statistics*, pages 986–994, 2015.

David A Van Dyk and Taeyoung Park. Partially collapsed Gibbs samplers: Theory and methods. *Journal of the American Statistical Association*, 103(482):790–796, 2008.

Nick Whiteley, Christophe Andrieu, and Arnaud Doucet. Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods. *ArXiv e-prints, arXiv:1011.2437*, 2010.

Nick Whiteley, Anthony Lee, and Kari Heine. On the role of interaction in sequential Monte Carlo algorithms. *Bernoulli*, 22(1):494–529, 02 2016.

Frank Wood, Jan Willem van de Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *Proceedings of the 17th International conference on Artificial Intelligence and Statistics*, pages 2–46, 2014.

Paper G

Reparameterization Gradients through Acceptance-Rejection Sampling Algorithms

Authors: Christian A. Naesseth, Francisco J. R. Ruiz, Scott W. Linderman, and David M. Blei

Edited version of the paper:

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.

This paper has been formatted to fit this layout.

Reparameterization Gradients through Acceptance-Rejection Sampling Algorithms

Christian A. Naesseth*, Francisco J. R. Ruiz†, Scott W. Linderman†, and David M. Blei†

* Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

† Dept. of Computer Science
Columbia University
New York City, United States of America
{f.ruiz,scott.linderman,david.blei}@columbia.edu

Abstract

Variational inference using the reparameterization trick has enabled large-scale approximate Bayesian inference in complex probabilistic models, leveraging stochastic optimization to sidestep intractable expectations. The reparameterization trick is applicable when we can simulate a random variable by applying a differentiable deterministic function on an auxiliary random variable whose distribution is fixed. For many distributions of interest (such as the gamma or Dirichlet), simulation of random variables relies on acceptance-rejection sampling. The discontinuity introduced by the accept–reject step means that standard reparameterization tricks are not applicable. We propose a new method that lets us leverage reparameterization gradients even when variables are outputs of a acceptance-rejection sampling algorithm. Our approach enables reparameterization on a larger class of variational distributions. In several studies of real and synthetic data, we show that the variance of the estimator of the gradient is significantly lower than other state-of-the-art methods. This leads to faster convergence of stochastic gradient variational inference.

1 Introduction

Variational inference (Hinton and van Camp, 1993; Jordan et al., 1999; Waterhouse et al., 1996) underlies many recent advances in large scale probabilistic modeling. It has enabled sophisticated modeling of complex domains such as im-

ages (Kingma and Welling, 2014) and text (Hoffman et al., 2013). By definition, the success of variational approaches depends on our ability to (i) formulate a flexible parametric family of distributions; and (ii) optimize the parameters to find the member of this family that most closely approximates the true posterior. These two criteria are at odds—the more flexible the family, the more challenging the optimization problem. In this paper, we present a novel method that enables more efficient optimization for a large class of variational distributions, namely, for distributions that we can efficiently simulate by acceptance-rejection sampling, or rejection sampling for short.

For complex models, the variational parameters can be optimized by stochastic gradient ascent on the *evidence lower bound* (ELBO), a lower bound on the marginal likelihood of the data. There are two primary means of estimating the gradient of the ELBO: the score function estimator (Mnih and Gregor, 2014; Paisley et al., 2012; Ranganath et al., 2014) and the reparameterization trick (Bonnet, 1964; Kingma and Welling, 2014; Price, 1958; Rezende et al., 2014), both of which rely on Monte Carlo sampling. While the reparameterization trick often yields lower variance estimates and therefore leads to more efficient optimization, this approach has been limited in scope to a few variational families (typically Gaussians). Indeed, some lines of research have already tried to address this limitation (Knowles, 2015; Ruiz et al., 2016).

There are two requirements to apply the reparameterization trick. The first is that the random variable can be obtained through a transformation of a simple random variable, such as a uniform or standard normal; the second is that the transformation be differentiable. In this paper, we observe that all random variables we simulate on our computers are ultimately transformations of uniforms, often followed by accept-reject steps. So if the transformations are differentiable then we can use these existing simulation algorithms to expand the scope of the reparameterization trick.

Thus, we show how to use existing rejection samplers to develop stochastic gradients of variational parameters. In short, each rejection sampler uses a highly-tuned transformation that is well-suited for its distribution. We can construct new reparameterization gradients by "removing the lid" from these black boxes, applying 65+ years of research on transformations (Devroye, 1986; von Neumann, 1951) to variational inference. We demonstrate that this broadens the scope of variational models amenable to efficient inference and provides lower-variance estimates of the gradient compared to state-of-the-art approaches.

We first review variational inference, with a focus on stochastic gradient methods. We then present our key contribution, rejection sampling variational inference (RSVI), showing how to use efficient rejection samplers to produce low-variance stochastic gradients of the variational objective. We study two concrete examples, analyzing rejection samplers for the gamma and Dirichlet to produce new reparameterization gradients for their corresponding variational factors. Finally, we analyze two datasets with a deep exponential family (DEF) (Ranganath et al., 2015), comparing RSVI to the state of the art. We found that RSVI achieves a sig-

nificant reduction in variance and faster convergence of the ELBO. Code for all experiments is provided at github.com/blei-lab/ars-reparameterization.

2 Variational Inference

Let $p(x, z)$ be a probabilistic model, i.e., a joint probability distribution of *data* x and *latent* (unobserved) variables z. In Bayesian inference, we are interested in the posterior distribution $p(z|x) = \frac{p(x,z)}{p(x)}$. For most models, the posterior distribution is analytically intractable and we have to use an approximation, such as Monte Carlo methods or variational inference. In this paper, we focus on variational inference.

In variational inference, we approximate the posterior with a *variational family* of distributions $q(z\,;\theta)$, parameterized by θ. Typically, we choose the *variational parameters* θ that minimize the KL divergence between $q(z\,;\theta)$ and $p(z|x)$. This minimization is equivalent to maximizing the ELBO (Jordan et al., 1999), defined as

$$\begin{aligned} \mathcal{L}(\theta) &= \mathbb{E}_{q(z\,;\theta)}\left[f(z)\right] + \mathbb{H}[q(z\,;\theta)], \\ f(z) &:= \log p(x, z), \\ \mathbb{H}[q(z\,;\theta)] &:= \mathbb{E}_{q(z\,;\theta)}[-\log q(z\,;\theta)]. \end{aligned} \tag{1}$$

When the model and variational family satisfy conjugacy requirements, we can use coordinate ascent to find a local optimum of the ELBO (Blei et al., 2016). If the conjugacy requirements are not satisfied, a common approach is to build a Monte Carlo estimator of the gradient of the ELBO (Kingma and Welling, 2014; Mnih and Gregor, 2014; Paisley et al., 2012; Ranganath et al., 2014; Salimans and Knowles, 2013). This results in a stochastic optimization procedure, where different Monte Carlo estimators of the gradient amount to different algorithms. We review below two common estimators: the score function estimator and the reparameterization trick.[1]

Score function estimator. The score function estimator, also known as the log-derivative trick or REINFORCE (Glynn, 1990; Williams, 1992), is a general way to estimate the gradient of the ELBO (Mnih and Gregor, 2014; Paisley et al., 2012; Ranganath et al., 2014). The score function estimator expresses the gradient as an expectation with respect to $q(z\,;\theta)$:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{q(z\,;\theta)}[f(z)\nabla_\theta \log q(z\,;\theta)] + \nabla_\theta \mathbb{H}[q(z\,;\theta)].$$

We then form Monte Carlo estimates by approximating the expectation with independent samples from the variational distribution. Though it is very general,

[1] In this paper, we assume for simplicity that the gradient of the entropy $\nabla_\theta \mathbb{H}[q(z\,;\theta)]$ is available analytically. The method that we propose in Section 3 can be easily extended to handle non-analytical entropy terms. Indeed, the resulting estimator of the gradient may have lower variance when the analytic gradient of the entropy is replaced by its Monte Carlo estimate. Here we do not explore that.

the score function estimator typically suffers from high variance. In practice we also need to apply variance reduction techniques such as Rao-Blackwellization (Casella and Robert, 1996) and control variates (Robert and Casella, 2004).

Reparameterization trick. The reparameterization trick (Bonnet, 1964; Kingma and Welling, 2014; Price, 1958; Salimans and Knowles, 2013) results in a lower variance estimator compared to the score function, but it is not as generally applicable. It requires that: (i) the latent variables z are continuous; and (ii) we can simulate from $q(z\,;\theta)$ as follows,

$$z = h(\varepsilon, \theta), \qquad \text{with } \varepsilon \sim s(\varepsilon). \tag{2}$$

Here, $s(\varepsilon)$ is a distribution that does not depend on the variational parameters; it is typically a standard normal or a standard uniform. Further, $h(\varepsilon, \theta)$ must be differentiable with respect to θ. In statistics, this is known as a non-central parameterization and has been shown to be helpful in, e.g., Markov chain Monte Carlo methods (Papaspiliopoulos et al., 2003).

Using (2), we can move the derivative inside the expectation and rewrite the gradient of the ELBO as

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{s(\varepsilon)}\left[\nabla_z f(h(\varepsilon, \theta))\nabla_\theta h(\varepsilon, \theta)\right] + \nabla_\theta \mathbb{H}[q(z\,;\theta)].$$

Empirically, the reparameterization trick has been shown to be beneficial over direct Monte Carlo estimation of the gradient using the score fuction estimator (Fan et al., 2015; Kingma and Welling, 2014; Salimans and Knowles, 2013; Titsias and Lázaro-Gredilla, 2014). Unfortunately, many distributions commonly used in variational inference, such as gamma or Dirichlet, are not amenable to standard reparameterization because samples are generated using a rejection sampler (Robert and Casella, 2004; von Neumann, 1951), introducing discontinuities to the mapping. We next show that taking a novel view of the acceptance-rejection sampler lets us perform exact reparameterization.

3 Reparameterizing the Acceptance-Rejection Sampler

The basic idea behind reparameterization is to rewrite simulation from a complex distribution as a deterministic mapping of its parameters and a set of simpler random variables. We can view the rejection sampler as a complicated deterministic mapping of a (random) number of simple random variables such as uniforms and normals. This makes it tempting to take the standard reparameterization approach when we consider random variables generated by rejection samplers. However, this mapping is in general not continuous, and thus moving the derivative inside the expectation and using direct automatic differentiation would not necessarily give the correct answer.

Our insight is that we can overcome this problem by instead considering only the marginal over the accepted sample, analytically integrating out the accept-reject

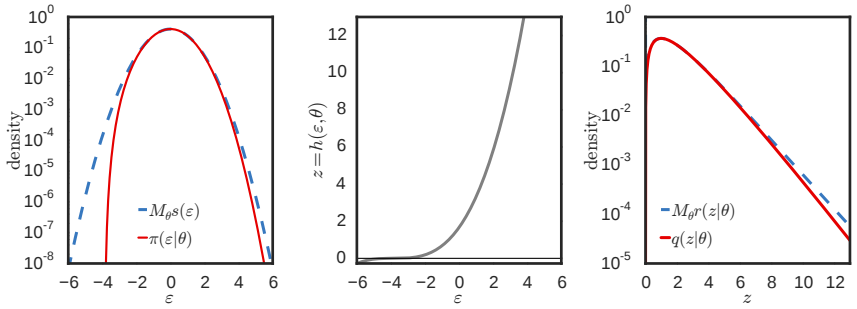


Figure 1: *Example of a reparameterized rejection sampler for $q(z\,;\theta) = \text{Gam}(\theta, 1)$, shown here with $\theta = 2$. We use the rejection sampling algorithm of Marsaglia and Tsang (2000), which is based on a non-linear transformation $h(\varepsilon, \theta)$ of a standard normal $\varepsilon \sim \mathcal{N}(0, 1)$ (c.f. Eq. 10), and has acceptance probability of 0.98 for $\theta = 2$. The marginal density of the accepted value of ε (integrating out the acceptance variables, $u_{1:i}$) is given by $\pi(\varepsilon\,;\theta)$. We compute unbiased estimates of the gradient of the ELBO (6) via Monte Carlo, using Algorithm 1 to rejection sample $\varepsilon \sim \pi(\varepsilon\,;\theta)$. By reparameterizing in terms of ε, we obtain a low-variance estimator of the gradient for challenging variational distributions.*

variable. Thus, the mapping comes from the proposal step. This is continuous under mild assumptions, enabling us to greatly extend the class of variational families amenable to reparameterization.

We first review rejection sampling and present the reparameterized rejection sampler. Next we show how to use it to calculate low-variance gradients of the ELBO. Finally, we present the complete stochastic optimization for variational inference, RSVI.

3.1 Reparameterized Rejection Sampling

Acceptance-Rejection sampling is a powerful way of simulating random variables from complex distributions whose inverse cumulative distribution functions are not available or are too expensive to evaluate (Devroye, 1986; Robert and Casella, 2004). We consider an alternative view of rejection sampling in which we explicitly make use of the reparameterization trick. This view of the rejection sampler enables our variational inference algorithm in Section 3.2.

To generate samples from a distribution $q(z\,;\theta)$ using rejection sampling, we first sample from a *proposal distribution* $r(z\,;\theta)$ such that $q(z\,;\theta) \leq M_\theta r(z\,;\theta)$ for some $M_\theta < \infty$. In our version of the rejection sampler, we assume that the proposal distribution is reparameterizable, i.e., that generating $z \sim r(z\,;\theta)$ is equivalent to generating $\varepsilon \sim s(\varepsilon)$ (where $s(\varepsilon)$ does not depend on θ) and then setting $z = h(\varepsilon, \theta)$ for a differentiable function $h(\varepsilon, \theta)$. We then accept the sample with

Algorithm 1: Reparameterized Rejection Sampling

Input: target $q(z\,;\theta)$, proposal $r(z\,;\theta)$, and constant M_θ, with $q(z\,;\theta) \leq M_\theta r(z\,;\theta)$
Output: ε such that $h(\varepsilon,\theta) \sim q(z\,;\theta)$

$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  Propose $\varepsilon_i \sim s(\varepsilon)$
  Simulate $u_i \sim \mathcal{U}[0,1]$
until $u_i < \frac{q(h(\varepsilon_i,\theta)\,;\theta)}{M_\theta r(h(\varepsilon_i,\theta)\,;\theta)}$
return ε_i

probability $\min\left\{1, \frac{q(h(\varepsilon,\theta)\,;\theta)}{M_\theta r(h(\varepsilon,\theta)\,;\theta)}\right\}$; otherwise, we reject the sample and repeat the process. We illustrate this in Figure 1 and provide a summary of the method in Algorithm 1, where we consider the output to be the (accepted) variable ε, instead of z.

