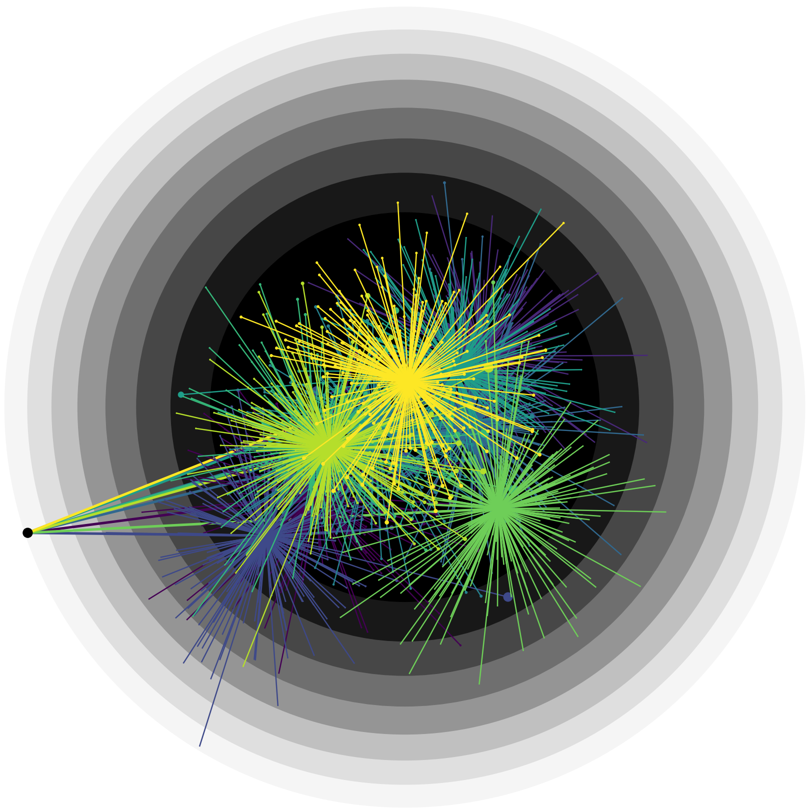
Department of Electrical Engineering and Automation

Computationally efficient statistical inference in Markovian models

Adrien Corenflos



Aalto University

DOCTORAL THESES

Aalto University publication series
DOCTORAL THESES 135/2024

Computationally efficient statistical inference in Markovian models

Adrien Corenflos

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering.
The public defence will take place on 26 August 2024 at noon (EEST) in lecture hall E (Otakaari 1, Espoo), and it will be broadcast online via the remote connection link https://aalto.zoom.us/j/64211832840.

Aalto University
School of Electrical Engineering
Department of Electrical Engineering and Automation
Sensor Informatics and Medical Technology

Supervising professor
Professor Simo Särkkä, Aalto University, Finland

Thesis advisor
Professor Simo Särkkä, Aalto University, Finland

Preliminary examiners
Doctor Michalis Titsias, Google DeepMind, United Kingdom
Professor Fredrik Lindsten, Linköping University, Sweden

Opponent
Professor Omiros Papaspiliopoulos, Bocconi University, Italy

Aalto University publication series
DOCTORAL THESES 135/2024

ISBN 978-952-64-1900-8 (printed)
ISBN 978-952-64-1901-5 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-64-1901-5

Images: Several realisations of an ensemble auxiliary MALA proposal for a centred Gaussian target distribution.

Unigrafia Oy
Helsinki 2024

Finland

A!
Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Adrien Corenflos

Name of the doctoral thesis	Computationally efficient statistical inference in Markovian models
Publisher	School of Electrical Engineering
Unit	Department of Electrical Engineering and Automation
Series	Aalto University publication series DOCTORAL THESES 135/2024
Field of research	
Manuscript submitted	4 March 2024
Date of the defence	26 August 2024
Permission for public defence granted (date)	10 May 2024
Language	English

☐ **Monograph** ☒ **Article thesis** ☐ **Essay thesis**

Abstract

Markovian systems are ubiquitous in nature, science, and engineering, to model the evolution of a system for which the future state of the system only depends on the past through the present state. These often appear as time series or stochastic processes, and when they are partially observed, they are known under the umbrella term of state-space models. Inferring the current state of the system from these partial, and often noisy, observations is a fundamental question in statistics and machine learning, and it is often solved using Bayesian inference methods that correct a prior belief on the state of the system through the likelihood of the observations. This perspective gives rise to typically recursive algorithms, which sequentially process the observations to slowly refine the estimate of the current state of the system. The most common of these algorithms are the Kalman filter and its extensions via linearisation procedures, and particle filtering methods, based on Monte Carlo. Another question, which often arises is that of the past state or past trajectory of the system, given all the observations. Furthermore, it may also be of interest to identify the model itself, whereby the most likely (or any other metric) model within a family is picked given the observations

In this thesis, we examine the three problems of Bayesian filtering, smoothing, and identification in the context of Markovian models, and we propose computationally efficient algorithms to solve them. In particular, we develop the parallelisation of the recursive structure of the filtering-smoothing algorithms, which, while optimal in a sequential setting, can be significantly sped up by using modern parallel computing architectures. This endeavour is tackled in both the context of particle approximations and Kalman-related methods. Another important aspect of the thesis is the use of gradient-based methods to perform inference in state-space models, taking several forms. One of these is the generalisation of the Metropolis-adjusted Langevin algorithm (MALA) and related algorithms to the context of particle and Kalman filters, and their implication for high-dimensional state inference. Another one is making particle filters differentiable by approximating the usual algorithm and then using the approximation to perform inference in state-space models using gradient-based methods. Finally, we also discuss the use of gradient-flows to perform automatic locally optimal filtering in state-space models. Some of these algorithms are *de facto* sequential and hardly parallelisable, but some instances can benefit from parallelisation, and we discuss the implications of this in terms of computational efficiency.

Keywords State-space models, Markov chain Monte Carlo, sequential Monte Carlo, particle filtering, variational inference, parallel computing

ISBN (printed)	978-952-64-1900-8	**ISBN (pdf)**	978-952-64-1901-5
ISSN (printed)	1799-4934	**ISSN (pdf)**	1799-4942
Location of publisher	Helsinki	**Location of printing**	Helsinki
Year	2024		
Pages	359	**urn**	http://urn.fi/URN:ISBN:978-952-64-1901-5

Preface

"I love deadlines. I like the whooshing sound they make as they fly by."

Douglas Adams

The research presented in this thesis was conducted under the supervision of Prof. Simo Särkkä between September 2020 and January 2024 at the Department of Electrical Engineering and Automation of Aalto University, Finland; the thesis itself was written in January and February 2024. During this time, my research was successively funded by the Academy of Finland (project ADAFUME) and Aalto ELEC Doctoral School.

This thesis is the result of a lot more efforts than just my own, and I would be remiss to not mention the people who made it happen. Perhaps the greatest thanks goes to my collaborators, who have been instrumental in the development of the ideas presented in this thesis, first and foremost my supervisor, Prof. Simo Särkkä. His continued insistance on hardware-acceleration and parallelisation has now become a core part of who I am and who I will be as a researcher, and I cannot thank him enough for opening the door to this world. The four years I spent at Aalto University are some of the most formative years of my life, and I am very grateful to Simo for making them happen. I also want to thank Prof. Arnaud Doucet, who was instrumental in the beginning of my academic career, essentially giving me the first chance to prove myself, when he had no reason to do so other than a cold email. The work we did together with Dr. James Thornton and Prof. George Deligiannidis is a landmark in my academic career, and I am very grateful for the opportunity to work with them. The same goes for Prof. Nicolas Chopin, who too, decided I was worth the time of day from just a few GitHub commits, and who has been a great mentor and friend since then, our continued collaboration is a great source of joy for me. Finally, I would like to thank Axel Finke, for getting excited about the same things I did, so much so that he felt like writing a paper with me.

Additionally to these, a special thanks goes to this thesis examiners, Prof. Fredrik Lindsten and Dr. Michalis Titsias, for agreeing to read this thesis and provide feedback on it, as well as to Prof. Omiros Papaspiliopoulos, for agreeing

to be the opponent for my defence.

Moving to a new country is never easy, and I am very grateful to the people who made life a pleasure on an everyday basis. Some of them are collaborators too, but they held such daily importance in my life that I prefer to mention them here: Prof. Christos Merkatas, for three long years of friendship and support, I hope we can have a frappé in Athens soon; Dr. Zheng Zhao, for being a good sport and a great collaborator, I hope I left you with more than a taste for chess and mushroom picking; Fatemeh Yaghoobi, for bearing to share an office with me for three years, and for being a great friend and collaborator, please forgive me for changing your wallpaper just before writing this; Linda Srbova, for too, sharing this office, and for always smiling when you lost my USB cables, I hope you find them soon, they're yours now; Dr. Hany Abdulsamad, for being a great friend and collaborator, for all the great discussions and the great times; Frieda, for all the belly rubs and the kisses, I hope Hany keeps taking good care of you; Sahel Iqbal, for his enthusiasm and energy, I hope we can continue our collaboration in the future, I think you'll do great; Casian Iacob, for being graceful when he beat me at a 10k run, don't worry, I'll get you next time; Émilie Calmettes, because finding that I had a school friend in a foreign country weirdly helped a lot. All my thanks and love also go to these other people that made work a nicer place to be: Cristian, Dennis, Ivan, Otto, Wendy, Zaeed, and all the ones I forget to mention. Additionally, I would like mention my former (pre-academic) colleagues, who taught me much over the years, and who I hope are doing well: Christophe Mignard, Édouard Ruiz, Dr. Charles Mathias, some of the stuff in here happened because of you (it is not your fault though!). Thanks, thanks, thanks!

I would also like to thank my friends and family: even if I always talk to, visit, and see you less than I should, let it be known that I love you all very much, and that I am very grateful for your support and love over the years. Papa, Clarisse, merci pour tout.

Finally, thank you Rachel, for being my life partner, my best friend, and my greatest support and source of strength, I love you very much, and I cannot wait to come back to you.

London and Espoo, July 23, 2024,

Adrien Corenflos

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Adrien Corenflos, James Thornton, George Deligiannidis, Arnaud Doucet. Differentiable Particle Filtering via Entropy-Regularized Optimal Transport. In *Proceedings of the 38th International Conference on Machine Learning*, Volume 139, Pages 2100–2111, July 2021.

II Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä. Parallel Iterated Extended and Sigma-Point Kalman Smoothers. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Pages 5350–5354, June 2021.

III Adrien Corenflos, Zheng Zhao, and Simo Särkkä. Temporal Gaussian Process Regression in Logarithmic Time. In *Proceedings of the 2022 International Conference on Information Fusion (FUSION)*, Linköping, Sweden, Pages 1–5, July 2022.

IV Adrien Corenflos, Nicolas Chopin, and Simo Särkkä. De-Sequentialized Monte Carlo: a parallel-in-time particle smoother. *Journal of Machine Learning Research*, Volume 23, Number 283, Pages 1–39, August 2022.

V Adrien Corenflos and Simo Särkkä. Auxiliary MCMC samplers for parallelisable inference in high-dimensional latent dynamical systems. Submitted to *Electronic Journal of Statistics*, July 2023.

VI Adrien Corenflos and Hany Abdulsamad. Variational Gaussian filtering via Wasserstein gradient flows. In *Proceedings of the 31st European Signal Processing Conference (EUSIPCO)*, Helsinki, Finland, Pages 1838–1842, September 2023.

VII Adrien Corenflos and Axel Finke. Particle-MALA and Particle-mGRAD: Gradient-based MCMC methods for high-dimensional state-space models. Submitted to *Journal of Machine Learning Research*, January 2024.

Author's Contribution

Publication I: "Differentiable Particle Filtering via Entropy-Regularized Optimal Transport"

The original idea for this article comes from discussions between Adrien Corenflos and Arnaud Doucet. The methodological development and original implementation are primarily due to Adrien Corenflos with critical inputs from Arnaud Doucet. Experimental evaluation was conducted by Adrien Corenflos and James Thornton. The theoretical guarantees were originally developed by James Thornton and later improved upon by George Deligiannidis. The first draft of the article was written by Adrien Corenflos and James Thornton, substantially revised by Arnaud Doucet, and reviewed by George Deligiannidis. The project was lead by Arnaud Doucet.

Publication II: "Parallel Iterated Extended and Sigma-Point Kalman Smoothers"

The writing of this article was primarily done by Fatemeh Yaghoobi following the original idea of Simo Särkkä. The methodological development was done by Fatemeh Yaghoobi and Simo Särkkä. The implementation is due to Adrien Corenflos, and subsequent experiments were conducted by Fatemeh Yaghoobi and Adrien Corenflos. The project was supervised by Simo Särkkä and Sakira Hassan.

Publication III: "Temporal Gaussian Process Regression in Logarithmic Time"

The methodological development, writing, implementation, and experimental evaluation are primarily due to Adrien Corenflos with substantial involvement from Zheng Zhao. The original idea was due to Simo Särkkä.

Publication IV: "De-Sequentialized Monte Carlo: a parallel-in-time particle smoother"

The writing and experimental evaluation are due to Adrien Corenflos. The methodological and theoretical development is primarily due to Adrien Corenflos with inputs from Nicolas Chopin. The original idea for this article comes from discussions between Adrien Corenflos and Simo Särkkä. Detailed contributions are given in the article.

Publication V: "Auxiliary MCMC samplers for parallelisable inference in high-dimensional latent dynamical systems"

The original idea, methodology, implementation, and redaction of the first version of this article are due to Adrien Corenflos. Simo Särkkä contributed the divide-and-conquer sampling algorithm and reviewed the final version of the manuscript.

Publication VI: "Variational Gaussian filtering via Wasserstein gradient flows"

The original idea and redaction of the article are due to Adrien Corenflos. Both authors contributed to the design of the methodology. The implementation and experiments are due to Hany Abulsamad.