The ability to simulate from $r(z\,;\theta)$ by a reparameterization through a differentiable $h(\varepsilon,\theta)$ is not needed for the rejection sampler to be valid. However, this is indeed the case for the rejection sampler of many common distributions.

3.2 The Reparameterized Rejection Sampler in Variational Inference

We now use reparameterized rejection sampling to develop a novel Monte Carlo estimator of the gradient of the ELBO. We first rewrite the ELBO in (1) as an expectation in terms of the transformed variable ε,

$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{q(z\,;\theta)}\left[f(z)\right] + \mathbb{H}[q(z\,;\theta)] \\ &= \mathbb{E}_{\pi(\varepsilon\,;\theta)}\left[f\left(h(\varepsilon,\theta)\right)\right] + \mathbb{H}[q(z\,;\theta)].\end{aligned} \tag{3}$$

In this expectation, $\pi(\varepsilon\,;\theta)$ is the distribution of the *accepted sample* ε in Algorithm 1. We construct it by marginalizing over the auxiliary uniform variable u,

$$\begin{aligned}\pi(\varepsilon\,;\theta) &= \int \pi(\varepsilon, u\,;\theta)\mathrm{d}u = \int M_\theta s(\varepsilon)\mathbf{1}\left[0 < u < \frac{q\left(h(\varepsilon,\theta)\,;\theta\right)}{M_\theta r\left(h(\varepsilon,\theta)\,;\theta\right)}\right]\mathrm{d}u \\ &= s(\varepsilon)\frac{q\left(h(\varepsilon,\theta)\,;\theta\right)}{r\left(h(\varepsilon,\theta)\,;\theta\right)},\end{aligned} \tag{4}$$

where $\mathbf{1}[x \in A]$ is the indicator function, and recall that M_θ is a constant used in the rejection sampler. This can be seen by the algorithmic definition of the

rejection sampler, where we propose values $\varepsilon \sim s(\varepsilon)$ and $u \sim \mathcal{U}[0,1]$ until acceptance, i.e., until $u < \frac{q(h(\varepsilon,\theta);\theta)}{M_\theta r(h(\varepsilon,\theta);\theta)}$. Eq. 3 follows intuitively, but we formalize it in Proposition 1.

Proposition 1. *Let f be any measurable function, and $\varepsilon \sim \pi(\varepsilon;\theta)$, defined by* (4) *(and implicitly by Algorithm 1). Then*

$$\mathbb{E}_{\pi(\varepsilon;\theta)}\left[f\left(h(\varepsilon,\theta)\right)\right] = \int f(z)q(z;\theta)dz.$$

Proof: Using the definition of $\pi(\varepsilon;\theta)$,

$$\begin{aligned}\mathbb{E}_{\pi(\varepsilon;\theta)}\left[f\left(h(\varepsilon,\theta)\right)\right] &= \int f\left(h(\varepsilon,\theta)\right) s(\varepsilon)\frac{q\left(h(\varepsilon,\theta);\theta\right)}{r\left(h(\varepsilon,\theta);\theta\right)}\mathrm{d}\varepsilon \\ &= \int f(z)r(z;\theta)\frac{q(z;\theta)}{r(z;\theta)}\mathrm{d}z = \int f(z)q(z;\theta)\mathrm{d}z,\end{aligned}$$

where the second to last equality follows because $h(\varepsilon,\theta)$, $\varepsilon \sim s(\varepsilon)$ is a reparameterization of $r(z;\theta)$. □

We can now compute the gradient of $\mathbb{E}_{q(z;\theta)}[f(z)]$ based on Eq. 3,

$$\begin{aligned}&\nabla_\theta \mathbb{E}_{q(z;\theta)}[f(z)] = \nabla_\theta \mathbb{E}_{\pi(\varepsilon;\theta)}[f(h(\varepsilon,\theta))] \\ &= \underbrace{\mathbb{E}_{\pi(\varepsilon;\theta)}[\nabla_\theta f(h(\varepsilon,\theta))]}_{=:g_{\text{rep}}} + \underbrace{\mathbb{E}_{\pi(\varepsilon;\theta)}\left[f(h(\varepsilon,\theta))\nabla_\theta \log \frac{q(h(\varepsilon,\theta);\theta)}{r(h(\varepsilon,\theta);\theta)}\right]}_{=:g_{\text{cor}}},\end{aligned} \tag{5}$$

where we have used the log-derivative trick and rewritten the integrals as expectations with respect to $\pi(\varepsilon;\theta)$ (see the supplement for all details.) We define g_{rep} as the reparameterization term, which takes advantage of gradients with respect to the model and its latent variables; we define g_{cor} as a correction term that accounts for *not* using $r(z;\theta) \equiv q(z;\theta)$.

Using (5), the gradient of the ELBO in (1) can be written as

$$\nabla_\theta \mathcal{L}(\theta) = g_{\text{rep}} + g_{\text{cor}} + \nabla_\theta \mathbb{H}[q(z;\theta)], \tag{6}$$

and thus we can build an unbiased one-sample Monte Carlo estimator $\hat{g} \approx \nabla_\theta \mathcal{L}(\theta)$ as

$$\begin{aligned}\hat{g} &:= \hat{g}_{\text{rep}} + \hat{g}_{\text{cor}} + \nabla_\theta \mathbb{H}[q(z;\theta)], \\ \hat{g}_{\text{rep}} &= \nabla_z f(z)\big|_{z=h(\varepsilon,\theta)} \nabla_\theta h(\varepsilon,\theta) \\ \hat{g}_{\text{cor}} &= f(h(\varepsilon,\theta))\nabla_\theta \log \frac{q(h(\varepsilon,\theta);\theta)}{r(h(\varepsilon,\theta);\theta)},\end{aligned} \tag{7}$$

where ε is a sample generated using Algorithm 1. Of course, one could generate more samples of ε and average, but we have found a single sample to suffice in practice.

Note if $h(\varepsilon, \theta)$ is invertible in ε then we can simplify the evaluation of the gradient of the log-ratio in g_{cor},

$$\nabla_\theta \log \frac{q(h(\varepsilon,\theta);\theta)}{r(h(\varepsilon,\theta);\theta)} = \nabla_\theta \log q(h(\varepsilon,\theta);\theta) + \nabla_\theta \log \left|\frac{dh}{d\varepsilon}(\varepsilon,\theta)\right|. \tag{8}$$

See the supplementary material for details.

Alternatively, we could rewrite the gradient as an expectation with respect to $s(\varepsilon)$ (this is an intermediate step in the derivation shown in the supplement),

$$\nabla_\theta \mathbb{E}_{q(z;\theta)}[f(z)] = \mathbb{E}_{s(\varepsilon)}\left[\frac{q\left(h(\varepsilon,\theta);\theta\right)}{r\left(h(\varepsilon,\theta);\theta\right)}\nabla_\theta f\left(h(\varepsilon,\theta)\right)\right] + \\ + \mathbb{E}_{s(\varepsilon)}\left[\frac{q\left(h(\varepsilon,\theta);\theta\right)}{r\left(h(\varepsilon,\theta);\theta\right)} f\left(h(\varepsilon,\theta)\right)\nabla_\theta \log \frac{q\left(h(\varepsilon,\theta);\theta\right)}{r\left(h(\varepsilon,\theta);\theta\right)}\right],$$

and build an importance sampling-based Monte Carlo estimator, in which the importance weights would be $q\left(h(\varepsilon,\theta);\theta\right)/r\left(h(\varepsilon,\theta);\theta\right)$. However, we would expect this approach to be beneficial for low-dimensional problems only, since for high-dimensional z the variance of the importance weights would be too high.

Algorithm 2: Rejection Sampling Variational Inference

Input: Data x, model $p(x,z)$, variational family $q(z;\theta)$
Output: Variational parameters θ^*

1: **repeat**
2: Run Algorithm 1 for θ^n to obtain a sample ε
3: Estimate the gradient $\hat{g}^n$ at $\theta = \theta^n$ (Eq. 7)
4: Calculate the stepsize ρ^n (Eq. 10)
5: Update $\theta^{n+1} = \theta^n + \rho^n \hat{g}^n$
6: **until convergence**

3.3 Full Algorithm

We now describe the full variational algorithm based on reparameterizing the rejection sampler. In Section 5 we give concrete examples of how to reparameterize common variational families.

We make use of Eq. 6 to obtain a Monte Carlo estimator of the gradient of the ELBO. We use this estimate to take stochastic gradient steps. We use the stepsize sequence ρ^n proposed by Kucukelbir et al. (2016) (also used by Ruiz et al. (2016)), which combines RMSPROP (Tieleman and Hinton, 2012) and Adagrad (Duchi et al., 2011). It is

$$\begin{aligned} \rho^n &= \eta \cdot n^{-1/2+\delta} \cdot \left(1 + \sqrt{s^n}\right)^{-1}, \\ s^n &= t\,(\hat{g}^n)^2 + (1-t)s^{n-1}, \end{aligned} \tag{9}$$

where n is the iteration number. We set $\delta = 10^{-16}$ and $t = 0.1$, and we try different values for η. (When θ is a vector, the operations above are element-wise.)

We summarize the full method in Algorithm 2. We refer to our method as RSVI.

4 Related Work

The reparameterization trick has also been used in automatic differentiation variational inference (ADVI) (Kucukelbir et al., 2015, 2016). ADVI applies a transformation to the random variables such that their support is on the reals and then places a Gaussian variational posterior approximation over the transformed variable ε. In this way, ADVI allows for standard reparameterization, but it cannot fit gamma or Dirichlet variational posteriors, for example. Thus, ADVI struggles to approximate probability densities with singularities, as noted by Ruiz et al. (2016). In contrast, our approach allows us to apply the reparameterization trick on a wider class of variational distributions, which may be more appropriate when the exact posterior exhibits sparsity.

In the literature, we can find other lines of research that focus on extending the reparameterization gradient to other distributions. For the gamma distribution, Knowles (2015) proposed a method based on approximations of the inverse cumulative density function; however, this approach is limited only to the gamma distribution and it involves expensive computations. For general expectations, Schulman et al. (2015) expressed the gradient as a sum of a reparameterization term and a correction term to automatically estimate the gradient in the context of stochastic computation graphs. However, it is not possible to directly apply it to variational inference with acceptance-rejection sampling. This is due to discontinuities in the accept–reject step and the fact that a rejection sampler produces a *random number* of random variables. Recently, another line of work has focused on applying reparameterization to discrete latent variable models (Jang et al., 2017; Maddison et al., 2017) through a continuous relaxation of the discrete space.

The generalized reparameterization (G-REP) method (Ruiz et al., 2016) exploits the decomposition of the gradient as $g_{\text{rep}} + g_{\text{cor}}$ by applying a transformation based on standardization of the sufficient statistics of z. Our approach differs from G-REP: instead of searching for a transformation of z that makes the distribution of ε weakly dependent on the variational parameters (namely, standardization), we do the opposite by choosing a transformation of a simple random variable ε such that the distribution of $z = h(\varepsilon, \theta)$ is *almost* equal to $q(z\,;\theta)$. For that, we reuse the transformations typically used in rejection sampling. Rather than having to derive a new transformation for each variational distribution, we leverage decades of research on transformations in the rejection sampling literature (Devroye, 1986). In rejection sampling, these transformations (and the distributions of ε) are chosen so that they have high acceptance probability, which

means we should expect to obtain $g_{\text{cor}} \approx 0$ with RSVI. In Sections 5 and 6 we compare RSVI with G-REP and show that it exhibits significantly lower variance, thus leading to faster convergence of the inference algorithm.

Finally, another line of research in non-conjugate variational inference aims at developing more expressive variational families (Maaløe et al., 2016; Ranganath et al., 2016; Salimans et al., 2015; Tran et al., 2016). RSVI can extend the reparameterization trick to these methods as well, whenever rejection sampling is used to generate the random variables.

5 Examples of Acceptance-Rejection Reparameterization

As two examples, we study rejection sampling and reparameterization of two well-known distributions: the gamma and Dirichlet. These have been widely used as variational families for approximate Bayesian inference. We emphasize that RSVI is not limited to these two cases, it applies to any variational family $q(z\,;\theta)$ for which a reparameterizable rejection sampler exists. We provide other examples in the supplement.

5.1 Gamma Distribution

One of the most widely used rejection sampler is for the gamma distribution. Indeed, the gamma distribution is also used in practice to generate e.g. beta, Dirichlet, and Student's t-distributed random variables. The gamma distribution, $\text{Gam}(\alpha, \beta)$, is defined by its shape α and rate β.

For $\text{Gam}(\alpha, 1)$ with $\alpha \geq 1$, Marsaglia and Tsang (2000) developed an efficient rejection sampler. It uses a truncated version of the following reparameterization

$$z = h_{\text{Gam}}(\varepsilon, \alpha) := \left(\alpha - \frac{1}{3}\right)\left(1 + \frac{\varepsilon}{\sqrt{9\alpha - 3}}\right)^3, \qquad (10)$$
$$\varepsilon \sim s(\varepsilon) := \mathcal{N}(0, 1).$$

When $\beta \neq 1$, we divide z by the rate β and obtain a sample distributed as $\text{Gam}(\alpha, \beta)$. The acceptance probability is very high: it exceeds 0.95 and 0.98 for $\alpha = 1$ and $\alpha = 2$, respectively. In fact, as $\alpha \to \infty$ we have that $\pi(\varepsilon\,;\theta) \to s(\varepsilon)$, which means that the acceptance probability approaches 1. Figure 1 illustrates the involved functions and distributions for shape $\alpha = 2$.

For $\alpha < 1$, we observe that $z = u^{1/\alpha}\tilde{z}$ is distributed as $\text{Gam}(\alpha, \beta)$ for $\tilde{z} \sim \text{Gam}(\alpha + 1, \beta)$ and $u \sim \mathcal{U}[0, 1]$ (Devroye, 1986; Stuart, 1962), and apply the rejection sampler above for $\tilde{z}$.

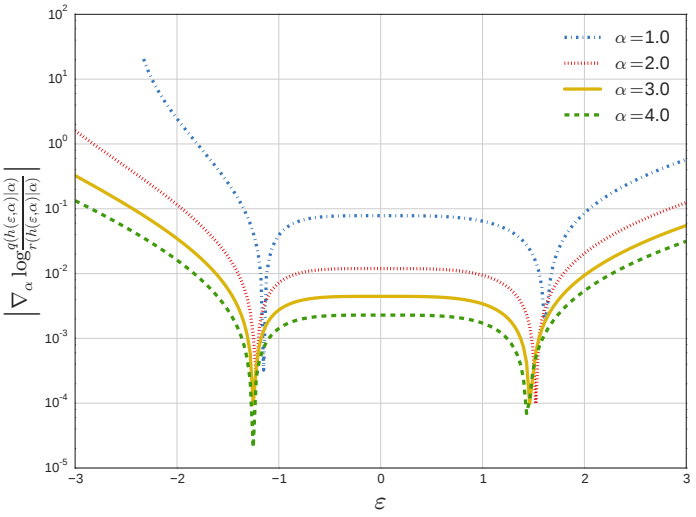


Figure 2: *The correction term of RSVI, and as a result the gradient variance, decreases with increasing shape α. We plot absolute value of the gradient of the log-ratio between the target (gamma) and proposal distributions as a function of ε.*

We now study the quality of the transformation in (10) for different values of the shape parameter α. Since $\pi(\varepsilon\,;\theta) \to s(\varepsilon)$ as $\alpha \to \infty$, we should expect the correction term g_{cor} to decrease with α. We show that in Figure 2, where we plot the log-ratio (8) from the correction term as a function of ε for four values of α. We additionally show in Figure 3 that the distribution $\pi(\varepsilon\,;\theta)$ converges to $s(\varepsilon)$ (a standard normal) as α increases. For large α, $\pi(\varepsilon\,;\theta) \approx s(\varepsilon)$ and the acceptance probability of the rejection sampler approaches 1, which makes the correction term negligible. In Figure 3, we also show that $\pi(\varepsilon\,;\theta)$ converges faster to a standard normal than the standardization procedure used in G-REP. We exploit this property—that performance improves with α—to artificially increase the shape for any gamma distribution. We now explain this trick, which we call *shape augmentation*.

Shape augmentation. Here we show how to exploit the fact that the rejection sampler improves for increasing shape α. We make repeated use of the trick above, using uniform variables, to control the value of α that goes into the rejection sampler. That is, to compute the ELBO for a $\text{Gam}(\alpha, 1)$ distribution, we can first express the random variable as $z = \tilde{z}\prod_{i=1}^{B} u_i^{\frac{1}{\alpha+i-1}}$ (for some positive integer B), $\tilde{z} \sim \text{Gam}(\alpha + B, 1)$ and $u_i \overset{\text{i.i.d.}}{\sim} \mathcal{U}[0,1]$. This can be proved by induction, since $\tilde{z}u_B^{\frac{1}{\alpha+B-1}} \sim \text{Gam}(\alpha + B - 1, 1)$, $\tilde{z}u_B^{\frac{1}{\alpha+B-1}} u_{B-1}^{\frac{1}{\alpha+B-2}} \sim \text{Gam}(\alpha + B - 2, 1)$, etc. Hence, we can apply the rejection sampling framework for $\tilde{z} \sim \text{Gam}(\alpha + B, 1)$ instead of the original z. We study the effect of shape augmentation on the variance in Section 5.2.

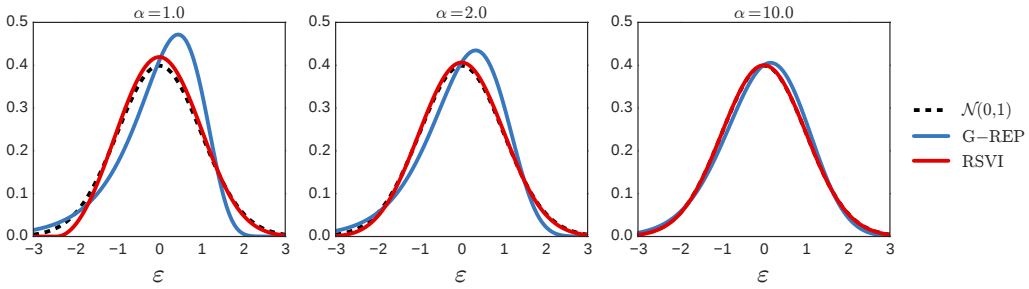


Figure 3: *In the distribution on the transformed space ε for a gamma distribution we can see that the rejection sampling-inspired transformation converges faster to a standard normal. Therefore it is less dependent on the parameter α, which implies a smaller correction term. We compare the transformation of RSVI (this paper) with the standardization procedure suggested in G-REP (Ruiz et al., 2016), for shape parameters $\alpha = \{1, 2, 10\}$.*

5.2 Dirichlet Distribution

The Dirichlet($\alpha_{1:K}$) distribution, with concentration parameters $\alpha_{1:K}$, is a K - dimensional multivariate distribution with $K-1$ degrees of freedom. To simulate random variables we use the fact that if $\tilde{z}_k \sim \text{Gam}(\alpha_k, 1)$ i.i.d., then $z_{1:K} = (\sum_\ell \tilde{z}_\ell)^{-1} (\tilde{z}_1, \ldots, \tilde{z}_K)^\top \sim \text{Dirichlet}(\alpha_{1:K})$.

Thus, we make a change of variables to reduce the problem to that of simulating independent gamma distributed random variables,

$$\mathbb{E}_{q(z_{1:K};\alpha_{1:K})}[f(z_{1:K})] = \int f\left(\frac{\tilde{z}_{1:K}}{\sum_{\ell=1}^K \tilde{z}_\ell}\right) \prod_{k=1}^K \text{Gam}(\tilde{z}_k\,;\alpha_k, 1)\mathrm{d}\tilde{z}_{1:K}.$$

We apply the transformation in Section 5.1 for the gamma-distributed variables, $\tilde{z}_k = h_{\text{Gam}}(\varepsilon_k, \alpha_k)$, where the variables ε_k are generated by independent gamma rejection samplers. To showcase this, we study a simple conjugate model where the exact gradient and posterior are available: a multinomial likelihood with Dirichlet prior and Dirichlet variational distribution. In Figure 4 we show the resulting variance of the first component of the gradient, based on simulated data from a Dirichlet distribution with $K = 100$ components, uniform prior, and $N = 100$ trials. We compare the variance of RSVI (for various shape augmentation settings) with the G-REP approach (Ruiz et al., 2016). RSVI performs better even without the augmentation trick, and significantly better with it.

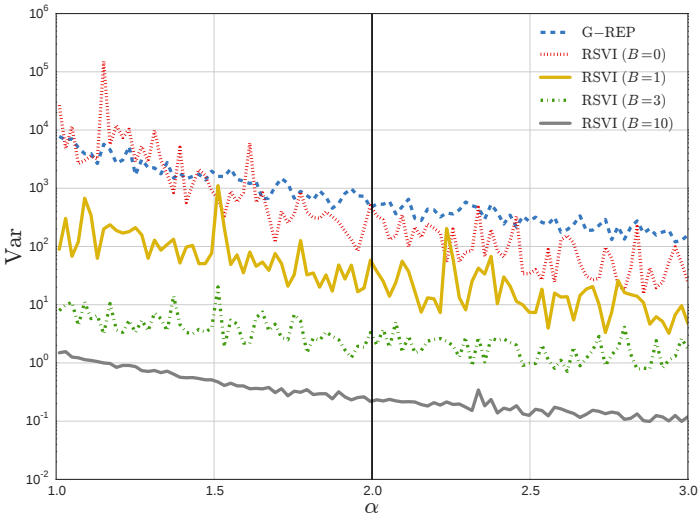


Figure 4: *RSVI (this paper) achieves lower variance compared to G-REP (Ruiz et al., 2016). The estimated variance is for the first component of Dirichlet approximation to a multinomial likelihood with uniform Dirichlet prior. Optimal concentration is $\alpha = 2$, and B denotes shape augmentation.*

6 Experiments

In Section 5 we compared rejection sampling variational inference (RSVI) with generalized reparameterization (G-REP) and found a substantial variance reduction on synthetic examples. Here we evaluate RSVI on a more challenging model, the sparse gamma deep exponential family (DEF) (Ranganath et al., 2015). On two real datasets, we compare RSVI with state-of-the-art methods: automatic differentiation variational inference (ADVI) (Kucukelbir et al., 2015, 2016), black-box variational inference (BBVI) (Ranganath et al., 2014), and G-REP (Ruiz et al., 2016).

Data. The datasets we consider are the Olivetti faces[2] and Neural Information Processing Systems (NIPS) 2011 conference papers. The Olivetti faces dataset consists of 64×64 gray-scale images of human faces in 8 bits, i.e., the data is discrete and in the set $\{0, \ldots, 255\}$. In the NIPS dataset we have documents in a bag-of-words format with an effective vocabulary of 5715 words.

Model. The sparse gamma DEF (Ranganath et al., 2015) is a multi-layered probabilistic model that mimics the architecture of deep neural networks. It models the data using a set of local latent variables $z_{n,k}^{\ell}$ where n indexes observations, k components, and ℓ layers. These local variables are connected between layers through global weights $w_{k,k'}^{\ell}$. The observations are $x_{n,d}$, where d denotes dimen-

[2] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

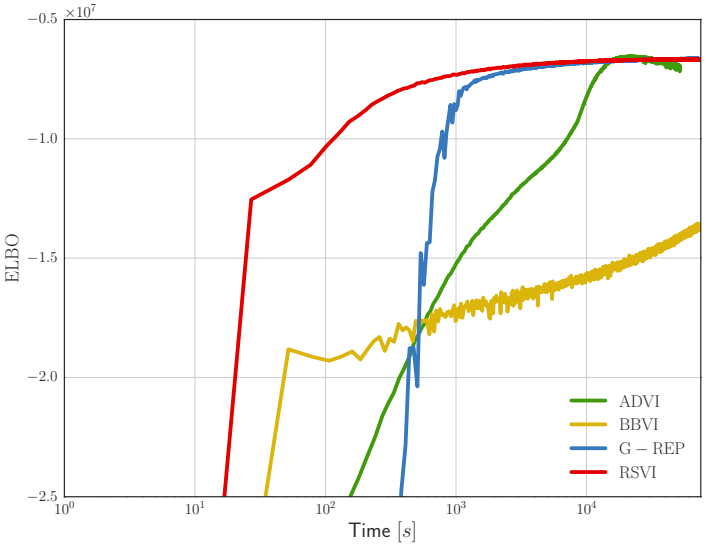


Figure 5: *RSVI (this paper) presents a significantly faster initial improvement of the* evidence lower bound *(ELBO) as a function of wall-clock time. The model is a sparse gamma DEF, applied to the Olivetti faces dataset, and we compare with ADVI (Kucukelbir et al., 2016), BBVI (Ranganath et al., 2014), and G-REP (Ruiz et al., 2016).*

sion. The joint probabilistic model is defined as

$$\begin{aligned} z_{n,k}^{\ell} &\sim \text{Gam}\left(\alpha_z, \frac{\alpha_z}{\sum_{k'} w_{k,k'}^{\ell} z_{n,k'}^{\ell+1}}\right), \\ x_{n,d} &\sim \text{Poisson}\left(\sum_k w_{k,d}^0 z_{n,k}^1\right). \end{aligned} \tag{11}$$

We set $\alpha_z = 0.1$ in the experiments. All priors on the weights are set to Gam(0.1, 0.3), and the top-layer local variables priors are set to Gam(0.1, 0.1). We use 3 layers, with 100, 40, and 15 components in each. This is the same model that was studied by Ruiz et al. (2016), where G-REP was shown to outperform both BBVI (with control variates and Rao-Blackwellization), as well as ADVI. In the experiments we follow their approach and parameterize the variational approximating gamma distribution using the shape and mean. To avoid constrained optimization we use the transform $\theta = \log(1 + \exp(\vartheta))$ for non-negative variational parameters θ, and optimize ϑ in the unconstrained space.

Results. For the Olivetti faces we explore $\eta \in \{0.75, 1, 2, 5\}$ and show the resulting ELBO of the best one in Figure 5. We can see that RSVI has a significantly faster initial improvement than any of the other methods.[3] The wall-clock time for RSVI

[3]The results of G-REP, ADVI and BBVI where reproduced with permission from Ruiz et al. (2016).

	RSVI $B = 1$	RSVI $B = 4$	G-REP
Min	6.0e−4	1.2e−3	2.7e−3
Median	**9.0e7**	**2.9e7**	1.6e12
Max	1.2e17	**3.4e14**	1.5e17

	RSVI $B = 1$	RSVI $B = 4$	G-REP
Min	1.8e−3	1.5e−3	2.6e−3
Median	1.2e4	**4.5e3**	1.5e7
Max	1.4e12	**1.6e11**	3.5e12

Table 1: *The RSVI gradient (this paper) exhibits lower variance than G-REP (Ruiz et al., 2016). We show estimated variance, based on* 10 *samples, of G-REP and RSVI (for* $B = 1, 4$ *shape augmentation steps), for parameters at the initialization point (*top*) and at iteration* 2600 *in RSVI (*bottom*), estimated for the NIPS data.*

is based on a Python implementation (average 1.5s per iteration) using the automatic differentiation package autograd (Maclaurin et al., 2015). We found that RSVI is approximately two times faster than G-REP for comparable implementations. One reason for this is that the transformations based on rejection sampling are cheaper to evaluate. Indeed, the research literature on rejection sampling is heavily focused on finding cheap and efficient transformations.

For the NIPS dataset, we now compare the variance of the gradients between the two estimators, RSVI and G-REP, for different shape augmentation steps B. In Table 1 we show the minimum, median, and maximum values of the variance across all dimensions. We can see that RSVI again clearly outperforms G-REP in terms of variance. Moreover, increasing the number of augmentation steps B provides even further improvements.

7 Conclusions

We introduced rejection sampling variational inference (RSVI), a method for deriving reparameterization gradients when simulation from the variational distribution is done using a acceptance-rejection sampler. In practice, RSVI leads to lower-variance gradients than other state-of-the-art methods. Further, it enables reparameterization gradients for a large class of variational distributions, taking advantage of the efficient transformations developed in the rejection sampling literature.

This work opens the door to other strategies that "remove the lid" from existing black-box samplers in the service of variational inference. As future work, we can consider more complicated simulation algorithms with accept-reject-like steps, such as adaptive rejection sampling, importance sampling, sequential Monte Carlo, or Markov chain Monte Carlo.

Acknowledgements

Christian A. Naesseth is supported by CADICS, a Linnaeus Center, funded by the Swedish Research Council (VR). Francisco J. R. Ruiz is supported by the EU H2020 programme (Marie Skłodowska-Curie grant agreement 706760). Scott W. Linderman is supported by the Simons Foundation SCGB-418011. This work is supported by NSF IIS-1247664, ONR N00014-11-1-0651, DARPA PPAML FA8750-14-2-0009, DARPA SIMPLEX N66001-15-C-4032, Adobe, and the Alfred P. Sloan Foundation. The authors would like to thank Alp Kucukelbir and Dustin Tran for helpful comments and discussion.