Publication VII: "Particle-MALA and Particle-mGRAD: Gradient-based MCMC methods for high-dimensional state-space models"

Adrien Corenflos and Axel Finke jointly developed the methodology, writing was primarily done by Axel Finke, Adrien Corenflos implemented and conducted the experiments, after which both Adrien Corenflos and Axel Finke edited and reviewed the final manuscript.

List of Figures and Tables

Abbreviations

a.e.	Almost every
a.s.	Almost surely
CLT	Central limit theorem
CPU	Central processing unit
CSMC	Conditional sequential Monte Carlo
DnC	Divide-and-conquer
e.g.	*Exempli gratia* (for example)
ELBO	Evidence lower bound
EM	Expectation–maximisation
ESS	Effective sample size
ETR	Ensemble transform resampling
GPU	Graphics processing unit
GSLR	Generalised statistical linear regression
HMC	Hamiltonian Monte Carlo
IAT	Integrated autocorrelation time
i.e.	*Id est* (that is)
i.i.d.	Independent and identically distributed
IEKS	Iterated extended Kalman smoother
IPLS	Iterated posterior linearisation smoother
IS	Importance sampling

KF	Kalman filter
KL	Kullback–Leibler divergence
LGSSM	Linear Gaussian state-space model
LLN	Law of large numbers
MALA	Metropolis-adjusted Langevin algorithm
MAP	Maximum a posteriori
MCMC	Markov chain Monte Carlo
MH	Metropolis–Hastings
MLE	Maximum likelihood estimation
MSE	Mean squared error
ODE	Ordinary differential equation
PF	Particle filter
PIT	Parallel-in-time
PIMH	Particle independent Metropolis–Hastings
PMCMC	Particle Markov chain Monte Carlo
PMMH	Particle marginal Metropolis–Hastings
PS	Particle smoother
RTS	Rauch–Tung–Striebel
SDE	Stochastic differential equation
SIR	Sampling importance resampling
SIS	Sampling importance sampling
SMC	Sequential Monte Carlo
SNIS	Self-normalised importance sampling
TPU	Tensor processing unit
TV	Total variation distance
VSMC	Variational sequential Monte Carlo

Symbols

The following notations are used throughout the thesis (in “alphabetical” order).

A^{-1}	Inverse of a matrix A.
$A^\top$	Transpose of a matrix A.
$A^{1/2}$	Cholesky lower triangular decomposition of a Hermitian matrix A.
$A^{-1/2}$	Inverse Cholesky decomposition of a Hermitian matrix A.
$A^{\top/2}$	Transpose Cholesky decomposition of a Hermitian matrix A.
a.s., a.e.	Almost surely, almost everywhere, i.e., with probability one. For example, $X = Y$ a.s. means that $\mathbb{P}(X = Y) = 1$. When context requires it, we will specify the measure with which this even occurs with $\mathbb{P}$-a.s. or $\mathbb{P}$-a.e.
$\mathbb{C}[X,Y]$	Covariance matrix of two random variables X and Y: $\mathbb{C}[X,Y] = \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])^\top\right]$.
$\mathbb{C}[X]$	Shorthand for $\mathbb{C}[X,X]$.
$\mathscr{D}$	Divergence.
$\mathscr{D}_{\mathbf{KL}}(\cdot \parallel \cdot)$	Kullback–Leibler divergence.
$\mathscr{D}_{\mathbf{TV}}(\cdot \parallel \cdot)$	Total variation divergence/distance.
$\mathbb{E}$	Expectation operator. When the distribution of X is unclear from context, we will write $\mathbb{E}_p[X]$ instead of $\mathbb{E}[X]$.
$\mathbb{E}[X \mid Y]$	Conditional expectation of the random variable X given the random variable Y.
$M(\cdot \mid x)$	This will most often denote a Markov kernel, i.e., a function $M : \mathrm{X} \to \mathscr{P}(\mathrm{X})$, mapping $x \in \mathrm{X}$ to a probability measure $M(\cdot \mid x)$ on X.

$M(\pi)$	Given a probability measure π and a Markov kernel $M(\cdot \mid x)$, $M(\pi)$ denotes the push-forward measure of π under M, i.e., $M(\pi)(A) = \int M(A \mid x)\pi(\mathrm{d}x)$.
$\mathcal{N}(b,C)$	Gaussian distribution with mean b and covariance C.
$\mathcal{N}_I(b,C)$	Gaussian distribution given in its information form: $\mathcal{N}_I(b,C) = \mathcal{N}(C^{-1}b, C^{-1})$.
$\mathcal{N}(a;b,C)$	Density of a Gaussian distribution with mean b and covariance C evaluated at a.
$\mathcal{N}_I(a;b,C)$	Density of a Gaussian distribution given in its information form: $\mathcal{N}_I(a;b,C) = \mathcal{N}(a;C^{-1}b;C^{-1})$.
$\mathbb{N}$	The set of natural numbers $\{1,2,\ldots\}$.
$\mathbb{N}_0$	The set of non-negative integers $\{0,1,2,\ldots\}$.
$\mathcal{O}(f)$	big-$\mathcal{O}$ notation, i.e., dominated by $C\lvert f\rvert$, where $C > 0$ is constant. This is often used to denote the order of convergence of an algorithm or the complexity of an algorithm, where f is a function of the problem size. Formally, $\mathcal{O}(f(n))$ is the set of functions $\{g : \exists C > 0, \exists N \in \mathbb{N}, \forall n \geq N, \lvert g(n)\rvert \leq C\lvert f(n)\rvert\}$.
$\mathbb{P}$	A probability measure on an abstract space $(\Omega, \mathcal{F})$, often used to denote the probability of a given event: $\mathbb{P}(A)$.
$\mathcal{P}(X)$	Set of probability measures on X, the underlying probability space is typically clear from context.
$\mathrm{prox}_{\gamma\mathcal{L}}(\phi)$	Proximal operator of the function $\mathcal{L}$, with regularisation parameter γ at ϕ with respect to a given distance d: $$\mathrm{prox}_{\gamma\mathcal{L}}(\phi) = \arg\min_{\psi\in\Phi}\left\{\mathcal{L}(\psi) + \frac{1}{2\gamma}d(\psi,\phi)^2\right\}.$$
$\mathbb{R}^d$	The d-dimensional Euclidean space. When $d = 1$, we often write $\mathbb{R}$ instead of $\mathbb{R}^1$.
$\mathcal{W}(p,q)$	Wasserstein distance between the probability measures p and q: $\mathcal{W}(p,q)^2 = \inf_{\pi\in\Pi(p,q)} \int_{\mathrm{X}\times\mathrm{X}} \lVert x,y\rVert_2^2\, \pi(\mathrm{d}x,\mathrm{d}y)$ where $\Pi(p,q) \subset \mathcal{P}(\mathrm{X}\times\mathrm{X})$ is the subset of joint distributions on $\mathrm{X}\times\mathrm{X}$ with marginals p and q.
$X \sim \pi$	A random variable X with distribution π.
$x_{a:b}$	A sequence of values $(x_a, x_{a+1}, \ldots, x_{b-1}, x_b)$, $\emptyset$ if $a > b$. Often used for 'time' indices.
$x^{1:N}$	A sequence of values $(x^1, x^2, \ldots, x^{N-1}, x^N)$. Often used for 'particle' indices.

X	The space in which the latent variables of a model take values. It is often assumed to be equipped with a σ-algebra $\mathscr{X}$ and a reference measure λ. Often, X is a subset of $\mathbb{R}^d$ for some $d \in \mathbb{N}$.
Y	The space in which the observations of a model take values. It is often assumed to be equipped with a σ-algebra $\mathscr{Y}$ and a reference measure μ. Often, Y is a subset of $\mathbb{R}^m$ for some $m \in \mathbb{N}$.
$\mathbb{Z}$	The set of integers $\{\ldots -2, -1, 0, 1, 2, \ldots\}$.
$\delta_a(\mathrm{d}x)$	The Dirac measure at a, i.e., if $A \subseteq \mathsf{X}$, then $\delta_a(A) = \mathbb{1}_{a \in A}$; informally, $X \sim \delta_a$ if and only if $\mathbb{P}(X = a) = 1$.
$\pi(\mathrm{d}x)$	A measure on the measurable space $(\mathsf{X}, \mathscr{X})$, the notation is equivalent to $\mathrm{d}\pi(x)$ corresponding to the volume of an infinitesimal region around x.
$\pi(f)$	Given a probability measure π and an integrable function f, $\pi(f)$ denotes the expectation of f under π, i.e., $\pi(f) = \int f(x)\pi(\mathrm{d}x)$.
$\pi(x) \propto f(x)$	Equal up to a multiplicative constant; often this implicitly corresponds to the stronger $\pi(x) = \frac{1}{Z} f(x)$ for some normalising constant Z such that $\int \pi(x)\mathrm{d}x = 1$.
$\lVert x \rVert_A$	Mahalanobis norm of x with respect to A, i.e., $\lVert x \rVert_A = \sqrt{x^\top A^{-1} x}$.
$\lVert \cdot \rVert_{\mathbf{TV}}$	Total variation norm.
$\begin{pmatrix} A & B \\ C & D \end{pmatrix}$	Block matrix with blocks A, B, C, and D of appropriate dimensions.

A few remarks about the assumptions and conventions used in the thesis are in order.

In this thesis we assume that all functions considered are measurable with respect to the space at hand. For example, if X is a subset of $\mathbb{R}^d$, then we assume that all functions $f : \mathsf{X} \to \mathbb{R}$ are Borel measurable. We also make liberal use of the measure-theoretic notation $\pi(\mathrm{d}x)$. While this notation is often not necessary (and when it is, we will make sure to explain why), it is used to emphasise what object is being sampled or integrated over: the following are equivalent

$$\pi(f) = \int f(x)\mathrm{d}\pi(x) = \int f(x)\pi(\mathrm{d}x) = \mathbb{E}_\pi[f(X)] = \mathbb{E}_\pi[f],$$

denoting the expectation of f under the probability measure π. When it exists, we also often equate the density (or probability mass function) of a measure with the measure itself, i.e., we write $\pi(x) \equiv \pi(\mathrm{d}x)$. The reference measure is often clear from context, and will always be either the Lebesgue measure or the counting measure on a discrete space (or a combination of these, e.g., in Chapter 4). Consequently, we typically assume that random variables are defined on an abstract probability space $(\Omega, \mathscr{F}, \mathbb{P})$ and take values in a measured space $(\mathrm{X}, \mathscr{X}, \lambda)$, where λ is the reference measure with respect to which the probability measure $\pi = (A \mapsto \mathbb{P}(X \in A))$ is assumed to have a density: $\pi(A) = \int_A \pi(x)\lambda(\mathrm{d}x)$. These notations are seldom used in the thesis. The language of the thesis follows from these conventions, and we often talk about measures, distribution, and densities interchangeably when context permits.

1. Introduction

This thesis comprises this short introduction, a background to the research field, as well as Publication I to VII, which are appended at the end of the thesis. The background serves as a brief overview of the field of (i) Bayesian inference (Chapter 2), (ii) Markovian models, such as state-space models and inference methods for these models (Chapter 3), (iii) Markov chain Monte Carlo (MCMC) methods, with a focus on their instantiation for Markovian models (Chapter 4), and (iv) parallelism in Bayesian inference and computation (Chapter 5), which the author thought would be helpful to understand the context and contributions of the publications. The publications themselves and their contributions in relation to the topics introduced in the rest of the thesis are then presented in Chapter 6, and the thesis concludes with an outlook on possible future research directions.

1.1 Computing Bayes

This section takes its title from the excellent review Martin et al. (2023a,b), which provides a historical account of the development of computational methods for Bayesian inference from Thomas Bayes's original manuscript to the present day. Bayesian inference is now a well-established field of statistics, and is often contrasted with the frequentist approach to statistics. Where frequentist statistics aim at providing a definitive answer to a certain question (also known as hypothesis testing), the Bayesian paradigm focuses on propagating uncertainty on the answer through the entire statistical pipeline, from the data collection to the final inference and eventual decision. This may be seen as a philosophical difference, but it has important practical consequences, and in particular, it requires the development of new computational methods to solve the problems that arise from the propagation of uncertainty. The development of these methods has been the focus of a large part of the statistical community for the past 50 years, and has led to the development of a wide range of algorithms and methods, the breadth of which is a testimony to the applicability of the framework. Some of these methods are the focus of this thesis.

1.2 Markovian models

Markovian models describe systems that evolve in time, and are characterised by the Markov property, which states that the future state of the system only depends on the past via its current state. For example, the drunkard's walk is a simple Markovian model, when you blink your eyes, the drunkard could move to the left or to the right, but their past movements do not affect their future movements. The Markov property is a strong assumption: most processes do exhibit a form of memory (or momentum), but it is also a very useful one, and has led to the development of a wide range of models and methods for inference in these models. Nonetheless, one may often find that the Markov property holds for an augmented version of the system: for example, the drunkard's walk may be augmented with their current direction of movement, and then the augmented system would be (again approximately) Markovian. This can be taken further, and the augmented system may be augmented again, and so on, until the modeller is comfortable with assuming that the system is Markovian. In particular, navigation systems usually follow a form of Markovian property: the next position only depends on the current position (and on the current speed), and not on the past positions. This is the foundation of the Kalman filter (Kalman, 1960), which took a central place in the navigation system of the Apollo missions to the moon, and is still used (in one form or another) in most navigation systems today.

1.3 Overview and outline of the thesis

This thesis is interested in the development of new algorithms for inference in Markovian models, some of them based on the Kalman filter, and some of them based on other methods, typically relying on simulating *a priori* possible trajectories of the system, and then correcting for mistakes made in the simulation. Such methods are often referred to as particle filters (Gordon et al., 1993), and are the focus of a large part of this thesis. In this thesis, a particular focus is put on computational efficiency, and on the use of parallel hardware such as general purpose graphics processing units (GPUs) to speed up the computations. Informally, some methods presented in this thesis were developed with a 'hardware-first' mindset, which is a departure from the traditional 'method-first' mindset of the statistical community, and is more in line with the approach of the deep learning community.[1] This sometimes comes at the cost of statistical efficiency, and some effort is made to understand the trade-offs between the two.