Bibliography

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *arXiv:1601.00670*, 2016.

G. Bonnet. Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire. *Annals of Telecommunications*, 19(9):203–220, 1964.

G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12: 2121–2159, jul 2011.

K. Fan, Z. Wang, J. Beck, J. Kwok, and K. A. Heller. Fast second order stochastic backpropagation for variational inference. In *Advances in Neural Information Processing Systems*, 2015.

P. W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, oct 1990.

Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13, New York, NY, USA, 1993. ACM.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, May 2013.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization using gumbel-softmax. In *International Conference on Learning Representations*, 2017. (accepted for publication).

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

David A. Knowles. Stochastic gradient variational Bayes for Gamma approximating distributions. *arXiv:1509.01631v1*, 2015.

A. Kucukelbir, R. Ranganath, A. Gelman, and D. M. Blei. Automatic variational inference in Stan. In *Advances in Neural Information Processing Systems*, 2015.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *arXiv:1603.00788*, 2016.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, 2016.

Dougal Maclaurin, David Duvenaud, Matthew Johnson, and Ryan P. Adams. Autograd: Reverse-mode differentiation of native Python, 2015. URL `http://github.com/HIPS/autograd`.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. (accepted for publication).

George Marsaglia and Wai Wan Tsang. A simple method for generating gamma variables. *ACM Transactions on Mathematical Software*, 26(3):363–372, September 2000.

A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, 2014.

J. W. Paisley, D. M. Blei, and M. I. Jordan. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*, 2012.

Omiros Papaspiliopoulos, Gareth O. Roberts, and Martin Sköld. Non-centered parameterisations for hierarchical models and data augmentation. In *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, page 307. Oxford University Press, USA, 2003.

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

R. Ranganath, L. Tang, L. Charlin, and D. M. Blei. Deep exponential families. In *Artificial Intelligence and Statistics*, 2015.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

Rajesh Ranganath, Dustin Tran, and David M. Blei. Hierarchical variational models. In *International Conference on Machine Learning*, 2016.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2004.

Francisco J. R. Ruiz, Michalis K. Titsias, and David M. Blei. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems*, 2016.

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

Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, 2015.

J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, 2015.

A. Stuart. Gamma-distributed products of independent random variables. *Biometrika*, 49:64–65, 1962.

T. Tieleman and G. Hinton. Lecture 6.5-RMSPROP: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 4, 2012.

M. K. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, 2014.

Dustin Tran, Rajesh Ranganath, and David M. Blei. The variational Gaussian process. In *International Conference on Learning Representations*, 2016.

John von Neumann. Various Techniques Used in Connection with Random Digits. *Journal of Research of the National Bureau of Standards*, 12:36–38, 1951.

Steve Waterhouse, David Mackay, and Tony Robinson. Bayesian methods for mixtures of experts. In *Advances in Neural Information Processing Systems*, pages 351–357. MIT Press, 1996.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Paper H

Variational Sequential Monte Carlo

Authors: Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei

Edited version of the paper:

Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei. Variational sequential Monte Carlo. In *Artificial Intelligence and Statistics (AISTATS)*, pages 968–977, 2018a.

This paper has been formatted to fit this layout.

Variational Sequential Monte Carlo

Christian A. Naesseth[⋆], Scott W. Linderman[†], Rajesh Ranganath[‡], and David M. Blei[†]

[⋆]Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
christian.a.naesseth@liu.se

[†]Dept. of Computer Science
Columbia University
New York City, United States of America
{scott.linderman,david.blei}@columbia.edu

[‡]Dept. of Computer Science
Princeton University
Princeton, United States of America
rajeshr@cs.princeton.edu

Abstract

Many recent advances in large scale probabilistic inference rely on variational methods. The success of variational approaches depends on (i) formulating a flexible parametric family of distributions, and (ii) optimizing the parameters to find the member of this family that most closely approximates the exact posterior. In this paper we present a new approximating family of distributions, the variational sequential Monte Carlo (VSMC) family, and show how to optimize it in variational inference. VSMC melds variational inference (VI) and sequential Monte Carlo (SMC), providing practitioners with flexible, accurate, and powerful Bayesian inference. The VSMC family is a variational family that can approximate the posterior arbitrarily well, while still allowing for efficient optimization of its parameters. We demonstrate its utility on state space models, stochastic volatility models for financial data, and deep Markov models of brain neural circuits.

1 Introduction

Complex data like natural images, text, and medical records require sophisticated models and algorithms. Recent advances in these challenging domains have relied upon variational inference (VI) (Hoffman et al., 2013; Kingma and

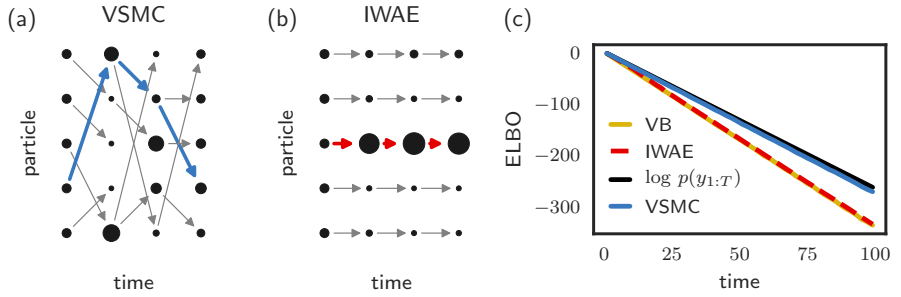


Figure 1: *Comparing* vsmc *and the* iwae. *(a) VSMC constructs a weighted set of particle trajectories using SMC and then samples one according to the final weight. Here, the size of the dot is proportional to the weight, w_t^i; the gray arrows denote the ancestors, a_{t-1}^i; and the blue arrows denote the chosen path, $b_{1:T}$. (b) IWAE does the same, but without resampling. This leads to particle degeneracy as time increases—only one particle has nonneglible weight at time T. (c) The ELBO suffers from this degeneracy: all are comparable when T is small, but as time increases the IWAE provides minimal improvement over standard VB, whereas VSMC still achieves nearly the true marginal likelihood.*

Welling, 2014; Ranganath et al., 2016a). Variational inference excels in quickly approximating the model posterior, yet these approximations are only useful insofar as they are accurate. The challenge is to balance faithful posterior approximation and fast optimization.

We present a new approximating family of distributions called variational sequential Monte Carlo (VSMC). VSMC blends VI and sequential Monte Carlo (SMC) (Gordon et al., 1993; Kitagawa, 1996; Stewart and McCarty, 1992), providing practitioners with a flexible, accurate, and powerful approximate Bayesian inference algorithm. VSMC is an efficient algorithm that can approximate the posterior arbitrarily well.

Standard SMC approximates a posterior distribution of latent variables with N weighted particles iteratively drawn from a proposal distribution. The idea behind *variational* SMC is to view the parameters of the proposal as indexing a family of distributions over latent variables. Each distribution in this variational family corresponds to a particular choice of proposal; to sample the distribution, we run SMC to generate a set of particles and then randomly select one with probability proportional to its weight. Unlike typical variational families, the VSMC family trades off fidelity to the posterior with computational complexity: its accuracy increases with the number of particles N, but so does its computational cost.

We develop the VSMC approximating family, derive its corresponding variational lower bound, and design a stochastic gradient ascent algorithm to optimize its parameters. We connect VSMC to the importance weighted auto-encoder (IWAE)

(Burda et al., 2016) and show that the IWAE lower bound is a special case of the VSMC bound. As an illustration, consider approximating the following posterior with latent variables $x_{1:T}$ and observations $y_{1:T}$,

$$p(x_{1:T} \mid y_{1:T}) = \prod_{t=1}^{T} \mathcal{N}(x_t ; 0, 1)\, \mathcal{N}(y_t ; x_t^2, 1) / p(y_{1:T}).$$

This is a toy Gaussian state space model (SSM) where the observed value at each time step depends on the square of the latent state. Figure 1c shows the approximating power of VSMC versus that of the IWAE and of standard variational Bayes (VB). As the length of the sequence T increases, naïve importance sampling effectively collapses to use only a single particle. VSMC on the other hand maintains a diverse set of particles and thereby achieves a significantly tighter lower bound of the log-marginal likelihood $\log p(y_{1:T})$.

We focus on inference in state space and time series models, but emphasize that VSMC applies to any sequence of probabilistic models, just like standard SMC (Del Moral et al., 2006; Doucet and Johansen, 2009; Naesseth et al., 2014).

In Section 5, we demonstrate the advantages of VSMC on both simulated and real data. First, we show on simulated linear Gaussian SSM data that VSMC can outperform the (locally) optimal proposal (Doucet and Johansen, 2009; Doucet et al., 2001). Then we compare VSMC with IWAE for a stochastic volatility model on exchange rates from financial markets. We find that VSMC achieves better posterior inferences and learns more efficient proposals. Finally, we study recordings of macaque monkey neurons using a probabilistic model based on recurrent neural networks. VSMC reaches the same accuracy as IWAE, but does so with less computation.

Related Work Much effort has been dedicated to learning good proposals for SMC (Cornebise, 2009). Guarniero et al. (2017) adapt proposals through iterative refinement. Naesseth et al. (2015) uses a Monte Carlo approximation to the (locally) optimal proposal (Doucet and Johansen, 2009). Gu et al. (2015) learn proposals by minimizing the Kullback-Leibler (KL) from the posterior to proposal using SMC samples; this strategy can suffer from high variance when the initial SMC proposal is poor. Paige and Wood (2016) learn proposals by forward simulating and inverting the model. In contrast to all these methods, VSMC optimizes the proposal directly with respect to KL divergence from the SMC sampling process to the posterior.

VSMC uses auxillary variables in a posterior approximation. This relates to work in VI, such as Hamiltonian VI (Salimans et al., 2015), variational Gaussian processes (Tran et al., 2016), hierarchical variational models (Ranganath et al., 2016b), and deep auxiliary variational auto-encoders (Maaløe et al., 2016). Another approach uses a sequence of invertible functions to transform a simple variational approximation to a complex one (Dinh et al., 2014; Rezende and Mohamed, 2015).

All of these rich approximations can be embedded inside VSMC to build more flexible proposals.

Archer et al. (2015); Johnson et al. (2016) develop variational inference for state space models with conjugate dynamics, while Krishnan et al. (2017) develop variational approximations for models with nonlinear dynamics and additive Gaussian noise. In contrast, VSMC is agnostic to the distributional choices in the dynamics and noise.

Importance weighted auto-encoders (Burda et al., 2016) obtain the same lower bound as variational importance sampling (VIS), a special case of VSMC. However, VIS provides a new interpretation that enables a more accurate variational approximation; this relates to another interpretation of IWAE by Bachman and Precup (2015); Cremer et al. (2017). Variational particle approximations (Saeedi et al., 2014) also provide variational approximation that improve with the number of particles, but they are restricted to discrete latent variables.

Finally, the log-marginal likelihood lower bound (6) was developed concurrently and independently by Maddison et al. (2017) and Le et al. (2017). The difference with our work lies in how we derive the bound and the implications we explore. Le et al. (2017); Maddison et al. (2017) derive the bound using Jensen's inequality on the SMC expected log-marginal likelihood estimate, focusing on approximate marginal likelihood estimation of model parameters. Rather, we derive (6) as a tractable lower bound to the exact *evidence lower bound* (ELBO) for the new variational family VSMC. In addition to a lower bound on the log-marginal likelihood, this view provides a new variational approximation to the posterior.

2 Background

We begin by introducing the foundation for variational sequential Monte Carlo (VSMC). Let $p(x_{1:t}, y_{1:t})$ be a sequence of probabilistic models for latent (unobserved) $x_{1:t}$ and data $y_{1:t}$, with $t = 1, \dots, T$. In Bayesian inference, we are interested in computing the posterior distribution $p(x_{1:T} \mid y_{1:T})$. Two concrete examples, both from the time-series literature, are hidden Markov models and state space models (Cappé et al., 2005). In both cases, the joint density factorizes as

$$p(x_{1:T}, y_{1:T}) = f(x_1) \prod_{t=2}^{T} f(x_t \mid x_{t-1}) \prod_{t=1}^{T} g(y_t \mid x_t),$$

where f is the prior on x, and g is the observation (data) distribution. For most models computing the posterior $p(x_{1:T} \mid y_{1:T})$ is computationally intractable, and we need approximations such as VI and SMC. Here we construct posterior approximations that combine these two ideas.

In the following sections, we review variational inference and sequential Monte Carlo, develop a variational approximation based on the samples generated by

SMC, and develop a tractable objective to improve the quality of the SMC variational approximation. For concreteness, we focus on the state space model above. But we emphasize that VSMC applies to any sequence of probabilistic models, just like standard SMC (Del Moral et al., 2006; Doucet and Johansen, 2009; Naesseth et al., 2014).

Variational Inference In variational inference we postulate an approximating family of distributions with variational parameters λ, $q(x_{1:T}; \lambda)$. Then we minimize a divergence, often the KL divergence, between the approximating family and the posterior so that $q(x_{1:T}; \lambda) \approx p(x_{1:T} \mid y_{1:T})$. This minimization is equivalent to maximizing the ELBO (Jordan et al., 1999),

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(x_{1:T};\lambda)} \left[\log p(x_{1:T}, y_{1:T}) - \log q(x_{1:T}; \lambda) \right]. \tag{1}$$

VI turns posterior inference into an optimization problem.

Sequential Monte Carlo SMC is a sampling method designed to approximate a sequence of distributions, $p(x_{1:t} \mid y_{1:t})$ for $t = 1 \dots T$ with special emphasis on the posterior $p(x_{1:T} \mid y_{1:T})$. For a thorough introduction to SMC see Doucet and Johansen (2009); Doucet et al. (2001); Schön et al. (2015).

To approximate $p(x_{1:t} \mid y_{1:t})$ SMC uses weighted samples,

$$p(x_{1:t} \mid y_{1:t}) \approx \widehat{p}(x_{1:t} \mid y_{1:t}) \triangleq \sum_{i=1}^{N} \frac{w_t^i}{\sum_\ell w_t^\ell} \delta_{x_{1:t}^i}, \tag{2}$$

where δ_X is the Dirac measure at X.