The remainder of this thesis is consequently organised as follows.

Chapter 2 introduces the general problem of Bayesian inference, and discusses

[1]In the AI community this is often taken to the extreme as scale is all you need, accessed February 5th, 2024

different approaches to solving it. This chapter is intended as both a compendium of the different basic tools and concepts used throughout the thesis, as well as a way to introduce notation and conventions.

Chapter 3 formally introduces Markovian models, which most of this thesis is concerned with, including state-space models, hidden Markov models, and Feynman–Kac models. Standard techniques for inference in these models are also introduced, including Gaussian (approximated) filters and smoothers such as the Kalman filter (KF), as well as sequential Monte Carlo methods. The latter are also known as particle filters and smoothers, or, as we call them in this thesis, sequential importance resampling (SIR) methods.

Chapter 4 introduces Markov chain Monte Carlo (MCMC) methods, which are a class of algorithms for sampling from probability distributions, and are a central tool in Bayesian inference. We specifically focus on its state-of-the-art instantiation to Bayesian inference in state-space models, namely particle MCMC algorithms. Some effort is additionally made to inscribe all the algorithms presented in this chapter in the general framework of auxiliary variable methods, which is a rich and successful framework for constructing MCMC algorithms.

Chapter 5 discusses parallelism in Bayesian inference and computation, with a particular focus on prefix-sums and their use in parallel algorithms, as well as on divide-and-conquer methodologies. Attention is given to separating two types of parallelism: computational parallelism, which does not affect statistical properties of the algorithms, and statistical parallelism, which does.

Chapter 6 offers a summary of the contributions of the seven publications included in this thesis, and links them to the different topics introduced in the previous chapters. Additionally, we give a highlight on Black-JAX (Cabezas et al., 2024), an open-source library for Bayesian inference which the author of this thesis is an active contributor to. Finally, we also offer an outlook on possible ongoing relevant research directions.

The key dependencies between the different chapters and publications are summarised in Table 1.1.

Table 1.1. Dependencies between the different chapters and publications. *Strong* dependencies are those that are critical to understand the core content/contribution, while *soft* dependencies are those that are helpful but not necessary for a first read.

Part	**Strong dependency**	**Soft dependency**
Chapter 2		
Chapter 3	Chapter 2	
Chapter 4	Chapter 3	Chapter 2
Chapter 5	Chapters 3, 4	Chapter 2
Publication I	Chapter 3	
Publication II	Chapters 3, 5	
Publication III	Chapter 5	Chapters 3, 4
Publication IV	Chapter 5, 4	Chapter 3, Publication II
Publication V	Chapter 4, Publications II, IV	Chapter 5
Publication VI	Chapters 2, 3	
Publication VII	Chapter 3, Publication V	Chapters 2, 5

2. Statistical inference and Computing Bayes

Statistical inference is rooted in the problem of decision-making under uncertainty, whereby a decision-maker is faced with a choice between several options, and must choose the one that is most likely to lead to a desirable outcome. In the context of statistical decision theory (Berger, 2013), this is expressed as a set of possible states of the world X, a set of possible actions $\mathscr{A}$, and a loss function $\ell \colon \mathrm{X} \times \mathscr{A} \to \mathbb{R}$ that quantifies the loss incurred by taking action $a \in \mathscr{A}$ when the true state of the world is $x \in \mathrm{X}$. Often, the decision-maker has access to an observation $y \in \mathrm{Y}$ that is related to the state of the world x through a statistical conditional model $p(y \mid x)$, and the goal of the Bayesian statistician is then to choose an action $a \in \mathscr{A}$ that minimises the expected loss $L(a) := \mathbb{E}[\ell(x,a)]$ under the posterior distribution

$$p(\mathrm{d}x \mid y) = \frac{p(y \mid x) p(\mathrm{d}x)}{p(y)}. \tag{2.1}$$

Here $p(\mathrm{d}x)$ is a prior distribution on the state of the world, corresponding to the decision-maker's beliefs about the state of the world before observing y, and $p(y) = \int p(y \mid x) p(\mathrm{d}x)$ is the marginal likelihood of the observation y, also known as the *normalising* constant. A classical example of loss function is the squared loss $\ell(x,a) = \|x - a\|^2$, which corresponds to minimising the squared error between a state-estimate a and the state of the world x, leading to the well-known posterior mean estimator $\hat{x} = \mathbb{E}[x \mid y]$. Similarly, for discrete state spaces, the loss function $\ell(x,a) = \mathbb{1}_{x \neq a}$, which informally corresponds to minimising the probability of an error, leads to the well-known posterior mode estimator $\hat{x} = \arg\max_x p(x \mid y)$, also known as the maximum a posteriori (MAP) estimator. This formulation of statistical inference both encompasses state and parameter estimation, as the state of the world x can be seen as a parameter of the conditional model $p(y \mid x)$, so that the so-called maximum likelihood estimator (MLE) $\hat{x} = \arg\max_x p(y \mid x)$ is a special case of the MAP estimator for an improper prior $p(x) \equiv 1$.

The problem of computing expectations (computing Bayes) with respect to the posterior distributions (2.1) is therefore at the heart of decision-making and statistical inference in general, and is the focus of this thesis. Computing Bayes (Martin et al., 2023b,a) has a long-standing, rich history in computational statistics, and can be broadly divided into three classes: (i) exact and conjugate methods, (ii) simulation methods, and (iii) variational methods. We review these three classes of methods which are used throughout this thesis, and provide a brief overview of the main ideas and results that are relevant for understanding the contributions therein. Throughout this chapter, $\pi(\mathrm{d}x)$ will denote a target distribution, with respect to which we want to compute an integral, or from which we want to (approximately) sample. It will often be given as the posterior distribution (2.1), but the methods we review are general and can be applied to most target distributions of interest.

2.1 Exact and conjugate methods

For a small set of models, the posterior distribution (2.1) can be computed in closed form, and the resulting algorithms are known as exact methods. While these models are of limited use in practice, they are often used as building blocks for more complex approximate models, and are therefore of great practical importance.

2.1.1 Finite models

Perhaps the simplest model for which the posterior distribution (2.1) can be computed in closed form is the finite-state model, where the world X is represented as a finite set $\mathsf{X} = \{x_1, \dots, x_K\}$, which, without loss of generality, can be assumed to be the integers $\mathsf{X} = \{1, \dots, K\}$. In this case, the prior distribution $p(\mathrm{d}x)$ is represented as a probability mass vector $p(x) = p_x \geq 0$ on X defined by the mass it assigns to each element $x \in \mathsf{X}$ and verifying $\sum_{x\in\mathsf{X}} p_x = 1$.

In this case, when the observation y is a single realisation of the random variable $Y \sim p(\mathrm{d}y \mid x)$, the posterior distribution (2.1) is a probability mass function on X defined by

$$\begin{aligned} p(x \mid y) &\propto p(y \mid x) p_x, \\ &= \frac{p(y \mid x) p_x}{\sum_{z\in\mathsf{X}} p(y \mid z) p_z}, \end{aligned} \tag{2.2}$$

where $\propto$ denotes equality up to a multiplicative constant, in this case the normalising constant $p(y) = \sum_{z\in\mathsf{X}} p(y \mid z) p_z$.

Clearly, when the state of the world x belongs to a finite set, and provided that $p(y \mid x)$ is known, the posterior distribution (2.2) can be computed exactly. In particular, the posterior mean and mode estimators are given by

$$\hat{x} = \mathbb{E}[x \mid y] = \sum_{x\in\mathsf{X}} x p(x \mid y), \quad \hat{x} = \arg\max_{x\in\mathsf{X}} p(x \mid y), \tag{2.3}$$

respectively. While Bayesian inference can *in theory* be performed exactly for finite models, two cases may arise in practice that make exact inference intractable.

First, the number of states K may be too large to allow for practical inference: for example, if the state of the world is represented by a binary vector of length K, then the number of possible states is 2^K, which makes computing the normalising constant $p(y)$ (and subsequent expectations) exactly implausible for all but the smallest values of K. This regime often arises in the context of probabilistic graphical models (Koller and Friedman, 2009), or particle physics (Taroni, 2015), which often involve world-states that encompass millions, or more, possible configurations. In this case, one must resort to approximate methods, such as the ones discussed in Section 2.3 and Section 2.2. Second, the likelihood $p(y \mid x)$ can itself be intractable, which may encompass at least one of three recurring cases: (i) the model $p(\mathrm{d}y \mid x)$ can only be simulated, (ii) the likelihood $p(y \mid x) \propto h(y \mid x)$ is known up to a normalising constant, or (iii) only an (unbiased) estimator $\hat{p}(y \mid x, \omega)$ is known, with $\mathbb{E}[\hat{p}(y \mid x, \omega)] = p(y \mid x)$. The (i) first instance is usually treated by a class of computation methods known as approximate Bayesian computation (ABC, Sisson et al., 2018), the (ii) second instance is known as doubly-intractable models (Møller et al., 2006; Murray et al., 2006), and the (iii) third instance gives rise to pseudo-marginal methods (Andrieu and Roberts, 2009), all of these typically falling under the umbrella of simulation methods, which we review briefly in Section 2.2.

2.1.2 Conjugate models

A second class of models for which the posterior distribution (2.1) can be computed in closed form is the class of conjugate models. This class of models is defined as pairs of prior and likelihood $(p(\mathrm{d}x), p(y \mid x))$ such that the posterior distribution (2.1) belongs to the same family as the prior distribution. Formally, suppose that the prior distribution $p(\mathrm{d}x)$ is given by a density (or mass function) $p(x; \alpha)$, where $\alpha \in A$ is a collection of parameters, then the family $(p(\cdot; \alpha))_{\alpha \in A}$ is said to be conjugate to $p(y \mid x)$ if the posterior distribution (2.1) belongs to the same parametric family, i.e., if the following definition holds.

Definition 2.1 (Conjugate prior)**.** *A family of distributions with density $p(x; \alpha)$, $\alpha \in A$, is said to be conjugate to a likelihood distribution $p(y \mid x)$ if, for every $y \in \mathcal{Y}$, there exists a vector of hyperparameters $\beta \in A$ such that*

$$\begin{aligned} p(x \mid y) &\propto p(y \mid x) p(x; \alpha), \\ &\propto p(x; \beta), \end{aligned} \tag{2.4}$$

for all $x \in \mathrm{X}$.

In this case, provided that $\beta = \beta(\alpha, y)$ can be computed in closed form, the posterior distribution (2.4) can be computed exactly. We will often informally say that the prior distribution $p(x; \alpha)$ is conjugate to the likelihood distribution $p(y \mid x)$, rather than the family, or even less so, that the prior $p(\mathrm{d}x)$ is, when the context is clear.

A typical example is the Gaussian model, where the prior distribution $p(\mathrm{d}x)$ is given by a Gaussian distribution with mean μ and covariance Σ, and the likelihood distribution $p(y \mid x)$ is given by a Gaussian distribution with mean x and covariance Γ.

Proposition 2.2. *The Gaussian distribution $\mathcal{N}(x;\mu,\Sigma)$ is conjugate to the Gaussian distribution $\mathcal{N}(y;x,\Gamma)$, and the posterior distribution (2.1) is given by*

$$p(x \mid y) = \mathcal{N}\left(x;\mu',\Sigma'\right), \tag{2.5}$$

where

$$\Sigma' = \left(\Sigma^{-1}+\Gamma^{-1}\right)^{-1}, \quad \mu' = \Sigma'\left(\Sigma^{-1}\mu+\Gamma^{-1}y\right). \tag{2.6}$$

In other terms, the Gaussian distribution is conjugate to itself.

Proof. See, for example, Murphy (2022, Section 4.6). □

This result is often useful when expressed in terms of the information form of the Gaussian distribution, which is defined as follows.

Definition 2.3 (Information form)**.** *The information form of a Gaussian distribution $\mathcal{N}(x;\mu,\Sigma)$ is defined as the parameterisation*

$$\mathcal{N}_I(x;\eta,\Lambda), \tag{2.7}$$

where $\eta = \Sigma^{-1}\mu$ and $\Lambda = \Sigma^{-1}$. Λ is then known as the precision or information matrix and η as the information vector.

In this case, the posterior distribution (2.5) can be written as

$$p(x \mid y) = \mathcal{N}_I\left(x;\eta',\Lambda'\right), \tag{2.8}$$

where $\Lambda' = \Lambda+\Gamma^{-1}, \quad \eta' = \eta+\Gamma^{-1}y$.

In general, conjugacy relationships can be obtained within the more general class of the exponential family of distributions (Efron, 2022), which includes the Gaussian distribution as a special case.

Definition 2.4 (Exponential family)**.** *A family of distributions is said to be an exponential family if the density (or mass function) $p(y \mid x)$ can be written as*

$$p(y \mid x) = h(y)\exp\left(\eta(x)^\top T(y) - A(x)\right), \tag{2.9}$$

where $h(y)$ is a non-negative function called the base measure, $\eta(x)$ is a vector of natural parameters, $T(y)$ is a vector of sufficient statistics, and $A(x)$ is the log-normalising constant.

Importantly, all exponential families have conjugate priors (Efron, 2022, Chapter 2), closed under the Bayesian update for the corresponding likelihood.

Example 2.5. *The Gaussian distribution $\mathcal{N}(y; x, \Gamma)$ with known (fixed) covariance is an exponential family with natural parameters $\eta(x) = \Gamma^{-1}x$, sufficient statistics $T(y) = y$, and log-normalising constant $A(x) = \frac{1}{2}x^\top\Gamma^{-1}x + \frac{1}{2}\log\det\Gamma$.*

Remark 2.6. *This self-conjugacy of the Gaussian distribution is crucial in numerical methods developed in the context of this thesis: they are explicitly used in Publications II, III, V, and VII. See also Section 3.2 for their use for inference in dynamic models.*

2.2 Monte Carlo simulation methods

In the previous section, we have seen that exact inference is possible for a small set of models, typically finite models and models for which conjugacy relationships can be established. In general, however, the model may not be finite, or the likelihood may not be conjugate to the prior, and exact inference is then not possible. This is the case for most models of practical interest, and one must then resort to approximation methods. In this section, we review the Monte Carlo method, which originated in the 1940s as a general means of computing integrals by means of simulation or sampling (for a historical account see Roger, 1987).