We construct the weighted set of particles sequentially for $t = 1, \dots, T$. At time $t = 1$ we use standard importance sampling $x_1^i \sim r(x_1)$. For $t > 1$, we start each step by *resampling* auxiliary *ancestor variables* $a_{t-1}^i \in \{1, \dots, N\}$ with probability proportional to the importance weights w_{t-1}^j; next we propose new values, append them to the end of the trajectory, and reweight as follows:

$$\begin{aligned}
&\textit{resample} && a_{t-1}^i \sim \text{Categorical}(w_{t-1}^j / \textstyle\sum_\ell w_{t-1}^\ell) \\
&\textit{propose} && x_t^i \sim r(x_t \mid x_{t-1}^{a_{t-1}^i}), \\
&\textit{append} && x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i), \\
&\textit{reweight} && w_t^i = f(x_t^i \mid x_{t-1}^{a_{t-1}^i})\, g(y_t \mid x_t^i) / r(x_t^i \mid x_{t-1}^{a_{t-1}^i}).
\end{aligned}$$

We refer to the final particles (samples) $x_{1:T}^i$ as *trajectories*. Panels (a) and (b) of Figure 1 show sets of weighted trajectories. The size of the dots represents the weights w_t^i and the arrows represent the ancestors a_{t-1}^i. Importance sampling omits the resampling step, so each ancestor is given by the corresponding particle for the preceding time step.

The trajectories $x_{1:T}^i$ and weights w_T^i define the SMC approximation to the posterior. Critically, as we increase the number of particles, the posterior approximation becomes arbitrarily accurate. SMC also yields an unbiased estimate of the marginal likelihood,

$$\widehat{p}(y_{1:T}) = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} w_t^i. \tag{3}$$

This estimate will play an important role in the VSMC objective.

The proposal distribution $r(x_t \mid x_{t-1})$ is the key design choice. A common choice is the model prior f—it is known as the bootstrap particle filter (BPF) (Gordon et al., 1993). However, proposing from the prior often leads to a poor approximation for a small number of particles, especially if x_t is high-dimensional. Variational SMC addresses this shortcoming; it learns parameterized proposal distributions for efficient inference.

3 Variational Sequential Monte Carlo

We develop VSMC, a new class of variational approximations based on SMC. We first define how to sample from the VSMC family and then derive its distribution. Though generating samples is straightforward, the density is intractable. To this end, we derive a tractable objective, a new lower bound to the ELBO, that is amenable to stochastic optimization. Then, we present an algorithm to fit the variational parameters. Finally, we explore how to learn model parameters using variational expectation–maximization.

To sample from the VSMC family, we run SMC (with the proposals parameterized by variational parameters λ) and then sample once from the empirical approximation of the posterior (2). Because the proposals $r(x_t \mid x_{t-1}\,; \lambda)$ depend on λ, so does the SMC empirical approximation. Algorithm 1 summarizes the generative process for the VSMC family.

The variational distribution $q(x_{1:T}\,; \lambda)$ marginalizes out all the variables produced in the sampling process, save for the output sample $x_{1:T}$. This marginal comes from the joint distribution of all variables generated by VSMC,

$$\widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}, b_T\,; \lambda) = \underbrace{\left[\prod_{i=1}^{N} r(x_1^i\,; \lambda)\right]}_{\textit{step 2}} \cdot \prod_{t=2}^{T} \prod_{i=1}^{N} \underbrace{\left[\frac{w_{t-1}^{a_{t-1}^i}}{\sum_\ell w_{t-1}^\ell}\right]}_{\textit{step 7}} \underbrace{r(x_t^i \mid x_{t-1}^{a_{t-1}^i}\,; \lambda)}_{\textit{step 8}} \underbrace{\left[\frac{w_T^{b_T}}{\sum_\ell w_T^\ell}\right]}_{\textit{step 13}}. \tag{4}$$

(We have annotated this equation with the steps from the algorithm.) In this joint, the final output sample is defined by extracting the b_T-th trajectory $x_{1:T} = x_{1:T}^{b_T}$.

Algorithm 1: Variational Sequential Monte Carlo

Require: Targets $p(x_{1:t}, y_{1:t})$, proposals $r(x_t \mid x_{t-1}\,; \lambda)$, and number of particles N.

for $i = 1 \dots N$ **do**
  Simulate x_1^i from $r(x_1\,; \lambda)$
  Set $w_1^i = f(x_1^i)\, g(y_1 \mid x_1^i) / r(x_1^i\,; \lambda)$
end for
for $t = 2 \dots T$ **do**
  for $i = 1 \dots N$ **do**
    Simulate a_{t-1}^i with $\Pr(a_{t-1}^i = j) = \frac{w_{t-1}^j}{\sum_\ell w_{t-1}^\ell}$
    Simulate x_t^i from $r(x_t \mid x_{t-1}^{a_{t-1}^i}\,; \lambda)$
    Set $x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i)$
    Set $w_t^i = f(x_t^i \mid x_{t-1}^{a_{t-1}^i})\, g(y_t \mid x_t^i) / r(x_t^i \mid x_{t-1}^{a_{t-1}^i}\,; \lambda)$
  end for
end for
Simulate b_T with $\Pr(b_T = j) = w_T^j / \sum_\ell w_T^\ell$
return $x_{1:T} \triangleq x_{1:T}^{b_T}$

Note that the data $y_{1:T}$ enter via the weights and (optionally) the proposal distribution. This joint density is easy to calculate, but for variational inference we need the marginal distribution of $x_{1:T}$. We derive this next.

Let $b_t \triangleq a_t^{b_{t+1}}$ for $t \leq T-1$ denote the ancestors for the trajectory $x_{1:T}$ returned by Algorithm 1. Furthermore, let $\neg b_{1:T}$ be all particle indices *not* equal to $(b_1, \dots, b_T)$, i.e. exactly all the particles that were not returned by Algorithm 1. Then the marginal distribution of $x_{1:T} = x_{1:T}^{b_{1:T}} = (x_1^{b_1}, x_2^{b_2}, \dots, x_T^{b_T})$ is given by the following proposition.

Proposition 1. The VSMC approximation on $x_{1:T}$ is

$$q(x_{1:T} \mid y_{1:T}\,; \lambda) = p(x_{1:T}, y_{1:T})\, \mathbb{E}_{\widetilde{\phi}\left(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,; \lambda\right)}\left[\widehat{p}(y_{1:T})^{-1}\right]. \tag{5}$$

Proof: See the supplementary material A.1. □

This has an intuitive form: the density of the variational posterior is equal to the exact joint times the expected inverse of the normalization constant (c.f. (3)). While we can estimate this expectation with Monte Carlo, it yields a biased estimate of $\log q(x_{1:T} \mid y_{1:T}; \lambda)$ and the ELBO (1).

The surrogate elbo. To derive a tractable objective, we develop a lower bound to the ELBO that is also amenable to stochastic optimization. It is

$$\widetilde{\mathcal{L}}(\lambda) \triangleq \sum_{t=1}^{T} \mathbb{E}_{\widetilde{\phi}(x_{1:t}^{1:N}, a_{1:t-1}^{1:N}; \lambda)} \left[\log \left(\frac{1}{N} \sum_{i=1}^{N} w_t^i \right) \right] = \mathbb{E} \left[\log \widehat{p}(y_{1:T}) \right] \tag{6}$$

We call $\widetilde{\mathcal{L}}(\lambda)$ the *surrogate ELBO*. It is a lower bound to the true ELBO for VSMC or, equivalently, an upper bound on the KL divergence. The following theorem formalizes this fact:

Theorem 1 (Surrogate ELBO). *The surrogate ELBO* (6)*, is a lower bound to the ELBO* (1) *when q is defined by* (5)*, i.e.*

$$\log p(y_{1:T}) \geq \mathcal{L}(\lambda) \geq \widetilde{\mathcal{L}}(\lambda).$$

Proof: See the supplementary material A.2. □

The surrogate ELBO is the expected SMC log-marginal likelihood estimate. We can estimate it unbiasedly as a byproduct of sampling from the VSMC variational approximation (Algorithm 1). We run the algorithm and use the estimate to perform stochastic optimization of the surrogate ELBO.

Stochastic Optimization. While the expectations in the surrogate ELBO are still not available in closed form, we can estimate it and its gradients with Monte Carlo. This admits a stochastic optimization algorithm for finding the optimal variational parameters of the VSMC family.

We assume the proposals $r(x_t \mid x_{t-1}; \lambda)$ are reparameterizable, i.e., we can simulate from r by setting $x_t = h(x_{t-1}, \varepsilon_t; \lambda)$, $\varepsilon_t \sim s(\varepsilon_t)$ for some distribution s not a function of λ. With this assumption, rewrite the gradient of (6) by using the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014),

$$\begin{aligned} \nabla \widetilde{\mathcal{L}}(\lambda) &= g_{\text{rep}} + g_{\text{score}} \\ g_{\text{rep}} &= \mathbb{E} \left[\nabla \log \widehat{p}(y_{1:T}) \right], \\ g_{\text{score}} &= \mathbb{E} \left[\log \widehat{p}(y_{1:T}) \nabla \log \widetilde{\phi}(a_{1:T-1}^{1:N} \mid \varepsilon_{1:T}^{1:N}; \lambda) \right]. \end{aligned} \tag{7}$$

This expansion follows from the product rule, just as in the generalized reparameterizations of Ruiz et al. (2016) and Naesseth et al. (2017). Note that all x_t^i, implicit in the weights w_t^i and $\widehat{p}(y_{1:T})$ are now replaced with their reparameterizations $h(\cdot\,; \lambda)$. The ancestor variables are discrete and cannot be reparameterized—this can lead to high variance in the score function term, g_{score} from (7).

In Section 5, we empirically assess the impact of ignoring g_{score} for optimization. We empirically study optimizing with and without the score function term for a small state space model where standard variance reduction techniques, explained

below, are sufficient. We lower the variance using Rao-Blackwellization (Ranganath et al., 2014; Robert and Casella, 2004), noting that the ancestor variables a_{t-1} have no effect on weights prior to time t,

$$g_{\text{score}} = \sum_{t=2}^{T} \mathbb{E}\left[\log \frac{\widehat{p}(y_{1:T})}{\widehat{p}(y_{1:t-1})}\left(\sum_{i=1}^{N} \nabla \log \frac{w_{t-1}^{a_{t-1}^i}}{\sum_\ell w_{t-1}^\ell}\right)\right]. \tag{8}$$

Furthermore, we use the score function $\nabla \log \widetilde{\phi}(a_{1:T-1}^{1:N} \mid \varepsilon_{1:T}^{1:N}; \lambda)$ with an estimate of the future log average weights as a control variate (Ranganath et al., 2014).

We found that ignoring the score function term g_{score} (8) from the ancestor variables, leads to faster convergence and very little difference in final ELBO. This corresponds to approximating the gradient of $\widetilde{\mathcal{L}}$ by

$$\nabla \widetilde{\mathcal{L}}(\lambda) \approx \mathbb{E}\left[\nabla \log \widehat{p}(y_{1:T})\right] = g_{\text{rep}}. \tag{9}$$

This is the gradient we propose to use for optimizing the variational parameters of VSMC. See the supplementary material A.3 for more details, where we also provide a general score function-like estimator and the control variates.

Algorithm. We now describe the full algorithm to optimize the VSMC variational approximation. We form stochastic gradients $\widehat{\nabla}\widetilde{\mathcal{L}}(\lambda)$ by estimating (9) using a single sample from $s(\cdot)\widetilde{\phi}(\cdot \mid \cdot; \lambda)$. The sample is obtained as a byproduct of sampling VSMC (Algorithm 1). We use the step-size sequence Adam (Kingma and Ba, 2015) or ρ^n proposed by Kucukelbir et al. (2017),

$$\begin{aligned} \rho^n &= \eta \cdot n^{-1/2+\delta} \cdot \left(1 + \sqrt{s^n}\right)^{-1}, \\ s^n &= t\left(\widehat{\nabla}\widetilde{\mathcal{L}}(\lambda^n)\right)^2 + (1-t)s^{n-1}, \end{aligned} \tag{10}$$

where n is the iteration number. We set $\delta = 10^{-16}$ and $t = 0.1$, and we try different values for η. Algorithm 2 summarizes this optimization algorithm.[1]

Variational Expectation Maximization. Suppose the target distribution of interest $p(x_{1:T} \mid y_{1:T}; \theta)$ has a set of unknown parameters θ. We can fit the parameters using variational expectation–maximization (VEM) (Beal and Ghahramani, 2003). The surrogate ELBO is updated accordingly

$$\log p(y_{1:T}; \theta) \geq \tilde{\mathcal{L}}(\lambda, \theta) \tag{11}$$

where the normalization constant $p(y_{1:T}; \theta)$ is now a function of the parameters θ. Note that the expression for $\tilde{\mathcal{L}}(\lambda, \theta)$ is exactly the same as (6), but where the weights (and potentially proposals) now include a dependence on the model parameters θ. Analogously, the reparameterization gradients have the same

[1] Reference implementation using Adam is available at github.com/blei-lab/variational-smc.

Algorithm 2: Stochastic Optimization for VSMC

Require: Data $y_{1:T}$, model $p(x_{1:T}, y_{1:T})$, proposals $r(x_t \mid x_{t-1}; \lambda)$, number of particles N
Ensure: Variational parameters $\lambda^\star$
1: **repeat**
2: Estimate the gradient $\widehat{\nabla}\widetilde{\mathcal{L}}(\lambda^n)$ given by (9)
3: Compute stepsize ρ^n with (10)
4: Update $\lambda^{n+1} = \lambda^n + \rho^n \widehat{\nabla}\widetilde{\mathcal{L}}(\lambda^n)$
5: **until convergence**

form as (9). We can maximize (11), with respect to both θ and λ, using stochastic optimization. With data subsampling, VSMC extends to large-scale datasets of conditionally independent sequences (Hoffman et al., 2013; Titsias and Lázaro-Gredilla, 2014).

4 Perspectives on Variational SMC

We give some perspectives on VSMC. First, we consider the VSMC special cases of $N = 1$ and $T = 1$. For $N = 1$, VSMC reduces to a structured variational approximation: there is no resampling and the variational distribution is exactly the proposal. For $T = 1$, VSMC leads to a special case we call variational importance sampling, and a reinterpretation of the IWAE (Burda et al., 2016), which we explore further in the first half of this section.

Then, we think of sampling from VSMC as sampling a highly optimized SMC approximation. This means many of the theoretical SMC results developed over the past 25 years can be adapted for VSMC. We explore some examples in the second half of this section.

Variational Importance Sampling (VIS). The case where $T = 1$ is SMC without any resampling, i.e., importance sampling. The corresponding special case of VSMC is VIS. The surrogate ELBO for VIS is exactly equal to the IWAE lower bound (Burda et al., 2016).

This equivalence provides new intuition behind the IWAE's variational approximation on the latent variables. If we want to make use of the approximation $q(x_{1:T}; \lambda^\star)$ learned with the IWAE lower bound, samples from the latent variables should be generated with Algorithm 1, i.e. VIS. For VIS it is possible to show that the surrogate ELBO is always tighter than the one obtained by standard VB (equivalent to VIS with $N = 1$) (Burda et al., 2016). This result does not carry over to VSMC, i.e. we can find cases when the resampling creates a looser bound compared to standard VB or VIS. However, in practice the VSMC lower bound outperforms the VIS lower bound.

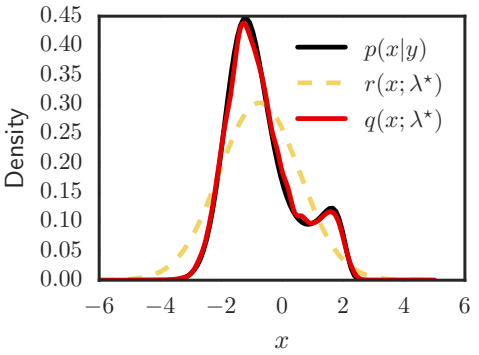


Figure 2: *Example of VIS $q(x\,;\lambda)$ approximating a multimodal $p(x \mid y)$ with a Gaussian proposal $r(x\,;\lambda)$.*

Figure 2 provides a simple example of VIS applied to a multimodal $p(x\,|\,y) \propto \mathcal{N}(x\,;0,1)\,\mathcal{N}(y\,;x^2/2, e^{x/2})$ with a normal proposal $r(x\,;\lambda) = \mathcal{N}(x\,;\mu,\sigma^2)$ and a kernel density estimate of the corresponding variational approximation $q(x\,;\lambda)$. The number of particles is $N = 10$. Standard VB with a Gaussian approximation only captures one of the two modes; which one depends on the initialization. We see that even a simple proposal can lead to a very flexible posterior approximation. This property is also inherited by the more general $T > 1$ case, VSMC.

Theoretical Properties. The normalization constant estimate of the SMC sampler, $\widehat{p}(y_{1:T})$, is unbiased (Del Moral, 2004; Naesseth et al., 2014; Pitt et al., 2012). This, together with Jensen's inequality, implies that the surrogate ELBO $\mathbb{E}[\log \widehat{p}(y_{1:T})]$ is a lower bound to $\log p(y_{1:T})$. If $\log \widehat{p}(y_{1:T})$ is uniformly integrable it follows (Del Moral, 2004), as $N \to \infty$, that

$$\widetilde{\mathcal{L}}(\lambda) = \mathcal{L}(\lambda) = \log p(y_{1:T}).$$

This fact means that the gap in Theorem 1 disappears and the distribution of the trajectory returned by VSMC will tend to the true target distribution $p(x_{1:T} \mid y_{1:T})$. A bound on the KL divergence gives us the rate

$$\mathrm{KL}\left(q(x_{1:T}\,;\lambda) \,\middle\|\, p(x_{1:T} \mid y_{1:T})\right) \leq \frac{c(\lambda)}{N},$$

for some constant $c(\lambda) < \infty$. This is a special case of a "propagation of chaos" result from Del Moral (2004, Theorem 8.3.2).

We can arrive at this result informally by studying (5): as the number of particles increases, the marginal likelihood estimate will converge to the true marginal likelihood and the variational posterior will converge to the true posterior. Huggins and Roy (2017) provide further bounds on various divergences and metrics between SMC and the target distribution.

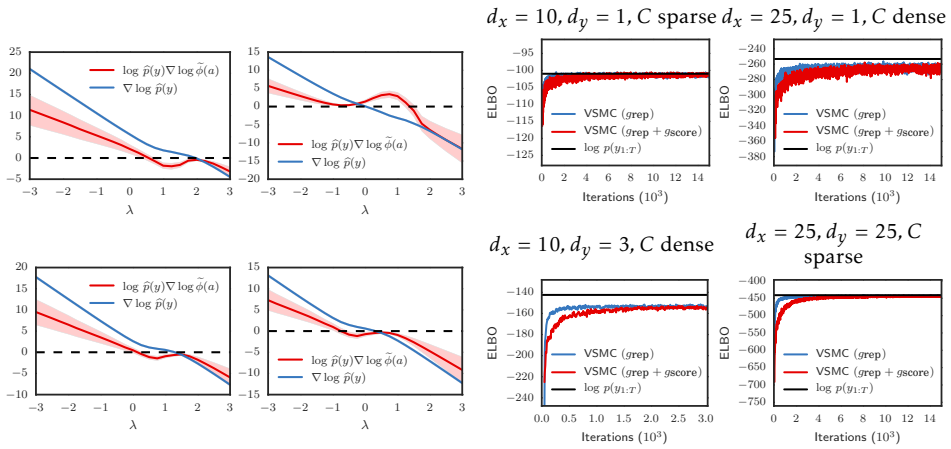


Figure 3: (Left) *Mean and spread of the stochastic gradient components* g_{score} (8) *and* g_{rep} (9)*, for the scalar linear Gaussian model on four randomly generated datasets, where the number of particles is* $N = 2$*.* (Right) *Log-marginal likelihood (*$\log p(y_{1:T})$*) and* ELBO *as a function of iterations for* VSMC *with biased gradients (blue) or unbiased gradients (red). Results for four different linear Gaussian models.*

vsmc and T**.** Like SMC, variational sequential Monte Carlo scales well with T. Bérard et al. (2014) show a central limit theorem for the SMC approximation $\log \widehat{p}(y_{1:T}) - \log p(y_{1:T})$ with $N = bT$, where $b > 0$, as $T \to \infty$. Under the same conditions as in that work, and assuming that $\log \widehat{p}(y_{1:T})$ is uniformly integrable, we can show that

$$\mathrm{KL}\left(q(x_{1:T}\,;\lambda)\,\middle\|\,p(x_{1:T}\,|\,y_{1:T})\right) \leq -\mathbb{E}\left[\log \frac{\widehat{p}(y_{1:T})}{p(y_{1:T})}\right] \xrightarrow[T\to\infty]{} \frac{\sigma^2(\lambda)}{2b}, \quad 0 < \sigma^2(\lambda) < \infty.$$

The implication for VSMC is significant. We can make the variational approximation *arbitrarily accurate* by setting $N \propto T$, even as T goes to infinity. The supplement shows that this holds in practice; see A.4 for the toy example from Figure 1. We emphasize that neither standard VB nor IWAE (VIS) have this property.

5 Empirical Study

Linear Gaussian State Space Model The linear Gaussian SSM is a ubiquitous model of time series data that enjoys efficient algorithms for computing the exact posterior. We use this model to study the convergence properties and impact of biased gradients for VSMC. We further use it to confirm that we learn good proposals. We compare to the bootstrap particle filter (BPF), which uses the prior as

a proposal, and the (locally) optimal proposal that tilts the prior with the likelihood.

The model is

$$x_t = Ax_{t-1} + v_t,$$
$$y_t = Cx_t + e_t,$$

where $v_t \sim \mathcal{N}(0, Q)$, $e_t \sim \mathcal{N}(0, R)$, and $x_1 \sim \mathcal{N}(0, I)$. The log-marginal likelihood $\log p(y_{1:T})$ can be computed using the Kalman filter.

We study the impact of the biased gradient (9) for optimizing the surrogate ELBO (6). First, consider a simple scalar model with $A = 0.5$, $Q = 1$, $C = 1$, $R = 1$, and $T = 2$. For the proposal we use $r(x_t \,|\, x_{t-1}\,; \lambda) = \mathcal{N}(x_t\,; \lambda + 0.5x_{t-1}, 1)$, with $x_0 \equiv 0$. Figure 3 (left) shows the mean and spread of estimates of g_{score} (8), with control variates, and g_{rep} (9), as a function of λ for four randomly generated datasets. The optimal setting of λ is where the sum of the means is equal to zero. Ignoring the score function term g_{score} (8) will lead to a perturbation of the optimal λ. However, even for this simple model, the variance of the score function term (red) is several orders of magnitude higher than that of the reparameterization term (blue), despite the variance reduction techniques of Section 3. This variance has a significant impact on the convergence speed of the stochastic optimization.

Next, we study the magnitude of the perturbation, and its effect on the surrogate ELBO. We generate data with $T = 10$, $(A)_{ij} = \alpha^{|i-j|+1}$ for $\alpha = 0.42$, $Q = I$, and $R = I$. We explored several settings of $d_x = \dim(x_t)$, $d_y = \dim(y_t)$, and C. Sparse C measures the first d_y components of x_t, and dense C has randomly generated elements $C_{ij} \sim \mathcal{N}(0, 1)$. Figure 3 (right) shows the true log-marginal likelihood and ELBO as a function of iteration. It shows VSMC with biased gradients (blue) and unbiased gradients (red). We choose the proposal

$$r(x_t \mid x_{t-1}\,; \lambda) = \mathcal{N}\left(x_t \mid \mu_t + \operatorname{diag}(\beta_t)Ax_{t-1}, \operatorname{diag}(\sigma_t^2)\right).$$

with $\lambda = \left\{\mu_t, \beta_t, \sigma_t^2\right\}_{t=1}^{T}$, and set the number of particles to $N = 4$. Note that while the gradients are biased, the resulting ELBO is not. We can see that the final VSMC ELBO values are very similar, regardless of whether we train with biased or unbiased gradients. However, biased gradients converge faster. Thus, we use biased gradients in the remainder of our experiments.

Next, we study the effect of learning the proposal using VSMC compared with standard proposals in the SMC literature. The most commonly used is the BPF, sampling from the prior f. We also consider the so-called optimal proposal, $r \propto f \cdot g$, which minimizes the variance of the incremental importance weights (Doucet and Johansen, 2009). Table 1 shows results for a linear Gaussian SSM when $T = 25$, $Q = 0.1^2 I$, $R = 1$, $d_x = 10$, and $d_y = 1$. Because of the relatively high-dimensional state, BPF exhibits significant bias whereas the optimal proposal SMC performs much better. VSMC outperforms them both, learning an

Table 1: *ELBO for BPF, SMC with (locally) optimal proposal, and VSMC. The true log-marginal likelihood is given by* $\log p(y_{1:T}) = -236.9$.

	BPF	Optimal SMC	VSMC
ELBO	−6701.4	−253.4	**−237.8**

Table 2: *ELBO for the stochastic volatility model with* $T = 119$ *on exchange rate data. We compare VSMC (this paper) with IWAE and structured VI.*

	Method	ELBO
	Structured VI	6905.1
$N = 4$	IWAE	6911.2
	VSMC	**6921.6**
$N = 8$	IWAE	6912.4
	VSMC	**6935.8**
$N = 16$	IWAE	6913.3
	VSMC	**6936.6**

accurate proposal that results in an ELBO only 0.9 nats lower than the true log-marginal likelihood. We further emphasize that the optimal proposal is unavailable for most models.

Stochastic Volatility A common model in financial econometrics is the (multivariate) stochastic volatility model (Chib et al., 2009). The model is

$$x_t = \mu + \phi(x_{t-1} - \mu) + v_t,$$
$$y_t = \beta \exp\left(\frac{x_t}{2}\right) e_t,$$

where $v_t \sim \mathcal{N}(0, Q)$, $e_t \sim \mathcal{N}(0, I)$, $x_1 \sim \mathcal{N}(\mu, Q)$, and $\theta = (\mu, \phi, Q, \beta)$. (In the multivariate case, multiplication is element-wise.) Computing $\log p(y_{1:T}\,;\theta)$ and its gradients for this model is intractable, we study the VEM approximation to find the unknown parameters θ. We compare VSMC with IWAE and structured VI. For the proposal in VSMC and IWAE we choose

$$r(x_t \,|\, x_{t-1}\,;\lambda,\theta) \propto f(x_t \,|\, x_{t-1}\,;\theta)\,\mathcal{N}(x_t\,;\mu_t,\Sigma_t),$$

with variational parameters $\lambda = (\mu_1, \ldots, \mu_T, \Sigma_1, \ldots, \Sigma_T)$. We define the variational approximation for structured VI to be $q(x_{1:T}\,;\lambda,\theta) = \prod_{t=1}^{T} r(x_t \,|\, x_{t-1}\,;\lambda,\theta)$.

We study 10 years of monthly returns (9/2007 to 8/2017) for the exchange rate of 22 international currencies with respect to US dollars. The data is from the Federal Reserve System. Table 2 reports the optimized ELBO (higher is better) for different settings of the number of particles/samples $N = \{4, 8, 16\}$. VSMC outperforms the competing methods with almost 0.2 nats per time-step.

In theory we can improve the bound of both IWAE and VSMC by increasing the number of samples N. This means we can first learn proposals using only a few particles N, for computational efficiency. Then, at test time, we can increase N as needed for improved accuracy. We study the impact of increasing the number of samples for VSMC and IWAE using fix $\theta^\star$ and $\lambda^\star$ optimized with $N = 16$. Figure 4 shows that the gain for IWAE is limited, whereas for VSMC it can be significant.

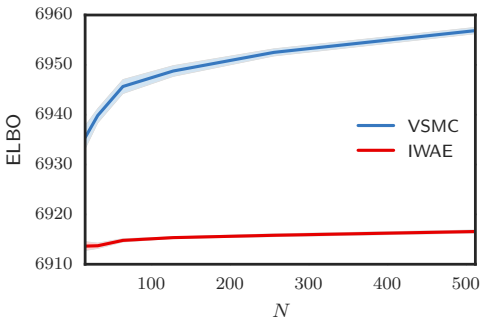


Figure 4: *The estimated ELBO for VSMC (this paper) and IWAE , with confidence bands, as a function of the number of particles N for fix $\theta^\star$, $\lambda^\star$.*

Deep Markov Model An important problem in neuroscience is understanding dynamics of neural circuits. We study a population of 105 motor cortex neurons simultaneously recorded in a macaque monkey as it performed reaching movements (c.f. Gao et al., 2016). In each trial, the monkey reached toward one of fourteen targets; each trial is $T = 21$ time steps long. We train on 700 trials and test on 84.