2.2.1 The classical Monte Carlo method

Central to the method is the law of large numbers (Kallenberg, 1997, Chap. 3), which states that the sample mean of a sequence of independent and identically distributed (i.i.d.) random variables converges to the true mean of the distribution as the number of samples tends to infinity. This is given formally in the following proposition.

Proposition 2.7 (Strong law of large numbers)**.** *Let $X_1, X_2, \ldots$ be a sequence of i.i.d. random variables with mean $\mu = \mathbb{E}[X_1]$. Then, the sample mean $\bar{X}_n = \frac{1}{n}\sum_{i=1}^n X_i$ converges to μ almost surely (a.s.) as $n \to \infty$. Namely,*

$$\mathbb{P}\left(\lim_{n\to\infty} \bar{X}_n = \mu\right) = 1. \tag{2.10}$$

This result is obtained under very mild conditions on the random variables X_i, namely that they are i.i.d. and have finite mean μ. Generalisations are available for even weaker conditions, see for example Kingman (1978) for a version in the non-i.i.d. case.

The law of large numbers (LLN) is the basis of Monte Carlo methods, which can be used to approximate expectations of the form $\mathbb{E}_\pi[f(X)]$ for some π-integrable function $f : \mathsf{X} \to \mathbb{R}$. Assume that we are given a sequence of i.i.d. random variables $X_1, X_2, \dots$ with distribution $\pi(\mathrm{d}x)$, and that we wish to approximate the expectation $\mathbb{E}_\pi[f(X)]$. The LLN states that the sample mean $\bar{f}_n := \frac{1}{n}\sum_{i=1}^n f(X_i)$ converges to $\mathbb{E}_\pi[f(X)]$ a.s. as $n \to \infty$.

Additionally, the speed of convergence of the Monte Carlo approximation $\bar{f}_n$ to the true expectation $\mathbb{E}_p[f(X)]$ can also be obtained via the central limit theorem (CLT, Kallenberg, 1997, Chap. 4).

Proposition 2.8 (Central limit theorem)**.** *Let $X_1, X_2, \dots$ be a sequence of i.i.d. random variables with mean $\mu = \mathbb{E}[X_1]$ and variance $\sigma^2 = \mathbb{V}[X_1] < \infty$. Then, the sample mean $\bar{X}_n = \frac{1}{n}\sum_{i=1}^n X_i$ converges to μ in distribution as $n \to \infty$, i.e., $\bar{X}_n \to \mu$ in distribution as $n \to \infty$. Namely, for every $x \in \mathbb{R}$,*

$$\lim_{n\to\infty} \mathbb{P}\left(\sqrt{n}\frac{\bar{X}_n - \mu}{\sigma} \le x\right) = \Phi(x), \tag{2.11}$$

where $\Phi(x)$ is the cumulative distribution function (CDF) of the standard normal distribution.

In other terms, the CLT states that the approximation $\frac{1}{n}\sum_{i=1}^n f(X_i)$ converges to $\mathbb{E}_\pi[f(X)]$ roughly as fast as $\sqrt{n}$ with a constant factor equal to the standard deviation of the integrand under the law of the X_i. Again, generalisations are available for even weaker conditions, see for example Klass and Teicher (1987) for a version in the non-i.i.d. case. In the remainder of this thesis, we refer to LLNs and CLTs as results taking the form of Proposition 2.7 and Proposition 2.8, respectively. That is, LLNs refer to convergence of a Monte Carlo approximation to a constant, and CLTs refer to the distribution of the asymptotic error of the approximation.

Fundamentally, the LLN and CLT ensure that if we have access to a generator of i.i.d. random variables $X_1, X_2, \dots$ with distribution $\pi(\mathrm{d}x)$, then we can compute expectations of the form $\mathbb{E}_\pi[f(X)]$ by computing the sample mean $\frac{1}{n}\sum_{i=1}^n f(X_i)$ for a sufficiently large number of samples n. In this thesis, and unless stated otherwise, we assume that such a generator is usually available for standard distributions such as the Gaussian distribution or the uniform distribution, and we refer to Gentle (2003) for a comprehensive discussion on how to generate i.i.d. random variables from a given distribution.

2.2.2 Importance sampling

When computing expectations of the form $\mathbb{E}_\pi[f(X)]$ for a given distribution $\pi(\mathrm{d}x)$, a problem that often arises in practice is that the distribution $\pi(\mathrm{d}x)$ and the target function f may not be "compatible", in the sense that the integrand f and the integrator π may place mass on different regions of the space X. In this case, in the Monte Carlo approximation $\frac{1}{n}\sum_{i=1}^n f(X_i)$, most of the samples X_i may be located in regions where the value of f is negligible, and the approximation may therefore be very poor. The following example, consisting in computing expectations under a truncated Gaussian distribution, demonstrates this point.

Example 2.9. *Let $X \sim \mathcal{N}(0,1)$ be a one-dimensional standard Gaussian random variable with mean 0 and variance 1 and let $f_a(x) = \frac{x\mathbb{1}[x>a]}{1-\Phi(a)}$, where $\Phi(x)$ is the cumulative density functions (CDF) of the standard Gaussian distribution. Then, the expectation $\mathbb{E}[f_a(X)]$ is given by $\frac{\varphi(a)}{1-\Phi(a)}$, where $\varphi(x)$ is the probability density functions (PDF) of the standard Gaussian distribution, corresponding to the mean of the truncated Gaussian distribution proportional to $\mathcal{N}(x;0,1)\mathbb{1}[x>a]$. Furthermore, $a \leq \frac{\varphi(a)}{1-\Phi(a)} \leq \frac{a^2+1}{a}$, and therefore $\mathbb{E}[f_a(X)] \approx a$ as $a \to \infty$. On the other hand, if we approximate $\mathbb{E}[f_a(X)]$ by the Monte Carlo approximation $\frac{1}{n}\sum_{i=1}^n f_a(X_i)$, where $X_1,\ldots,X_n$ are i.i.d. samples from $\mathcal{N}(0,1)$, then we have, for a fixed number of samples n,*

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n f_a(X_i) = 0\right) = (1-\Phi(a))^n \to 1 \quad \textit{as} \quad a \to \infty. \tag{2.12}$$

This problem naturally arises in the context of Bayesian inference, where the posterior distribution (2.1) is defined as a product of a prior distribution $p(\mathrm{d}x)$ and a likelihood distribution $p(y \mid x)$, under which the posterior distribution $p(\mathrm{d}x \mid y)$ may end up being concentrated in regions where the prior distribution $p(\mathrm{d}x)$ is not. In order to solve this problem, we can instead write $\mathbb{E}_\pi[f(X)] = \mathbb{E}_q\left[f(X)\frac{\pi(X)}{q(X)}\right]$, where q is another distribution on X such that the ratio $\frac{\pi}{q}$ exists. The resulting Monte Carlo approximation is then given by

$$\frac{1}{n}\sum_{i=1}^n \omega(X_i) f(X_i), \tag{2.13}$$

where $X_1,\ldots,X_n$ are i.i.d. samples from $q(\mathrm{d}x)$ and $\omega(x) := \frac{\pi(x)}{q(x)}$ is the *importance weight* function.

Remark 2.10. *This also holds in the more general case of $\omega(x)$ being the Radon–Nikodym derivative of π with respect to q, i.e., when we have*

$$\pi(A) = \int_A \omega(x)\mathrm{d}q(x) \tag{2.14}$$

for all measurable sets $A \subseteq \mathsf{X}$ such that $q(A) > 0$, in which case, we write $\omega(x) = \frac{\mathrm{d}\pi}{\mathrm{d}q}(x)$. This context arises regularly in the context of Chapter 3.

This method is known as *importance sampling* (for a review, see Elvira and Martino, 2021). It is worth noting that, similar to the classical Monte Carlo procedure, the importance sampling approximation (2.13) is unbiased too, i.e., $\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n f(X_i)\pi(X_i)/q(X_i)\right] = \mathbb{E}_\pi[f(X)]$, where the X_i are samples from $q(\mathrm{d}x)$.

It is easy to see that the mean squared error (MSE) obtained by importance sampling is directly related to the variance of the term $\omega(X_i)f(X_i)$.

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n \omega(X_i)f(X_i) - \mathbb{E}_\pi[f(X)]\right)^2\right] = \frac{1}{n}\mathbb{C}_q\left[\omega(X)f(X)\right], \tag{2.15}$$

where $\mathbb{C}_q[\cdot]$ denotes the variance with respect to the distribution $q(\mathrm{d}x)$.

The following proposition shows that importance sampling *may* reduce the variance of the Monte Carlo approximation provided that the choice of q is appropriate.

Proposition 2.11 (Importance sampling mean squared error). *The error* (2.15) *obtained by importance sampling is minimised when* $q(\mathrm{d}x) = \frac{|f|(x)}{\int |f|(z)\pi(\mathrm{d}z)}\pi(\mathrm{d}x)$.

Proof. See, for example, Särkkä and Svensson (2023, Chap. 11) and Chopin and Papaspiliopoulos (2020, Chap. 8). □

In practice, finding the optimal choice of q is often intractable, and one must resort to heuristic methods, some of which will be discussed in Section 2.3.

Example 2.12. *Coming back to Example 2.9, it is known (see, e.g., Robert, 1995) that truncated Gaussian distributions can be approximated (for example) by a shifted exponential distribution* $q(x; a, \lambda) = \lambda \exp(-\lambda(x-a))\mathbb{1}[x > a]$, *where λ is the rate of the exponential distribution and a is the shift. In this case, the importance weight function is given by* $\omega(x) = \frac{\mathcal{N}(x;0,1)}{\lambda \exp(-\lambda(x-a))}\mathbb{1}[x \geq a]$, *and the Monte Carlo approximation* (2.13) *is given by*

$$\frac{1}{n}\sum_{i=1}^{n} \omega(X_i), \tag{2.16}$$

which, under $X_i \sim q$, will almost surely not be zero. We can then minimise the variance of the Monte Carlo approximation (2.16) *by choosing the rate λ of the exponential distribution such that the variance of the term $\omega(X_i)$ is minimised. We do not provide the details for this here, a similar problem is solved in Robert (1995).*

2.2.3 Self-normalised importance sampling

In the context of Bayesian inference, one is often interested in computing expectations of the form $\mathbb{E}[f(X) \mid y]$ for a given distribution $\pi(\mathrm{d}x) \equiv p(\mathrm{d}x \mid y)$, defined as the posterior distribution (2.1) for a given likelihood $p(y \mid x)$, a given prior $p(\mathrm{d}x)$, and a given observation y. It may, at first sight, seem like we can then simply apply the importance sampling method (2.13) to the posterior distribution $p(\mathrm{d}x \mid y)$, by choosing $q(\mathrm{d}x) \equiv p(\mathrm{d}x)$ to be the prior distribution. However, this forgets that the normalising constant $p(y)$ is unknown, and therefore that the ratio derivative $\frac{\pi}{q}$ is not available in closed form but rather only up to a multiplicative constant, that is, we have access to a function $\omega(x) \propto \frac{\pi}{q}$, which we still refer to as the *importance weight function* by a slight abuse of language.

In this case, the importance sampling method (2.13) can be modified to obtain the so-called self-normalised importance sampling (SNIS) method (see, e.g., Särkkä and Svensson, 2023; Chopin and Papaspiliopoulos, 2020, Chap. 8 and

Chap. 11, resp.), which follows from the identity

$$\begin{aligned}\mathbb{E}_\pi[f(X)] &= \mathbb{E}_q\left[\frac{\pi(X)}{q(X)}f(X)\right],\\ &= \frac{\mathbb{E}_q\left[\omega(X)f(X)\right]}{\mathbb{E}_q\left[\omega(X)\right]},\end{aligned} \tag{2.17}$$

where the multiplicative proportionality constant cancels out.

Contrary to the importance sampling method (2.13), the SNIS method (2.18) is not unbiased, i.e., $\mathbb{E}_q\left[\frac{1}{n}\sum_{i=1}^n w_i f(X_i)\right] \neq \mathbb{E}_\pi[f(X)]$, but is still consistent in the sense that the approximation error converges to 0 as the number of samples n tends to infinity. This corresponds to the following law of large numbers, which holds under integrability conditions on f and ω.

Proposition 2.13 (LLN for self-normalised importance sampling). *The self-normalised importance sampling approximation*

$$\frac{1}{n}\sum_{i=1}^n w_i f(X_i), \tag{2.18}$$

where $X_1,\ldots,X_n$ are i.i.d. samples from $q(\mathrm{d}x)$ and $w_i = \frac{\omega(X_i)}{\sum_{j=1}^n \omega(X_j)}$ are the normalised importance weights, converges to $\mathbb{E}_\pi[f(X)]$ a.s. as $n \to \infty$.

Proof. See, for example, Chopin and Papaspiliopoulos (2020, Chap. 11).

Moreover, a CLT for the SNIS approximation (2.18) can still be obtained.

Proposition 2.14 (CLT for self-normalised importance sampling). *Without loss of generality, assume that $\mathbb{E}_\pi[f(X)] = 0$. Then, under integrability conditions, the self-normalised importance sampling approximation* (2.18) *satisfies*

$$\frac{\sqrt{n}}{n}\sum_{i=1}^n w_i f(X_i) \to \mathcal{N}\left(0, \frac{\mathbb{E}_q\left[\omega(X)^2 f(X)^2\right]}{\mathbb{E}_q\left[\omega(X)\right]^2}\right) \tag{2.19}$$

in distribution as $n \to \infty$. Here, $X_1,\ldots,X_n$ are i.i.d. samples from $q(\mathrm{d}x)$ and $w_i = \frac{\omega(X_i)}{\sum_{j=1}^n \omega(X_j)}$ are the normalised importance weights.

Proof. This result is known as Slutsky's theorem, see for example, Chopin and Papaspiliopoulos (2020, Chap. 11).

Additional convergence results exist for SNIS approximations, for example convergence in the $\mathscr{L}^l$ sense, i.e., showing that $\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n w_i f(X_i) - \mathbb{E}_\pi[f(X)]\right)^l\right] \to 0$ as $n \to \infty$ for $l \geq 1$ (see, e.g., Del Moral, 2004, Chapter 7, for the rate of convergence). Given the form of the asymptotic covariance appearing in (2.19), optimality results for the choice of q for a generic integrand f are not as easily available as in the case of importance sampling (see, however, Agapiou et al., 2017, for worst case bounds), (2.19) and instead, we must resort to heuristic methods. A typical choice consists in minimising the variance of the un-normalised importance weights, given by $\mathbb{E}_q\left[\omega(X)^2\right]$, and is justified by the fact that it minimises the variance of the normalising constant estimator $\frac{1}{N}\sum_{i=1}^n \omega(X_i) \approx \mathbb{E}_q[\omega(X)]$ appearing in the denominator of the SNIS approximation (2.18). This is achieved by choosing $q(\mathrm{d}x) \equiv \pi(\mathrm{d}x)$, which, again, is often intractable in practice.

Remark 2.15. *These convergence results typically extend to contexts treated in this thesis, and we make explicit use of them in Publication I and Publication IV.*

2.2.4 Advanced Monte Carlo methods

The methods presented in this section form the basis of most Monte Carlo methods used in practice. Many more advanced methods have been developed in the literature, such as sequential Monte Carlo (SMC) methods (Del Moral et al., 2006; Chopin and Papaspiliopoulos, 2020), Markov chain Monte Carlo (MCMC) methods (Metropolis et al., 1953; Brooks et al., 2011), which we postpone the discussion of to Chapters 3 and 4, respectively. Additionally, other methods, not discussed in this thesis, have also been developed in the literature.

Three such examples are given by adaptive and annealed importance sampling (Neal, 2001; Bugallo et al., 2017), quasi-Monte Carlo methods (see, e.g., Pagès, 2018, for an introduction) and repulsive point processes (Bardenet and Hardy, 2020), which are based on the idea of replacing the i.i.d. samples $X_1, \dots, X_n$ by dependent ones, covering the space in a more systematic way than Monte Carlo ones, thereby increasing the convergence rate of the Monte Carlo approximation. While these have direct implications in the broader context of some of the methods discussed in this thesis (see, e.g., Gerber and Chopin, 2015), we do not build upon these and therefore do not discuss them further.

2.3 Variational methods

In the previous section, we have seen that Monte Carlo methods can be used to compute asymptotically exact approximations of expectations of the form $\mathbb{E}_\pi[f(X)]$ for a given distribution $\pi(\mathrm{d}x)$, in the sense that the approximation error converges to 0 as the number of samples n tends to infinity. In general, however, the techniques employed assume that a "good enough" sampling procedure is employed, and we have mentioned several times that "heuristic" approximations to the target distribution should be employed. In this section, we review variational methods, which are such a class of methods that approximate the posterior distribution (2.1) by a distribution $q(\mathrm{d}x)$ that is chosen from a tractable family of distributions $\mathscr{Q}$. For a more detailed review of variational methods, see Blei et al. (2017).

2.3.1 The Laplace approximation

Perhaps the first variational[1] method developed in the Bayesian statistical literature is the Laplace approximation (originally developed in Laplace, 1774, which is surprisingly readable if you read French), which aims at approximat-

[1]While it is not *stricto sensu* a variational method, it has the flavour of one as it is based on optimising an objective, and we therefore present it here.

ing the posterior distribution (2.1) by a Gaussian distribution. It follows from the premise that, if the posterior distribution (2.1) is sufficiently concentrated around its mode, then it can be approximated by a Gaussian distribution with mean equal to the mode of the posterior distribution and covariance equal to the negative inverse of the Hessian of the log-posterior evaluated at the mode. Formally, let $x^* = \arg\max_{x \in \mathsf{X}} \log \pi(x)$ be the mode of the posterior distribution (2.1), and let $H(x) = \nabla^2 \log \pi(x)$ be the Hessian of the log-density evaluated at x. Then, the Laplace approximation of the posterior distribution (2.1) is given by

$$q(\mathrm{d}x) = \mathcal{N}\left(x; x^*, -H(x^*)^{-1}\right). \tag{2.20}$$

This approximation is noteworthy in at least two respects. First, it is the first instance of casting intractable inference problems as tractable optimisation problems, which is a common theme in variational inference. Second, in the context of Bayesian inference, it is consistent in the limit of a large number of observations, or, equivalently highly informative observations, in the sense that the true posterior and the Laplace approximation coincide as the number of observations n tends to infinity (Walker, 1969). This is a desirable property, as it means that the Laplace approximation is asymptotically exact in the limit of large data.

On the other hand, its adequacy in the realistic case of finite data is questionable, and centering the (Gaussian) variational distribution around the mode of the posterior distribution may lead to poor approximations of uncertainty, in particular in the case of multimodal posterior distributions. This justifies introducing more flexible approximation methods, which we discuss in the subsequent sections.

2.3.2 Statistical inference as divergence minimisation

The most common approach to variational inference is to minimise a divergence between the posterior distribution (2.1) and the approximating distribution $q(\mathrm{d}x)$. Formally, let $\mathscr{Q}$ be a family of distributions on X, and let $\mathscr{D}(\cdot \parallel \cdot)$ be a divergence between distributions on X, that is a function $\mathscr{D} : \mathscr{P}(\mathsf{X}) \times \mathscr{P}(\mathsf{X}) \to \mathbb{R}_+$ such that $\mathscr{D}(q \parallel \pi) = 0$ if and only if $q \equiv \pi$. Then, the variational inference problem is defined as

$$\min_{q \in \mathscr{Q}} \mathscr{D}(q \parallel \pi). \tag{2.21}$$

The most common choice of divergence is the Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951), which is defined as follows.

Definition 2.16 (Kullback–Leibler divergence)**.** *The Kullback–Leibler divergence between two distributions $q(\mathrm{d}x)$ and $\pi(\mathrm{d}x)$ on X is defined as*

$$\mathscr{D}_{KL}(q \parallel \pi) = \mathbb{E}_q\left[\log \frac{q(X)}{\pi(X)}\right]. \tag{2.22}$$

The KL divergence is a special case of the general class of f-divergences (Rényi, 1961), which are defined as follows.

Definition 2.17 (f-divergence)**.** *Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a convex function such that $f(1) = 0$. The f-divergence between two distributions $q(\mathrm{d}x)$ and $\pi(\mathrm{d}x)$ on X is defined as*

$$\mathscr{D}_f(q \mid \pi) = \int f\left(\frac{q(x)}{\pi(x)}\right)\pi(\mathrm{d}x). \tag{2.23}$$

The KL divergence is then obtained by choosing $f(x) = x\log x$. It provides a measure of the information lost when using q to approximate p, and is therefore a natural choice of divergence for statistical inference. It is worth noting that the KL divergence is not symmetric, i.e., $\mathscr{D}_{\mathrm{KL}}(q \parallel \pi) \neq \mathscr{D}_{\mathrm{KL}}(\pi \parallel q)$, and is therefore not a metric on the space of probability distributions.

Remark 2.18. *An important remark is that, when $\pi(x) \propto h(x)$ is only known up to a normalising constant, minimising the KL divergence (2.22) between $q(\mathrm{d}x)$ and $\pi(\mathrm{d}x)$ is equivalent to minimising*

$$\mathscr{D}_{KL}(q \parallel \pi) = \mathbb{E}_q\left[\log\frac{q(X)}{h(X)}\right], \tag{2.24}$$

where $X \sim q$. This is because the normalising constant $\int h(x)\mathrm{d}x$ does not depend on q and therefore does not affect the minimisation problem (2.21). This property is particularly useful in the context of Bayesian inference, where the normalising constant of p is often intractable.

Another f-divergence is given by the total variation (TV) *distance*, obtained by choosing $f(x) = \frac{1}{2}|x-1|$ and which defines a proper metric on the space of measures.

Definition 2.19 (Total variation distance)**.** *The total variation (TV) distance between two distributions $q(\mathrm{d}x)$ and $\pi(\mathrm{d}x)$ on X is defined as*

$$\begin{aligned}\mathscr{D}_{TV}(q \parallel \pi) &:= \|q - \pi\|_{TV}, \\ &= \sup_{A \in \mathscr{X}} |q(A) - \pi(A)|, \\ &= \sup_f \int f(x)q(\mathrm{d}x) - \int f(x)\pi(\mathrm{d}x),\end{aligned} \tag{2.25}$$

where the second supremum is taken over all measurable functions $f : \mathsf{X} \to [-1, 1]$.

The equivalence between the two representations of the TV distance being given by Donsker and Varadhan's variational principle (Donsker and Varadhan, 1976, Lemma 2.3), also known as Banerjee's compression lemma (Banerjee, 2006).

The total variation distance is often used in the context of statistical inference, as it defines a metric for the weak convergence of bounded functions (see, e.g., Douc et al., 2018, Proposition D.2.6).

Proposition 2.20 (Weak convergence and total variation distance). *Let $q_n(\mathrm{d}x)$ be a sequence of distributions on $(\mathrm{X}, \mathcal{X})$. If $\mathcal{D}_{\mathrm{TV}}(q_n \parallel \pi) \to 0$ as $n \to \infty$, then for any bounded continuous function $f : \mathrm{X} \to \mathbb{R}$, we have*

$$\int f(x) q_n(\mathrm{d}x) \to \int f(x) \pi(\mathrm{d}x) \quad as \quad n \to \infty. \tag{2.26}$$

An important property of the KL divergence is that convergence in KL divergence dominates convergence in the TV sense. This result is a direct consequence of the often-used Pinsker's inequality (Tsybakov, 2009, Chap. 2).

Proposition 2.21 (Pinsker's inequality). *Let $q(\mathrm{d}x)$ and $\pi(\mathrm{d}x)$ be two distributions on $(\mathrm{X}, \mathcal{X})$. Then*

$$\mathcal{D}_{\mathrm{TV}}(q \parallel \pi) \leq \sqrt{\frac{1}{2} \mathcal{D}_{KL}(q \parallel \pi)}. \tag{2.27}$$

Pinsker's inequality also provides a justification for the use of the KL divergence in variational inference, as it quantifies the worst case error incurred by using a variational approximation q instead of the true posterior π, namely, for any $q \in \mathcal{Q}$ and bounded measurable function $f : \mathrm{X} \to \mathbb{R}$, we have

$$\left| \int f(x) q(\mathrm{d}x) - \int f(x) \pi(\mathrm{d}x) \right| \leq \|f\|_\infty \sqrt{\frac{1}{2} \mathcal{D}_{\mathrm{KL}}(q \parallel \pi)}, \tag{2.28}$$

where $\|f\|_\infty = \sup_{x \in \mathrm{X}} |f(x)|$ is the supremum norm of f.

Remark 2.22. *In Publication VII, we use Pinsker's inequality to show an interpolation property between the different stochastic methods we propose, namely, showing that under different regimes, some of the methods are equivalent to others.*

2.3.3 Evidence lower bounds and Monte Carlo objectives

When the target measure $\pi(\mathrm{d}x)$ is the posterior distribution of a Bayesian model with joint distribution $p(\mathrm{d}x, \mathrm{d}y)$, i.e., when $\pi(\mathrm{d}x) \equiv p(\mathrm{d}x \mid y)$ for some y, another interpretation of the KL divergence minimisation is given by the following proposition.

Proposition 2.23 (Evidence lower bound). *Let $p(\mathrm{d}x, \mathrm{d}y)$ be a joint distribution on $(\mathrm{X}, \mathcal{X}) \times \mathcal{Y}$. Then*

$$\mathbb{E}_q \left[\log \frac{\mathrm{d}p(\cdot, y)}{\mathrm{d}q} \right] \tag{2.29}$$

is a lower bound on the marginal likelihood $p(y)$, i.e.,

$$\begin{aligned} \log p(y) &= \log \int p(y \mid x) p(\mathrm{d}x), \\ &\geq \mathbb{E}_q \left[\log \frac{\mathrm{d}p(\cdot, y)}{\mathrm{d}q} \right]. \end{aligned} \tag{2.30}$$

Proof. We have

$$\begin{aligned}
0 \leq \mathscr{D}_{\mathrm{KL}}(q \parallel p(\cdot \mid y)), &= \mathbb{E}_q\left[\log \frac{\mathrm{d}q}{\mathrm{d}p(\cdot \mid y)}\right], \\
&= \mathbb{E}_q\left[\log\left\{p(y)\frac{\mathrm{d}q}{\mathrm{d}p(\cdot, y)}\right\}\right], \qquad (2.31)\\
&= -\mathbb{E}_q\left[\log \frac{\mathrm{d}p(\cdot, y)}{\mathrm{d}q}\right] + \log p(y).
\end{aligned}$$

□

The lower bound (2.30) is known as the evidence lower bound (ELBO) in the Bayesian inference literature, and is often used as an objective function in variational Bayesian inference. It is tight if and only if $q(\mathrm{d}x) \equiv p(\mathrm{d}x \mid y)$, in which case the KL divergence is zero.

The ELBO is particularly useful in the context of parametric models, where the joint distribution $p(\mathrm{d}x, \mathrm{d}y)$ and the variational approximation $q(\mathrm{d}x)$ belong to parametric family of distributions $\mathscr{P} = \{p(\mathrm{d}x, \mathrm{d}y \mid \theta)\}_{\theta\in\Theta}$ and $\mathscr{Q} = \{q(\mathrm{d}x; \phi)\}_{\phi\in\Phi}$, respectively. In this case, the ELBO often provides a tractable lower bound on the marginal likelihood, even when the marginal likelihood itself, given by an integral, is intractable. One can then alternatively maximise the ELBO with respect to θ, and then with respect to ϕ, which is often easier than maximising the marginal likelihood directly.