We use recurrent neural networks to model both the dynamics and observations. The model is

$$\begin{aligned} x_t &= \mu_\theta(x_{t-1}) + \exp\left(\sigma_\theta(x_{t-1})/2\right) v_t, \\ y_t &\sim \text{Poisson}\left(\exp\left(\eta_\theta(x_t)\right)\right), \end{aligned}$$

where $v_t \sim \mathcal{N}(0, I)$, $x_0 \equiv 0$, and μ, σ, η are neural networks parameterized by θ. The multiplication in the transition dynamics is element-wise. This is a deep Markov model (Krishnan et al., 2017).

For inference we use the following proposal for both VSMC and IWAE,

$$\begin{aligned} r(x_t \,|\, x_{t-1}, y_t \,; \lambda) \propto\; & \mathcal{N}\left(x_t \,; \mu^x_\lambda(x_{t-1}), \exp\left(\sigma^x_\lambda(x_{t-1})\right)\right) \\ & \times\; \mathcal{N}\left(x_t \,; \mu^y_\lambda(y_t), \exp\left(\sigma^y_\lambda(y_t)\right)\right), \end{aligned}$$

where $\mu^x, \sigma^x, \mu^y, \sigma^y$ are neural networks parameterized by λ, and the proposal factorizes over the components of x_t. Figure 5 illustrates the result for $d_x = \{3, 5, 10\}$ with $N = 8$. VSMC gets to the same ELBO faster.

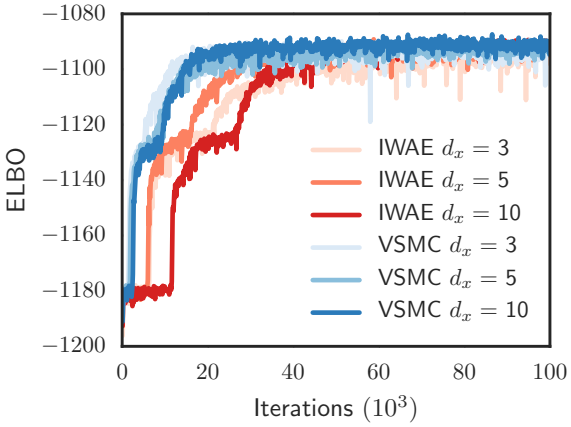


Figure 5: *The estimated* ELBO *of the neural population test data as a function of iterations for* VSMC *(this paper) and* IWAE*, for* $d_x = \{3, 5, 10\}$ *and* $T = 21$.

6 Conclusions

We introduced the variational sequential Monte Carlo (VSMC) family, a new variational approximating family that provides practitioners with a flexible, accurate, and powerful approximate Bayesian inference algorithm. VSMC melds variational inference (VI) and sequential Monte Carlo (SMC). This results in a variational approximation that lets us trade-off fidelity to the posterior with computational complexity.

Acknowledgements

Christian A. Naesseth is supported by CADICS, a Linnaeus Center, funded by the Swedish Research Council (VR). Scott W. Linderman is supported by the Simons Foundation SCGB-418011. This work is supported by ONR N00014-11-1-0651, DARPA PPAML FA8750-14-2-0009, the Alfred P. Sloan Foundation, and the John Simon Guggenheim Foundation.

Appendix

A Variational Sequential Monte Carlo – Supplementary Material

A.1 Proof of Proposition 1

We start by noting that the distribution of all random variables generated by the VSMC algorithm is given by

$$\widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}, b_T\,;\lambda) = \frac{w_T^{b_T}}{\sum_\ell w_T^\ell}\prod_{i=1}^{N} r(x_1^i\,;\lambda)\cdot\prod_{t=2}^{T}\prod_{i=1}^{N}\frac{w_{t-1}^{a_{t-1}^i}}{\sum_\ell w_{t-1}^\ell} r(x_t^i|x_{t-1}^{a_{t-1}^i}\,;\lambda). \quad (12)$$

We are interested in the marginal distribution

$$q(x_{1:T}\,;\lambda) \triangleq \widetilde{\phi}(x_{1:T}\,;\lambda) = \mathbb{E}_{b_{1:T}}[\widetilde{\phi}(x_{1:T}^{b_{1:T}}, b_{1:T}\,;\lambda)].$$

A key observation is that the distribution of $b_{1:T} \mid x_{1:T}$, the conditional distribution of the ancestral path of the returned particle, is uniform on $\{1,\dots,N\}^T$. Thus we get

$$q(x_{1:T}\,;\lambda) = \frac{\widetilde{\phi}(x_{1:T}^{b_{1:T}}, b_{1:T}\,;\lambda)}{\widetilde{\phi}(b_{1:T}\mid x_{1:T}\,;\lambda)} = \frac{1}{N^{-T}}\sum_{a_{1:T-1}^{\neg b_{1:T-1}}}\int \widetilde{\phi}(x_{1:T}^{b_{1:T}}, x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda)\,\mathrm{d}x_{1:T}^{\neg b_{1:T}}, \quad (13)$$

where

$$\begin{aligned}
\frac{1}{N^{-T}}\widetilde{\phi}(x_{1:T}^{b_{1:T}}, x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda) &= N^T\frac{w_1^{b_1}}{\sum_\ell w_1^\ell} r(x_1^{b_1}\,;\lambda)\prod_{t=2}^{T}\frac{w_t^{b_t}}{\sum_\ell w_t^\ell} r(x_t^{b_t}\mid x_{t-1}^{b_{t-1}}\,;\lambda)\\
&\cdot\prod_{\substack{i=1\\ i\neq b_1}}^{N} r(x_1^i\,;\lambda)\cdot\prod_{t=2}^{T}\prod_{\substack{i=1\\ i\neq b_t}}^{N}\frac{w_{t-1}^{a_{t-1}^i}}{\sum_\ell w_{t-1}^\ell} r(x_t^i|x_{t-1}^{a_{t-1}^i}\,;\lambda)\\
&= p(x_1^{b_1}, y_1)\prod_{t=2}^{T}\frac{p(x_{1:t}^{b_{1:t}}, y_{1:t})}{p(x_{1:t-1}^{b_{1:t-1}}, y_{1:t-1})}\prod_{t=1}^{T}\frac{1}{\frac{1}{N}\sum_\ell w_t^\ell}\cdot\prod_{\substack{i=1\\ i\neq b_1}}^{N} r(x_1^i\,;\lambda)\\
&\cdot\prod_{t=2}^{T}\prod_{\substack{i=1\\ i\neq b_t}}^{N}\frac{w_{t-1}^{a_{t-1}^i}}{\sum_\ell w_{t-1}^\ell} r(x_t^i|x_{t-1}^{a_{t-1}^i}\,;\lambda)\\
&= p(x_{1:T}^{b_{1:T}}, y_{1:T})\prod_{t=1}^{T}\frac{1}{\frac{1}{N}\sum_\ell w_t^\ell}\cdot\widetilde{\phi}(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda).
\end{aligned}$$

We insert the above expression in (13) and we get

$$q(x_{1:T}\,;\lambda) = p(x_{1:T}^{b_{1:T}}, y_{1:T}) \sum_{a_{1:T-1}^{\neg b_{1:T-1}}} \int \left(\prod_{t=1}^{T} \frac{1}{N}\sum_{i=1}^{N} w_t^i\right)^{-1} \cdot \widetilde{\phi}(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda)\,\mathrm{d}x_{1:T}^{\neg b_{1:T}}$$

$$= p(x_{1:T}^{b_{1:T}}, y_{1:T})\mathbb{E}_{\widetilde{\phi}(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda)}\left[\left(\prod_{t=1}^{T} \frac{1}{N}\sum_{i=1}^{N} w_t^i\right)^{-1}\right]. \tag{14}$$

□

A.2 Proof of Theorem 1

The *evidence lower bound* (ELBO), using the above result about the distribution of $q(x_{1:T}\,;\lambda)$, is given by

$$\begin{aligned}\mathcal{L}(\lambda) &= \mathbb{E}_{q(x_{1:T}\,;\lambda)}\left[\log p(x_{1:T}, y_{1:T}) - \log q(x_{1:T}\,;\lambda)\right] \\ &= -\int \left\{ p(x_{1:T}^{b_{1:T}}, y_{1:T})\mathbb{E}_{\widetilde{\phi}(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda)}\left[\frac{1}{\prod_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N} w_t^i}\right] \cdot \right. \\ &\qquad \left. \cdot \log \mathbb{E}_{\widetilde{\phi}(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda)}\left[\frac{1}{\prod_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N} w_t^i}\right]\right\} \mathrm{d}x_{1:T}^{b_{1:T}}. \end{aligned} \tag{15}$$

Note that $-t\log t$ is a concave function for $t > 0$, this means by the conditional Jensen's inequality we have $-\mathbb{E}[t]\log\mathbb{E}[t] \geq -\mathbb{E}[t\log t]$. If we apply this to (15) we get

$$\begin{aligned}\mathcal{L}(\lambda) &\geq \int \mathbb{E}_{\widetilde{\phi}(x_{1:T}^{\neg b_{1:T}}, a_{1:T-1}^{\neg b_{1:T-1}}\,;\lambda)}\left[\frac{p(x_{1:T}^{b_{1:T}}, y_{1:T})}{\prod_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N} w_t^i}\sum_{t=1}^{T}\log\left(\frac{1}{N}\sum_{i=1}^{N} w_t^i\right)\right]\mathrm{d}x_{1:T}^{b_{1:T}} \\ &= \mathbb{E}_{\widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\,;\lambda)}\left[\sum_{t=1}^{T}\log\left(\frac{1}{N}\sum_{i=1}^{N} w_t^i\right)\right] = \widetilde{\mathcal{L}}(\lambda),\end{aligned}$$

where the last step follows because $q(x_{1:T}\,;\lambda)$ is the marginal of $\widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\,;\lambda)$. □

A.3 Stochastic Optimization

For the control variates we use

$$\sum_{t=2}^{T} c_t \mathbb{E}_{s(\cdot)\widetilde{\phi}(\cdot|\cdot\,;\lambda)}\left[\sum_{i=1}^{N}\nabla\log w_{t-1}^{a_{t-1}^i} - \sum_{\ell=1}^{N}\frac{w_{t-1}^{\ell}}{\sum_m w_{t-1}^m}\nabla\log w_{t-1}^{\ell}\right]$$

where

$$c_t = \mathbb{E}_{s(\cdot)\widetilde{\phi}(\cdot|\cdot\,;\lambda)}\left[\sum_{t'=t}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} w_{t'}^i\right)\right].$$

In practice we use a stochastic estimate of c_t.

For $T = 2$ we can use a leave-one-out estimator of the ancestor variable score function gradient

$$\sum_{i=1}^{N} \mathbb{E}_{s(\cdot)\widetilde{\phi}(\cdot|\cdot\,;\lambda)}\left[\log\left(\frac{N-1}{N}\frac{\sum_{\ell=1}^{N} w_2^\ell}{\sum_{j\neq i} w_2^j}\right)\left(\nabla \log w_1^{a_1^i} - \sum_{\ell=1}^{N}\frac{w_1^\ell}{\sum_m w_1^m}\nabla \log w_1^\ell\right)\right].$$

Score Function Gradient Below we provide the derivation of a score function-like estimator that is applicable in very general settings. However, we have found that in practice the variance tends to be quite high.

$$\begin{aligned}\nabla\widetilde{\mathcal{L}}(\lambda) &= \nabla\mathbb{E}_{\widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N};\lambda)}\left[\log \widehat{p}(y_{1:T})\right] \\ &= \mathbb{E}_{\widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N};\lambda)}\left[\nabla \log \widehat{p}(y_{1:T}) + \log \widehat{p}(y_{1:T})\nabla \log \widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\,;\lambda)\right],\end{aligned}$$

with

$$\nabla \log \widehat{p}(y_{1:T}) = \nabla \sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} w_t^i\right) = \sum_{t=1}^{T}\sum_{i=1}^{N}\frac{w_t^i}{\sum_\ell w_t^\ell}\nabla \log w_t^i,$$

and

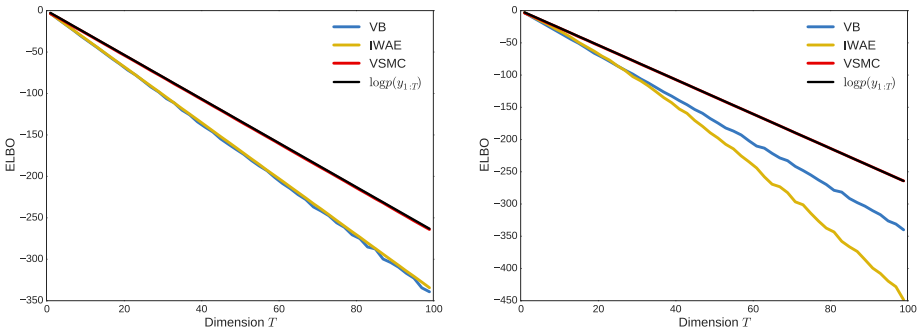
$$\begin{aligned}&\nabla \log \widetilde{\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}\,;\lambda) \\ &= \sum_{i=1}^{N}\left[\nabla \log r(x_1^i\,;\lambda) + \sum_{t=2}^{T}\left[\nabla \log r(x_t^i|x_{t-1}^{a_{t-1}^i}\,;\lambda) + \nabla \log w_{t-1}^{a_{t-1}^i} - \sum_{\ell=1}^{N}\bar{w}_{t-1}^\ell \nabla \log w_{t-1}^\ell\right]\right].\end{aligned}$$

A.4 Scaling With Dimension

In this section we study how the methods compare on a simple toy model defined by

$$p(x_{1:T}, y_{1:T}) = \prod_{t=1}^{T} \mathcal{N}(x_t\,;0,1)\mathcal{N}(y_t\,;x_t^2,1).$$

We study the data set $y_t = 3, \forall t$. Figure 6 shows the result when we let the number of samples in importance weighted auto-encoder (IWAE) (variational importance sampling (VIS)) and VSMC grow with the dimension $N = 2T$. For low T the



(a) $\lambda_{IWAE} = \lambda^{\star}_{VB}$ ***(b)*** $\lambda_{IWAE} = \lambda^{\star}_{VSMC}$

Figure 6: *ELBO, for standard VB, IWAE, and VSMC, as a function of the dimension T of a toy problem. Here we set the number of samples in IWAE and VSMC to be $N = 2T$.*

optimal parameters for IWAE are close to $\lambda^{\star}_{\text{VSMC}}$. On the other hand for high T, the optimal parameters for IWAE are close to those of standard variational Bayes (VB), i.e. $\lambda^{\star}_{\text{VB}}$. Figure 6 indicates that just by letting $N \propto T$, VSMC can achieve arbitrarily good approximation of $p(x_{1:T} \mid y_{1:T})$ even if $T \to \infty$. This holds, under some regularity conditions, even if $p(x_{1:T}, y_{1:T})$ is a state space model (Bérard et al., 2014). This asymptotic approximation property is not satisfied by VIS, we see in Figure 6 that the approximation deteriorates as T increases. Note that this does not hold if the dimension of the latent space, i.e. $\dim(x_t)$, tends to infinity rather than the number of time points T.