A typical example is given by the case when the approximate distribution $q(\mathrm{d}x; \phi)$ can easily be sampled from, so that the ELBO can be approximated by Monte Carlo methods:

$$\mathbb{E}_q\left[\log \frac{\mathrm{d}p(\cdot, y)}{\mathrm{d}q}\right] \approx \frac{1}{N}\sum_{n=1}^{N} \log \frac{\mathrm{d}p(x_n, y)}{\mathrm{d}q(x_n; \phi)} \qquad (2.32)$$

where $x_n \sim q(\mathrm{d}x; \phi)$ for $n = 1, \ldots, N$. In this case, in order to maximise the ELBO with respect to ϕ, one can use stochastic gradient ascent (see, e.g., Murphy, 2022, Chap. 8), provided that the gradient of the Monte Carlo approximation (2.32) with respect to ϕ can be computed or approximated. However, computing the gradient of Monte Carlo *samples* with respect to the parameters ϕ used to generate them is not directly possible, and one must therefore resort to one of several methods to estimate the gradient of the ELBO with respect to ϕ. Two common choices are the score function estimator (Kleijnen and Rubinstein, 1996), also known as the REINFORCE trick (Williams, 1992), and the reparameterisation trick (Kingma and Welling, 2014), given in the following two propositions. For the sake of simplicity, we assume that q and p have densities $q(x; \phi)$ and $p(x, y)$ with respect to the Lebesgue measure on their respective value space.

Proposition 2.24 (Score function estimator)**.** *Provided that the gradient of* $\log q(x; \phi)$ *with respect to* ϕ *can be computed, the gradient of the ELBO* (2.29) *with respect to ϕ is given by*

$$\nabla_\phi \mathbb{E}_q \left[\log \frac{p(X,y)}{q(X;\phi)} \right] = \mathbb{E}_q \left[\log \frac{p(X,y)}{q(X;\phi)} \nabla_\phi \log q(X;\phi) \right] - \mathbb{E}_q \left[\nabla_\phi \log q(X;\phi) \right] \quad (2.33)$$

Proof. This follows from the identity

$$q(x;\phi) \nabla_\phi \log q(x;\phi) = \nabla_\phi q(x;\phi) \quad (2.34)$$

and differentiation under the integral sign.

Proposition 2.25 (Reparameterisation trick)**.** *Let $q(\mathrm{d}x;\phi)$ be a distribution on $(\mathsf{X},\mathscr{X})$ and suppose that there exist a probability measure $r(\mathrm{d}z)$ independent of ϕ and a function $f(\cdot,\phi)$ such that $q(\mathrm{d}x;\phi)$ can be written as*

$$q(\cdot;\phi) = f_\#(r,\phi) \quad (2.35)$$

where $f_\#(r,\phi)$ denotes here the push-forward measure of $r(\mathrm{d}z)$ by $f(\cdot,\phi)$, i.e., $f_\#(r,\phi)(A) = r(f^{-1}(A,\phi))$ for all measurable sets $A \subseteq \mathsf{X}$, or equivalently, $Z \sim r(\mathrm{d}z)$ and $X = f(Z,\phi)$ implies that $X \sim f_\#(r,\phi)$. Then provided that f is differentiable with respect to ϕ, the gradient of the ELBO (2.29) with respect to ϕ is given by

$$\begin{aligned} \nabla_\phi \mathbb{E}_q \left[\log \frac{p(X,y)}{q(X;\phi)} \right] &= \nabla_\phi \mathbb{E}_r \left[\log \frac{p(f(Z,\phi),y)}{q(f(Z,\phi);\phi)} \right], \\ &= \mathbb{E}_r \left[\nabla_\phi \log \frac{p(f(Z,\phi),y)}{q(f(Z,\phi);\phi)} \right]. \end{aligned} \quad (2.36)$$

In order to learn the parameters ϕ of the approximate distribution $q(\mathrm{d}x;\phi)$, one can then use stochastic gradient ascent by using Monte Carlo estimates of either (2.33) or (2.36). Score function estimators are (theoretically) easier to obtain than reparameterisation gradients, but are often associated with high variance (Roeder et al., 2017), which can make learning difficult as the performance of stochastic gradient methods is directly related to the variance of the gradient estimates (see, e.g., Lotz, 2018, Theorem 23.1). Reparameterisation gradients, on the other hand, are often associated with lower variance, but are not always available (see Ruiz et al., 2016, however for some generalisation of the method), as shown in the following example.

Example 2.26. *As an illustrative example, consider the discrete distribution $q(\mathrm{d}x;\phi)$ on $\mathrm{X} = \{1,\dots,K\}$, characterised by its vector of probabilities $\phi = (\phi_1,\dots,\phi_K)$, with $q(x;\phi) = \phi_x$ for all $x \in \mathrm{X}$. We wish to compute $\nabla_\phi \mathbb{E}_q[f(X)]$ for some function $f : \mathrm{X} \to \mathbb{R}$. In this case, the score function estimator (2.33) is given by*

$$\nabla_\phi \mathbb{E}_q[f(X)] = \sum_{x\in\mathrm{X}} f(x)\phi_x \nabla_{\phi_x} \log \phi_x. \tag{2.37}$$

A natural way to reparameterise the discrete distribution $q(\mathrm{d}x;\phi)$ consists in writing as a function of a uniform distribution on $[0,1]$ $r(z) = \mathbb{1}_{[0,1]}(z)$, that is as the push-forward measure of $r(\mathrm{d}z)$ under

$$f(z,\phi) = \inf\left\{ 1 \le j \le K \,\middle|\, \sum_{k=1}^{j} \phi_k \le z \right\}. \tag{2.38}$$

Clearly, $f(z,\phi)$ is not differentiable with respect to ϕ at $z = \phi_k$ for $k = 1,\dots,K-1$, and its gradient is 0 everywhere else, so that the reparameterisation gradient (2.36) is not valid.

While Example 2.26 may seem contrived as the gradient can be computed exactly without resorting to Monte Carlo methods, discrete distributions appear as building blocks of more complicated models, such as latent variable models. This has prompted the developments of methods to reduce the variance of the score function estimator thereof, often relying on introducing sophisticated control variates (Glynn and Szechtman, 2002) schemes relying on a leave-one-out approach, whereby J scores estimators are combined, each estimator using the $J-1$ remaining ones to form a baseline (Kool et al., 2019; Dong et al., 2021; Titsias and Shi, 2022; Shi et al., 2022).

Nonetheless, composing reparameterisable distributions is often easier done than computing score estimators of composed distributions. A solution to this specific problem is often to approximate the distribution of interest by a "softened" approximation thereof, trading off bias for differentiability via the reparametrisation trick. In the case of discrete distributions, this is often done by replacing the categorical distribution of interest by a Gumbel-softmax distribution (Maddison et al., 2016; Jang et al., 2016), which is a continuous relaxation of the categorical distribution.

Remark 2.27. *A related approach, relying on an optimal transport relaxation of empirical distributions, is used together with the reparametrisation trick of Proposition 2.25 in Publication I to propagate gradients through distributions involving both discrete and random variables, see also Section 3.4.2 in Chapter 3.*

2.3.4 Generalised statistical linear regression

In the previous section, we have discussed parametric variational inference via numerical optimisation of the KL divergence or, equivalently, the ELBO. A particular focus was put on Monte Carlo methods, which are often used to approximate the ELBO in practice. In this section, we review a class of Gaussian approximations to the posterior distribution (2.1) that are often used in practical scenarios when the posterior distribution of interest is unimodal and a Gaussian approximation thereof is sufficient for the task at hand.

In this context, one is given a Gaussian prior distribution $p(\mathrm{d}x) \sim \mathcal{N}(x; \mu, \Sigma)$ and a likelihood distribution $p(\mathrm{d}y \mid x)$. Given that Gaussian family is closed under conditioning, a natural way to derive a Gaussian approximation to the posterior distribution (2.1) is to first derive one to the likelihood distribution $p(\mathrm{d}y \mid x)$, and then to condition it on the observation y. Namely, we wish to introduce an *enabling (variational) approximation* (Särkkä and Svensson, 2023, Chap. 9) $q(y \mid x) \sim \mathcal{N}(y; Ax + b, C)$ to the likelihood distribution $p(\mathrm{d}y \mid x)$ and then to condition it on the observation y to obtain a Gaussian approximation q to the posterior distribution (2.1).

Prior statistical linear regression. To do so, we can consider the MSE $\ell(A, b) = \mathbb{E}\left[\|Y - AX - b\|_2^2\right]$ corresponding to using the predictor $\mathbb{E}[Y \mid x] \approx Ax + b$ under the expectation of the prior distribution for X, and the resulting residual noise covariance matrix $\mathbb{C}[Y - AX - b] \approx C$. This transforms the problem of finding a Gaussian approximation to the likelihood distribution $p(\mathrm{d}y \mid x)$ into the problem of finding the optimal parameters A, b minimising the MSE $\ell(A, b)$ and *then* the covariance matrix C of the residual noise. This method is justified in the context of variational inference by the following proposition (Garcìa-Fernàndez et al., 2015).

Proposition 2.28 (Variational inference as MSE minimisation)**.** *Suppose that $p(y \mid x) \sim \mathcal{N}(y; m^Y(x), R)$ is conditionally Gaussian, so that $\mathbb{E}[Y \mid x] = m^Y(x)$ and $\mathbb{C}[Y \mid x] = R$. Minimising the MSE $\ell(A, b)$ and then finding C is equivalent to minimising the expected (with respect to $p(x)$) KL divergence between $q(\mathrm{d}y \mid x) \sim \mathcal{N}(y; Ax + b, C)$ and $p(\mathrm{d}y \mid x)$, i.e., finding*

$$A, b, C = \arg\min \mathbb{E}_{p(x)}[\mathcal{D}_{KL}(q(\cdot \mid x) \parallel p(\cdot \mid x))] \tag{2.39}$$

where the minimisation is taken over the parameters of q.

Thankfully, the solution to the MSE problem is known to be given by the following proposition (see, e.g., Tronarp et al., 2018, Theorem 1)

Proposition 2.29 (Statistical linear regression)**.** *The optimal parameters A, b*

and C minimising $L(A,b,C) = \mathbb{E}[\ell(X;A,b,C)]$ for $X \sim \mathcal{N}(x;\mu,\Sigma)$ are given by

$$\begin{aligned} A &= \mathbb{C}[Y,X]\Sigma^{-1}, \\ b &= \mathbb{E}[Y] - A\mathbb{E}[X], \\ C &= \mathbb{C}[Y] - A\Sigma A^T. \end{aligned} \tag{2.40}$$

Suppose now that the conditional first two moments of $p(y \mid x)$ can be computed (or approximated) in closed form:

$$\begin{aligned} \mathbb{E}[Y \mid x] &=: m^Y(x), \\ \mathbb{C}[Y \mid x] &=: C^Y(x), \end{aligned} \tag{2.41}$$

where $\mathbb{C}[Y \mid x] := \mathbb{E}[(Y - \mathbb{E}[Y \mid x])(Y - \mathbb{E}[Y \mid x])^T]$ denotes the covariance matrix of the random variable Y given the realisation x of X. Then, using the laws of total expectation and total variance, we can compute A, b and C using that

$$\begin{aligned} \mathbb{C}[Y,X] &= \mathbb{C}\left[m^Y(X), X\right], \\ \mathbb{E}[Y] &= \mathbb{E}\left[m^Y(X)\right], \\ \mathbb{C}[Y] &= \mathbb{E}\left[C^Y(X)\right] + \mathbb{C}\left[m^Y(X)\right]. \end{aligned} \tag{2.42}$$

In other terms, the problem of finding a Gaussian approximation to the posterior distribution (2.1) reduces to that of finding the expectation and covariance matrices of *deterministic* functions of X. This problem has a long-standing history in the signal processing literature and is the basis of linearisation techniques used intensively in the context of Gaussian-approximated non-linear filtering (Garcìa-Fernàndez et al., 2015; García-Fernández et al., 2016). We note that many methods have been developed in the literature to compute the parameters A, b, and C (Tronarp et al., 2018), such as Taylor linearisation of (2.42), unscented transforms (Julier and Uhlmann, 2004), or other Gaussian quadrature rules, which we do not discuss further in this thesis.

Once the parameters A, b, and C have been computed, we can then condition the approximate posterior distribution $q(\mathrm{d}x \mid y) \propto p(\mathrm{d}x)q(y \mid x)$ on the observation y to obtain a Gaussian approximation to the posterior distribution (2.1):

$$q(\mathrm{d}x \mid y) \sim \mathcal{N}(x;\mu',\Sigma'), \tag{2.43}$$

where

$$\begin{aligned} \Sigma' &= \left(\Sigma^{-1} + A^T C^{-1} A\right)^{-1}, \\ \mu' &= \Sigma'\left(\Sigma^{-1}\mu + A^T C^{-1} y - A^T C^{-1} b\right). \end{aligned} \tag{2.44}$$

The resulting variational approximation $q(\mathrm{d}x \mid y)$ can then be used as is or, for instance, as part of a Monte Carlo approximation, such as in the SNIS approximation (2.18) of posterior expectations of the form $\mathbb{E}_p[f(X)]$.

Generalised statistical linear regression. In the previous section, we have tackled the problem of finding a Gaussian approximation to the posterior distribution (2.1) by first minimising the MSE $\ell(A,b)$ and then finding the residual noise covariance matrix C under the expectation of the prior distribution $\mathcal{N}(\mu,\Sigma)$ for X. This method, which we referred to as *prior statistical linear regression*, is justified in the context of variational inference by Proposition 2.28, relating it to a minimisation of the expected KL divergence between $q(\mathrm{d}y \mid x) \sim \mathcal{N}(y; Ax+b, C)$ and $p(\mathrm{d}y \mid x)$, where the expectation is taken under the prior $p(x)$. However, this method is not always appropriate, in particular when the prior distribution $\mathcal{N}(\mu,\Sigma)$ is not a good approximation to the posterior distribution $p(\mathrm{d}x \mid y)$ in the first place. Indeed, in this case, minimising the KL divergence for values that are unlikely to be realised when observing y may lead to an enabling approximation $q(\mathrm{d}y \mid x)$ that does not capture the true posterior distribution $p(\mathrm{d}x \mid y)$ well. Instead, we would rather minimise the KL divergence with respect to the expected *true* unknown posterior distribution $p(\mathrm{d}x \mid y)$.

A natural way to achieve this is by ensuring that the variational approximation $q(\mathrm{d}x \mid y)$ is unchanged under the statistical linearisation procedure, i.e., by requiring that, under $X \sim \mathcal{N}(m,P)$ corresponding to $q(\mathrm{d}x \mid y)$, we have

$$m = \mu', \quad P = \Sigma', \tag{2.45}$$

where μ' and Σ' are given by (2.44). In other terms, $q(\mathrm{d}x \mid y) \sim \mathcal{N}(m,P)$ should be a fixed point of the statistical linearisation procedure. This is summarised in Algorithm 1.

Algorithm 1: Iterated statistical linearisation

```
input :Prior distribution $p(\mathrm{d}x) \sim \mathcal{N}(\mu,\Sigma)$, conditional mean and
       covariance functions $m^Y(x)$, $C^Y(x)$, observation $y$.
       Optional: initialisation $\mu', \Sigma'$, if not provided, use $\mu' = \mu$, $\Sigma' = \Sigma$.
output:Approximate posterior distribution $q(\mathrm{d}x \mid y) \sim \mathcal{N}(x; \mu', \Sigma')$
1 while not converged do
2     Compute $A$, $b$, and $C$ using (2.42) for $X \sim \mathcal{N}(\mu',\Sigma')$
3     Compute $\mu'$ and $\Sigma'$ using (2.44)
4 return $q(\mathrm{d}x \mid y) \sim \mathcal{N}(x; \mu', \Sigma')$
```

In Publication V, we use this technique to generalise the work of Titsias and Papaspiliopoulos (2018a), which considers the problem of sampling from distributions with Gaussian priors to non-Gaussian priors.

2.3.5 Gradient flows

In the previous sections, we have discussed variational inference as a minimisation problem typically over the KL divergence between the posterior distribution (2.1) and an approximating distribution $q(\mathrm{d}x)$. Let us abstract this problem for a moment and consider the more general problem of minimising a loss function $\mathscr{L} : \Phi \to \mathbb{R}_+$, over a Euclidean space Φ

$$\min_{\phi \in \Phi} \mathscr{L}(\phi). \tag{2.46}$$

We define the (Euclidean) proximal operator of $\mathscr{L}$ as

$$\mathrm{prox}_{\gamma \mathscr{L}}(\phi) = \arg\min_{\phi' \in \Phi} \left\{ \mathscr{L}(\phi') + \frac{1}{2\gamma} \left\| \phi' - \phi \right\|^2 \right\}, \tag{2.47}$$

then, under some conditions on the loss function, for $\gamma > 0$, the following iterative procedure converges to a local minimum of $\mathscr{L}$ (Karimi et al., 2016):

$$\phi_{k+1} = \mathrm{prox}_{\gamma \mathscr{L}}(\phi_k). \tag{2.48}$$

When the step-size γ is taken to 0, it can be shown to converge to a *gradient flow*, which is a continuous-time dynamical system defined by the following ordinary differential equation (ODE):

$$\begin{aligned} \dot{\phi} &= \nabla_\phi \mathscr{L}(\phi_t), \\ \phi(0) &= \phi_0. \end{aligned} \tag{2.49}$$

This gradient flow is also called steepest descent flow, as it takes the direction of steepest descent of the loss function $\mathscr{L}$ at each time t. Furthermore, and assuming that $\mathscr{L}$ is λ-strongly convex, i.e., that there exists $\lambda > 0$ such that $\phi \mapsto \mathscr{L}(\phi) - \frac{\lambda}{2} \left\| \phi \right\|^2$ is convex (noting that this implies that $\mathscr{L}$ is convex), it can be shown by an application of Polyak–Łojasiewicz inequality (see, e.g., Karimi et al., 2016) and Grönwall's inequality (Pachpatte, 1998, Chap. 1) that the gradient flow (2.49) converges to a local minimum of $\mathscr{L}$ at an exponential rate:

$$\mathscr{L}(\phi_t) - \mathscr{L}(\phi^*) \le \exp(-2\lambda t) \left(\mathscr{L}(\phi_0) - \mathscr{L}(\phi^*) \right), \tag{2.50}$$

where ϕ^* is a local minimum of $\mathscr{L}$. However, its behaviour is tied to the choice of the parameterisation of the family of approximating distributions $\mathscr{Q}$, which often will not correspond to a convex loss function $\mathscr{L}$. As a consequence, it is more natural to consider dynamics that are agnostic to the parameterisation of the approximating distribution $q_\phi(\mathrm{d}x)$, and instead consider the dynamics of the approximating distribution itself.

In order to do so, we can consider instead the proximal operator over the space of probability distributions $\mathscr{P}$ endowed with the Wasserstein metric $\mathscr{W}_2$, defined as follows.

Definition 2.30 (Wasserstein distance)**.** *Let $\mathscr{P}$ be the space of probability distributions on $(\mathrm{X}, \mathscr{X})$. The Wasserstein distance between two distributions $p(\mathrm{d}x)$ and $q(\mathrm{d}x)$ on $(\mathrm{X}, \mathscr{X})$ is defined as*

$$\mathscr{W}_2(p, q) = \inf_{\pi \in \Pi(p,q)} \left\{ \int \|x - y\|^2 \pi(\mathrm{d}x, \mathrm{d}y) \right\}^{1/2}, \tag{2.51}$$

where $\Pi(p,q)$ *denotes the set of all joint distributions* $\pi(\mathrm{d}x, \mathrm{d}y)$ *on* $(\mathrm{X}, \mathscr{X}) \times (\mathrm{X}, \mathscr{X})$ *with marginals* $p(\mathrm{d}x)$ *and* $q(\mathrm{d}y)$*, also known as couplings of* p *and* q*.*

Then, given a loss function $\mathscr{L} : \mathscr{P} \to \mathbb{R}_+$, we define the proximal operator of $\mathscr{L}$ as

$$\operatorname{prox}_{\gamma \mathscr{L}}(q) = \arg\min_{q' \in \mathscr{P}} \left\{ \mathscr{L}(q') + \frac{1}{2\gamma} \mathscr{W}_2^2(q', q) \right\}. \tag{2.52}$$

When taking the limit $\gamma \to 0$, the following iterative procedure recovers a partial differential equation (PDE) known as the Wasserstein gradient flow (see, e.g., Villani, 2009; Santambrogio, 2017, Chap. 15, Section 4, respectively):

$$\begin{aligned} \partial_t q(t,x) &= \nabla_x \cdot \left(q(t,x) \nabla_x \frac{\delta \mathscr{L}}{\delta q}(q(t,x)) \right), \\ q(0,x) &= q_0(x), \end{aligned} \tag{2.53}$$

where $\frac{\delta \mathscr{L}}{\delta q}$ denotes the functional derivative of $\mathscr{L}$ with respect to q. Thankfully (see, e.g., Villani, 2009, Exercise 15.10), when the loss is given by the KL divergence, $\mathscr{L}(q) = \int q(x) \log \frac{q(x)}{\pi(x)} \mathrm{d}x$, we have[2]

$$\frac{\delta \mathscr{D}_{\mathrm{KL}}(q \parallel \pi)}{\delta q}(q)(x) = \log \frac{q(x)}{\pi(x)} + 1, \tag{2.54}$$

and we have

$$\begin{aligned} \partial_t q(t,x) &= \nabla_x \cdot \left(q(t,x) \nabla_x \left(\log \frac{q(t,x)}{\pi(x)} \right) \right), \\ q(0,x) &= q_0(x). \end{aligned} \tag{2.55}$$

The Wasserstein gradient flow (2.55) can be interpreted as a continuous-time version of the proximal gradient descent (2.48) in the space of probability distributions $\mathscr{P}$ endowed with the Wasserstein metric $\mathscr{W}_2$. Moreover, as soon as $\pi(x) \propto \exp(-V(x))$ for some potential function $V : (\mathrm{X}, \mathscr{X}) \to \mathbb{R}$ which is strongly convex, the $\mathscr{D}_{\mathrm{KL}}(q \parallel \pi)$ divergence will be too (see, e.g., Villani, 2009, Chap. 17), and the Wasserstein gradient flow (2.55) will therefore enjoy fast convergence properties. Such property is independent of the parameterisation of the approximating distribution q, which one can then retrieve by projecting the solution of the Wasserstein gradient flow (2.55) onto a chosen subspace of distributions $\mathscr{Q}$. This is the basis of Lambert et al. (2022) who consider projecting the solution onto the space of Gaussian distributions: $q_t(x) = \mathcal{N}(x; \mu_t, \Sigma_t)$, obtaining the following pair of interacting ODEs:

$$\begin{aligned} \frac{\mathrm{d}\mu_t}{\mathrm{d}t} &= -\mathbb{E}[\nabla V(Z_t)], \\ \frac{\mathrm{d}\Sigma_t}{\mathrm{d}t} &= 2I - \mathbb{E}\left[\nabla V(Z_t)(Z_t - \mu_t)^\top\right] - \mathbb{E}\left[(Z_t - \mu_t)\nabla V(Z_t)^\top\right], \end{aligned} \tag{2.56}$$

[2] Remember that $\frac{\delta \mathscr{D}_{\mathrm{KL}}(q \parallel \pi)}{\delta q}$ maps a distribution to a function, so that $\frac{\delta \mathscr{D}_{\mathrm{KL}}(q \parallel \pi)}{\delta q}(q)$ is a function.

where $Z_t \sim \mathcal{N}(\mu_t, \Sigma_t)$, introduced in the literature for treating the problem of Gaussian-assumed filtering (Särkkä, 2007), see also Section 3.2.2 in Chapter 3. Computing the best Gaussian approximation to the posterior distribution (2.1) then amounts to solving the ODEs (2.56), which can be done using standard numerical methods, such as the Euler method.

Remark 2.31. *This technique is the basis of the approximating methods developed in Publication VI, where we consider iteratively applying the Wasserstein gradient flow* (2.55) *to the problem of approximating the posterior distribution of a state-space model (see Chapter 3).*

3. Exact and approximate inference in Markovian models

In the previous chapter, we have introduced the problem of Bayesian inference in intractable general statistical models. In this chapter we introduce *Markovian* models, which are a recurring theme of this thesis, both as a computational target as well as a computational tool. We start by introducing the basic concepts and notations of Markovian models in Section 3.1, after which we focus on the case of linear Gaussian state-space models (LGSSMs) in Section 3.2. Some variational approximations to the filtering and smoothing distributions are then discussed in Section 3.3, before we turn to general Monte Carlo methods, i.e., particle filtering and smoothing solutions in Section 3.4. The former is an instance of doing exact inference in approximate models, while the latter is an instance of doing approximate inference in exact models.

In this chapter, we make use of the notations and conventions introduced in the preamble of this thesis. In particular, we identify measures with their densities when they exist, and we consequently use the measure-theoretic notation $\pi(\mathrm{d}x)$ or $\mathrm{d}\pi(x)$ to denote the measure with density $\pi(x)$ when integrating over it or when considered as part of a sampling procedure.

3.1 Markovian models

Markovian models are a class of statistical models that are defined by a Markovian structure, i.e., a structure that represents sequential independence assumptions between random variables. They are ubiquitous in statistics and machine learning, where they are used to model time series, dynamical systems, and more generally any system that evolves in physical or abstract time (Doucet and Johansen, 2011; Chopin and Papaspiliopoulos, 2020; Särkkä and Svensson, 2023). In this section, we introduce the basic concepts and notations of Markovian models, and provide a few examples.

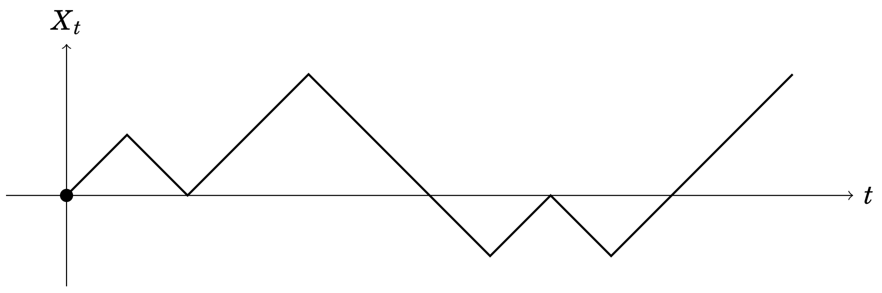


Figure 3.1. An example realisation from a random walk with $T = 6$ steps.

3.1.1 Basic concepts and notations

Markovian models, which are the main focus of this chapter are defined as follows.

Definition 3.1. *Let $T \in \mathbb{N}_0$, we define a Markovian model on the space X as a sequence of measures π_t on X^{t+1}, $t \in \{0, \dots, T\}$, such that for all $t \in \{0, \dots, T-1\}$ and a.e. $x_{0:t} \in \mathrm{X}^{t+1}$,*

$$\pi_{t+1}(\mathrm{d}x_{0:t+1}) \propto \pi_t(\mathrm{d}x_{0:t}) \Gamma_{t+1}(x_{s+1}, x_s), \tag{3.1}$$

where, $\Gamma_{t+1}(x_{t+1}, x_t) \geq 0$ is a weighting kernel.