Bibliography

E. Archer, I. Memming Park, L. Buesing, J. Cunningham, and L. Paninski. Black box variational inference for state space models. *arXiv:1511.07367*, November 2015.

Philip Bachman and Doina Precup. Training deep generative models: Variations on a theme. In *NIPS Approximate Inference Workshop*, 2015.

Matthew J. Beal and Zoubin Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian statistics*, 2003.

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

Jean Bérard, Pierre Del Moral, and Arnaud Doucet. A lognormal central limit theorem for particle approximations of normalizing constants. *Electronic Journal of Probability*, 2014.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer-Verlag New York, 2005.

Siddhartha Chib, Yasuhiro Omori, and Manabu Asai. *Multivariate Stochastic Volatility*, pages 365–400. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

J. Cornebise. *Adaptive Sequential Monte Carlo Methods*. PhD thesis, University Pierre and Marie Curie–Paris 6, 2009.

C. Cremer, Q. Morris, and D. Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv:1704.02916*, April 2017.

Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and interacting particle systems with applications*. Springer-Verlag New York, 2004.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2006.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv:1410.8516*, 2014.

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, Nando De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*. Springer-Verlag New York, 2001.

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

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.

Shixiang Gu, Zoubin Ghahramani, and Richard E Turner. Neural adaptive sequential Monte Carlo. In *Advances in Neural Information Processing Systems*, 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.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, May 2013.

J. H. Huggins and D. M. Roy. Sequential Monte Carlo as approximate sampling: bounds, adaptive resampling via ∞-ESS, and an application to particle Gibbs. *arXiv:1503.00966v2*, March 2017.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, 2016.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

R. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(1):430–474, January 2017. ISSN 1532-4435.

T. A. Le, M. Igl, T. Jin, T. Rainforth, and F. Wood. Auto-Encoding Sequential Monte Carlo. *arXiv:1705.10306*, May 2017.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, 2016.

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.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B Schön. Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems*, 2014.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B Schön. Nested sequential Monte Carlo methods. In *International Conference on Machine Learning*, 2015.

Christian A. Naesseth, Francisco J. R. Ruiz, Scott W. Linderman, and David M. Blei. Reparameterization gradients through acceptance-rejection sampling algorithms. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Brooks Paige and Frank Wood. Inference networks for sequential Monte Carlo in graphical models. In *International Conference on Machine Learning*, 2016.

Michael K. Pitt, Ralph dos Santos Silva, Paolo Giordani, and Robert Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 2012.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei. Deep survival analysis. In *Proceedings of the 1st Machine Learning for Healthcare Conference*, pages 101–114, 2016a.

Rajesh Ranganath, Dustin Tran, and David M. Blei. Hierarchical variational models. In *International Conference on Machine Learning*, 2016b.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Christian Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, 2004.

Francisco J. R. Ruiz, Michalis K. Titsias, and David M. Blei. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems*, 2016.

Ardavan Saeedi, Tejas D Kulkarni, Vikash Mansinghka, and Samuel Gershman. Variational particle approximations. *arXiv preprint arXiv:1402.5715*, 2014.

Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, 2015.

Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai. Sequential Monte Carlo methods for system identification. In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, Oct 2015.

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.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Dustin Tran, Rajesh Ranganath, and David M. Blei. The variational Gaussian process. In *International Conference on Learning Representations*, 2016.

PhD Dissertations
Division of Automatic Control
Linköping University

M. Millnert: Identification and control of systems subject to abrupt changes. Thesis No. 82, 1982. ISBN 91-7372-542-0.

A. J. M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis No. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis No. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis No. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis No. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis No. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis No. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis No. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis No. 194, 1988. ISBN 91-7870-380-8.

A. Isaksson: On system identification in one and two dimensions with signal processing applications. Thesis No. 196, 1988. ISBN 91-7870-383-2.

M. Viberg: Subspace fitting concepts in sensor array processing. Thesis No. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis No. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis No. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis No. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis No. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis No. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis No. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis No. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis No. 353, 1994. ISBN 91-7871-430-3.

K. Wang Chen: Transformation and symbolic calculations in filtering and control. Thesis No. 361, 1994. ISBN 91-7871-467-2.

T. McKelvey: Identification of state-space models from time and frequency data. Thesis No. 380, 1995. ISBN 91-7871-531-8.

J. Sjöberg: Non-linear system identification with neural networks. Thesis No. 381, 1995. ISBN 91-7871-534-2.

R. Germundsson: Symbolic systems – theory, computation and applications. Thesis No. 389, 1995. ISBN 91-7871-578-4.

P. Pucar: Modeling and segmentation using multiple models. Thesis No. 405, 1995. ISBN 91-7871-627-6.
H. Fortell: Algebraic approaches to normal forms and zero dynamics. Thesis No. 407, 1995. ISBN 91-7871-629-2.
A. Helmersson: Methods for robust gain scheduling. Thesis No. 406, 1995. ISBN 91-7871-628-4.
P. Lindskog: Methods, algorithms and tools for system identification based on prior knowledge. Thesis No. 436, 1996. ISBN 91-7871-424-8.
J. Gunnarsson: Symbolic methods and tools for discrete event dynamic systems. Thesis No. 477, 1997. ISBN 91-7871-917-8.
M. Jirstrand: Constructive methods for inequality constraints in control. Thesis No. 527, 1998. ISBN 91-7219-187-2.
U. Forssell: Closed-loop identification: Methods, theory, and applications. Thesis No. 566, 1999. ISBN 91-7219-432-4.
A. Stenman: Model on demand: Algorithms, analysis and applications. Thesis No. 571, 1999. ISBN 91-7219-450-2.
N. Bergman: Recursive Bayesian estimation: Navigation and tracking applications. Thesis No. 579, 1999. ISBN 91-7219-473-1.
K. Edström: Switched bond graphs: Simulation and analysis. Thesis No. 586, 1999. ISBN 91-7219-493-6.
M. Larsson: Behavioral and structural model based approaches to discrete diagnosis. Thesis No. 608, 1999. ISBN 91-7219-615-5.
F. Gunnarsson: Power control in cellular radio systems: Analysis, design and estimation. Thesis No. 623, 2000. ISBN 91-7219-689-0.
V. Einarsson: Model checking methods for mode switching systems. Thesis No. 652, 2000. ISBN 91-7219-836-2.
M. Norrlöf: Iterative learning control: Analysis, design, and experiments. Thesis No. 653, 2000. ISBN 91-7219-837-0.
F. Tjärnström: Variance expressions and model reduction in system identification. Thesis No. 730, 2002. ISBN 91-7373-253-2.
J. Löfberg: Minimax approaches to robust model predictive control. Thesis No. 812, 2003. ISBN 91-7373-622-8.
J. Roll: Local and piecewise affine approaches to system identification. Thesis No. 802, 2003. ISBN 91-7373-608-2.
J. Elbornsson: Analysis, estimation and compensation of mismatch effects in A/D converters. Thesis No. 811, 2003. ISBN 91-7373-621-X.
O. Härkegård: Backstepping and control allocation with applications to flight control. Thesis No. 820, 2003. ISBN 91-7373-647-3.
R. Wallin: Optimization algorithms for system analysis and identification. Thesis No. 919, 2004. ISBN 91-85297-19-4.
D. Lindgren: Projection methods for classification and identification. Thesis No. 915, 2005. ISBN 91-85297-06-2.
R. Karlsson: Particle Filtering for Positioning and Tracking Applications. Thesis No. 924, 2005. ISBN 91-85297-34-8.
J. Jansson: Collision Avoidance Theory with Applications to Automotive Collision Mitigation. Thesis No. 950, 2005. ISBN 91-85299-45-6.
E. Geijer Lundin: Uplink Load in CDMA Cellular Radio Systems. Thesis No. 977, 2005. ISBN 91-85457-49-3.
M. Enqvist: Linear Models of Nonlinear Systems. Thesis No. 985, 2005. ISBN 91-85457-64-7.
T. B. Schön: Estimation of Nonlinear Dynamic Systems — Theory and Applications. Thesis No. 998, 2006. ISBN 91-85497-03-7.

I. Lind: Regressor and Structure Selection — Uses of ANOVA in System Identification. Thesis No. 1012, 2006. ISBN 91-85523-98-4.

J. Gillberg: Frequency Domain Identification of Continuous-Time Systems Reconstruction and Robustness. Thesis No. 1031, 2006. ISBN 91-85523-34-8.

M. Gerdin: Identification and Estimation for Models Described by Differential-Algebraic Equations. Thesis No. 1046, 2006. ISBN 91-85643-87-4.

C. Grönwall: Ground Object Recognition using Laser Radar Data – Geometric Fitting, Performance Analysis, and Applications. Thesis No. 1055, 2006. ISBN 91-85643-53-X.

A. Eidehall: Tracking and threat assessment for automotive collision avoidance. Thesis No. 1066, 2007. ISBN 91-85643-10-6.

F. Eng: Non-Uniform Sampling in Statistical Signal Processing. Thesis No. 1082, 2007. ISBN 978-91-85715-49-7.

E. Wernholt: Multivariable Frequency-Domain Identification of Industrial Robots. Thesis No. 1138, 2007. ISBN 978-91-85895-72-4.

D. Axehill: Integer Quadratic Programming for Control and Communication. Thesis No. 1158, 2008. ISBN 978-91-85523-03-0.

G. Hendeby: Performance and Implementation Aspects of Nonlinear Filtering. Thesis No. 1161, 2008. ISBN 978-91-7393-979-9.

J. Sjöberg: Optimal Control and Model Reduction of Nonlinear DAE Models. Thesis No. 1166, 2008. ISBN 978-91-7393-964-5.

D. Törnqvist: Estimation and Detection with Applications to Navigation. Thesis No. 1216, 2008. ISBN 978-91-7393-785-6.

P-J. Nordlund: Efficient Estimation and Detection Methods for Airborne Applications. Thesis No. 1231, 2008. ISBN 978-91-7393-720-7.

H. Tidefelt: Differential-algebraic equations and matrix-valued singular perturbation. Thesis No. 1292, 2009. ISBN 978-91-7393-479-4.

H. Ohlsson: Regularization for Sparseness and Smoothness — Applications in System Identification and Signal Processing. Thesis No. 1351, 2010. ISBN 978-91-7393-287-5.

S. Moberg: Modeling and Control of Flexible Manipulators. Thesis No. 1349, 2010. ISBN 978-91-7393-289-9.

J. Wallén: Estimation-based iterative learning control. Thesis No. 1358, 2011. ISBN 978-91-7393-255-4.

J. D. Hol: Sensor Fusion and Calibration of Inertial Sensors, Vision, Ultra-Wideband and GPS. Thesis No. 1368, 2011. ISBN 978-91-7393-197-7.

D. Ankelhed: On the Design of Low Order H-infinity Controllers. Thesis No. 1371, 2011. ISBN 978-91-7393-157-1.

C. Lundquist: Sensor Fusion for Automotive Applications. Thesis No. 1409, 2011. ISBN 978-91-7393-023-9.

P. Skoglar: Tracking and Planning for Surveillance Applications. Thesis No. 1432, 2012. ISBN 978-91-7519-941-2.

K. Granström: Extended target tracking using PHD filters. Thesis No. 1476, 2012. ISBN 978-91-7519-796-8.

C. Lyzell: Structural Reformulations in System Identification. Thesis No. 1475, 2012. ISBN 978-91-7519-800-2.

J. Callmer: Autonomous Localization in Unknown Environments. Thesis No. 1520, 2013. ISBN 978-91-7519-620-6.

D. Petersson: A Nonlinear Optimization Approach to H2-Optimal Modeling and Control. Thesis No. 1528, 2013. ISBN 978-91-7519-567-4.

Z. Sjanic: Navigation and Mapping for Aerial Vehicles Based on Inertial and Imaging Sensors. Thesis No. 1533, 2013. ISBN 978-91-7519-553-7.

F. Lindsten: Particle Filters and Markov Chains for Learning of Dynamical Systems. Thesis No. 1530, 2013. ISBN 978-91-7519-559-9.

P. Axelsson: Sensor Fusion and Control Applied to Industrial Manipulators. Thesis No. 1585, 2014. ISBN 978-91-7519-368-7.

A. Carvalho Bittencourt: Modeling and Diagnosis of Friction and Wear in Industrial Robots. Thesis No. 1617, 2014. ISBN 978-91-7519-251-2.

M. Skoglund: Inertial Navigation and Mapping for Autonomous Vehicles. Thesis No. 1623, 2014. ISBN 978-91-7519-233-8.

S. Khoshfetrat Pakazad: Divide and Conquer: Distributed Optimization and Robustness Analysis. Thesis No. 1676, 2015. ISBN 978-91-7519-050-1.

T. Ardeshiri: Analytical Approximations for Bayesian Inference. Thesis No. 1710, 2015. ISBN 978-91-7685-930-8.

N. Wahlström: Modeling of Magnetic Fields and Extended Objects for Localization Applications. Thesis No. 1723, 2015. ISBN 978-91-7685-903-2.

J. Dahlin: Accelerating Monte Carlo methods for Bayesian inference in dynamical models. Thesis No. 1754, 2016. ISBN 978-91-7685-797-7.

M. Kok: Probabilistic modeling for sensor fusion with inertial measurements. Thesis No. 1814, 2016. ISBN 978-91-7685-621-5.

J. Linder: Indirect System Identification for Unknown Input Problems: With Applications to Ships. Thesis No. 1829, 2017. ISBN 978-91-7685-588-1.

M. Roth: Advanced Kalman Filtering Approaches to Bayesian State Estimation. Thesis No. 1832, 2017. ISBN 978-91-7685-578-2.

I. Nielsen: Structure-Exploiting Numerical Algorithms for Optimal Control. Thesis No. 1848, 2017. ISBN 978-91-7685-528-7.

D. Simon: Fighter Aircraft Maneuver Limiting Using MPC: Theory and Application. Thesis No. 1881, 2017. ISBN 978-91-7685-450-1.

C. Veibäck: Tracking the Wanders of Nature. Thesis No. 1958, 2018. ISBN 978-91-7685-200-2.