In other terms, a model is Markovian if bridging from the previous distribution $\pi_t(\mathrm{d}x_{0:t})$ to the next $\pi_{t+1}(\mathrm{d}x_{0:t+1})$ is done by multiplying the previous distribution by a weighting function $\Gamma_{t+1}(x_{t+1}, x_t)$. Note that, in the above, we have assumed that the measures π_t had a density with respect to a product measure. In Section 3.1.2, we will introduce the Feynman–Kac formalism, which allows us to represent Markovian models in a more algorithmically convenient way, but also in a way that reduces the need for densities.

Markovian models appear in many contexts, and we will encounter them in the rest of this thesis. We now provide two illustrative examples of Markovian models.

Example 3.2. *Consider the space $\mathrm{X} = \mathbb{Z}$, the probability measure $\pi_0^{\mathrm{rw}}(\mathrm{d}x_0) = \delta_0(\mathrm{d}x_0)$ and the conditional probability kernel*

$$\Gamma_{t+1}^{\mathrm{rw}}(x_{t+1}, x_t) = \left(\delta_{x_t - 1}(\mathrm{d}x_{t+1}) + \delta_{x_t + 1}(\mathrm{d}x_{t+1})\right).$$

In other terms, we start at 0 and at each step we either increase or decrease the current position by one with equal probability. Then the sequence of measures π_t^{rw} defined by (3.1) is a Markovian model on X and the corresponding Markov chain is a random walk. We illustrate this model in Figure 3.1.

It is possible to see by direct calculation that, for the random walk of Example 3.2, if $X_t \sim \pi_t^{\mathrm{rw}}$, then $\mathbb{P}(X_{2t} = k) = 0$ if k is odd, and $\mathbb{P}(X_{2t} = k) = \binom{2t}{k+t} 2^{-2t}$ if k is even, and similarly for X_{2t+1}. In other terms, the marginal distribution of

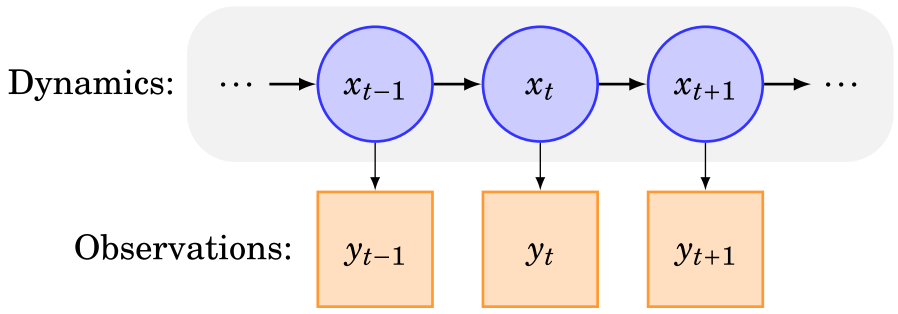


Figure 3.2. Graphical representation of a state-space model.

X_t can be computed explicitly. This is not the case in general, and, often, Γ_t is not directly tractable, making computation or simulation in Markovian models challenging. To illustrate this, we now turn to a more complex example, where X_t corresponds to the state of a latent physical system, which is not directly observable, we only have access to noisy observations Y_t of this system. This is a common setting in many applications, and is known as a state-space model (SSM).

Example 3.3. *Consider a Markov chain $(X_t)_{t=0}^T$ on X, with initial distribution $p_0(\mathrm{d}x_0)$ and transition kernel $p_{t+1}(\mathrm{d}x_{t+1} \mid x_t)$. We assume that we have access to noisy observations of this Markov chain, i.e., we observe realisations of $Y_t \sim p_t(y_t \mid X_t)$ for $t \in \{0, \dots, T\}$. Then the sequence of measures π_t defined by*

$$\begin{aligned}\pi_t(\mathrm{d}x_{0:t}) &= p(\mathrm{d}x_{0:t} \mid y_{0:t}) \\ &\propto p_0(\mathrm{d}x_0) \prod_{s=0}^{t-1} p_{s+1}(\mathrm{d}x_{s+1} \mid x_s) \prod_{s=0}^{t} p_s(y_s \mid x_s)\end{aligned} \tag{3.2}$$

is called a state-space model (SSM) on (X, Y). We illustrate this in Figure 3.2. In this case, and when all the distributions appearing in the definition of π_t have a density, we have

$$\Gamma_t(x_{t+1}, x_t) = p_{t+1}(x_{t+1} \mid x_t) p_{t+1}(y_{t+1} \mid x_{t+1})$$

with additionally

$$\pi_0(x_0) = \frac{p_0(x_0) p_0(y_0 \mid x_0)}{\int p_0(\mathrm{d}z_0) p_0(y_0 \mid z_0)},$$

which are often known up to a normalising constant only, making inference intractable in most cases, and we indeed have

$$\pi_t(x_{0:t}) \propto \Gamma_t(x_t, x_{t-1}) \pi_{t-1}(x_{0:t-1})$$

for all $t \in \{1, \dots, T\}$.

The structure of Example 3.3 often holds more generally, whereby the weighting function $\Gamma_{t+1}(x_{t+1}, x_t)$ is often expressed as the product of a transition model X_{t+1} given the previous state X_t and a likelihood term (see Section 3.1.2 for a formal definition) depending on both terms too, and is therefore rarely amenable to direct computation, making the computation of the sequence of measures π_t in Definition 3.1 challenging.

We can now formulate the three main inference problems in Markovian models.

Definition 3.4 (The filtering problem). *Let $(\pi_t)_{t=0}^T$ be a Markovian model on $\mathbb{X}$. The filtering problem consists in computing the marginal filtering distribution of the current state X_t in the current model, i.e., computing $\pi_t(\mathrm{d}x_t)$, the marginal distribution of X_t in π_t, or expectations of the form $\mathbb{E}_{\pi_t}[\varphi(X_t)]$ for some admissible function φ.*

Definition 3.5 (The (marginal) smoothing problem). *Let $(\pi_t)_{t=0}^T$ be a Markovian model on $\mathbb{X}$. The smoothing problem consists in computing the marginal smoothing distribution of the past state X_t in the final model, i.e., computing $\pi_T(\mathrm{d}x_t)$, the marginal distribution of X_t in π_T, or expectations of the form $\mathbb{E}_{\pi_T}[\varphi(X_t)]$ for some admissible function φ.*

Definition 3.6 (The pathwise smoothing problem). *Let $(\pi_t)_{t=0}^T$ be a Markovian model on $\mathbb{X}$. The pathwise smoothing problem consists in computing the joint smoothing distribution of the states $X_{0:T}$ in the final model, i.e., computing $\pi_T(\mathrm{d}x_{0:T})$ or expectations of the form $\mathbb{E}_{\pi_T}[\varphi(X_{0:T})]$ for some admissible function φ.*

It is worth noting that the filtering problem is a special case of the marginal smoothing problem, itself a special case of the pathwise smoothing problem; in other terms, it suffices to solve the pathwise smoothing problem to solve both of the other problems. This however would disregard the structure of the problem: for instance, the filtering distribution $\pi_t(\mathrm{d}x_t)$ is essentially an *online* quantity, that can be computed sequentially, while the smoothing distribution $\pi_T(\mathrm{d}x_t)$ is an *offline* quantity and, as we will see in the remainder of this chapter, can often be computed as a byproduct of the filtering distribution.

In addition to the above, it often happens that the Markovian model of interest depends on an unknown parameter θ, i.e., we have $\pi_t(\cdot \mid \theta)$, and the parameter may or may not be equipped with a prior distribution $p(\mathrm{d}\theta)$. In this case, we are also interested in performing inference on the parameter θ. This can correspond to a variety of problems, depending on the model and the application of interest, but typically corresponds to one of the following two problems.

Definition 3.7 (The pointwise parameter estimation problem). *Let $(\pi_t(\cdot \mid \theta))_{t=0}^T$ be a parametric Markovian model on $\mathbb{X}$. Given a loss function $\mathscr{L}(x_{0:T}, \theta)$, the pointwise parameter estimation problem consists in minimising the expected loss*

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{\pi_T(\cdot \mid \theta)}[\mathscr{L}(X_{0:T}, \theta)].$$

Example 3.8. *As explained in Chapter 2, this can take the more familiar form of the MLE*

$$\theta^* = \arg\max_{\theta\in\Theta} p(y_{0:T} \mid \theta)$$

and the MAP

$$\theta^* = \arg\max_{\theta\in\Theta} p(y_{0:T} \mid \theta)p(\theta),$$

where we have used the SSM formalism for clarity.

Definition 3.9 (The Bayesian parameter estimation problem). *Let* $(\pi_t(\cdot \mid \theta))_{t=0}^T$ *be a parametric Markovian model on* X *and* $p(\mathrm{d}\theta)$ *be a prior distribution on* Θ*. The Bayesian parameter estimation problem consists in computing the posterior distribution of the parameter* θ*, i.e., computing* $\pi_T(\mathrm{d}\theta)$*, the marginal distribution of* θ *under* $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T}) = \pi_T(\mathrm{d}x_{0:T} \mid \theta)p(\mathrm{d}\theta)$*, or expectations of the form* $\mathbb{E}_{\pi_T(\mathrm{d}\theta)}[\varphi(\theta)]$ *for some admissible function* φ*.*

Example 3.10. *In the case when* $\pi_T(\cdot \mid \theta)$ *is given by a state-space model as in Example 3.3, the Bayesian parameter estimation problem of Definition 3.9 corresponds to computing the posterior distribution of the parameter* θ *given the observations* $y_{0:T}$*, i.e., computing* $p(\mathrm{d}\theta \mid y_{0:T}) =: \pi_T(\mathrm{d}\theta)$*.*

3.1.2 The Feynman–Kac formalism

Markovian models can be described in an algorithmically convenient way using the Feynman–Kac formalism (Del Moral, 2004; Chopin and Papaspiliopoulos, 2020). This formalism is based on representing the Markovian incremental weights $\Gamma_{t+1}(x_{t+1}, x_t)$ as the product of two terms: a *transition kernel* $M_{t+1}(\mathrm{d}x_{t+1} \mid x_t)$ and a *potential function* $G_{t+1}(x_{t+1}, x_t)$. In other terms, we have

$$\pi_{t+1}(\mathrm{d}x_{0:t+1}) \propto G_{t+1}(x_{t+1}, x_t)M_{t+1}(\mathrm{d}x_{t+1} \mid x_t\pi_t(\mathrm{d}x_{0:t}), \tag{3.3}$$

defining $\Gamma_{t+1}(x_{t+1}, x_t) = G_{t+1}(x_{t+1}, x_t)M_{t+1}(x_{t+1} \mid x_t)$ up to a normalising constant when the measures have densities. This representation is not unique, and given another choice $M'_{t+1}(\mathrm{d}x_{t+1} \mid x_t)$, and when $M_{t+1}(\mathrm{d}x_{t+1} \mid x_t)$ has a density with respect to the new kernel $M'_{t+1}(\mathrm{d}x_{t+1} \mid x_t)$, i.e., when there exists a non-negative measurable function (the Radon–Nykodim derivative) ν verifying

$$M_{t+1}(\mathrm{d}x_{t+1} \mid x_t) = \underbrace{\nu(x_{t+1}, x_t)}_{=:\frac{M_{t+1}(\mathrm{d}x_{t+1} \mid x_t)}{M'_{t+1}(\mathrm{d}x_{t+1} \mid x_t)}} M'_{t+1}(\mathrm{d}x_{t+1} \mid x_t), \tag{3.4}$$

then we can define another potential function $G'_{t+1}(x_{t+1}, x_t)$ such that

$$G'_{t+1}(x_{t+1}, x_t) = \frac{M_{t+1}(\mathrm{d}x_{t+1} \mid x_t)}{M'_{t+1}(\mathrm{d}x_{t+1} \mid x_t)} G_{t+1}(x_{t+1}, x_t). \tag{3.5}$$

Remark 3.11. *It is worth noting that when both $M_{t+1}(\mathrm{d}x_{t+1} \mid x_t)$ and $M'_{t+1}(\mathrm{d}x_{t+1} \mid x_t)$ have densities with respect to a common measure, the potential function $G'_{t+1}(x_{t+1}, x_t)$ defined in (3.5) can be simply expressed in terms of the ratio of the densities of M_{t+1} and M'_{t+1}:*

$$G'_{t+1}(x_{t+1}, x_t) = \frac{M_{t+1}(x_{t+1} \mid x_t)}{M'_{t+1}(x_{t+1} \mid x_t)} G_{t+1}(x_{t+1}, x_t), \tag{3.6}$$

where we have identified the measures with their densities.

Similarly, the initial distribution $\pi_0(\mathrm{d}x_0)$ can be represented as the product of an *initial distribution* $M_0(\mathrm{d}x_0)$ and an initial *potential function* $G_0(x_0)$, i.e.,

$$\pi_0(\mathrm{d}x_0) \propto M_0(\mathrm{d}x_0) G_0(x_0), \tag{3.7}$$

which, too, is not unique, and can be changed as in (3.4).

This can be summarised by the following model, which we call a *Feynman–Kac representation* of the Markovian model π_t:

$$\pi_t(\mathrm{d}x_{0:t}) \propto M_0(\mathrm{d}x_0) G_0(x_0) \prod_{s=0}^{t-1} M_{s+1}(\mathrm{d}x_{s+1} \mid x_s) \prod_{s=1}^{t-1} G_s(x_s, x_{s-1}). \tag{3.8}$$

Example 3.12. *Consider the state-space model of Example 3.3. A Feynman–Kac representation of this model is given by*

$$M_t(\mathrm{d}x_t \mid x_{t-1}) = p_t(\mathrm{d}x_t \mid x_{t-1}) \quad \textit{and} \quad G_t(x_t, x_{t-1}) = p_t(y_t \mid x_t).$$

This is called the bootstrap *Feynman–Kac representation of the state-space model (see, e.g., Chopin and Papaspiliopoulos, 2020, Chap. 10). Another representation is given by*

$$M_t(\mathrm{d}x_t \mid x_{t-1}) = p_t(\mathrm{d}x_t \mid x_{t-1}, y_t) \quad \textit{and} \quad G_t(x_t, x_{t-1}) = \frac{p_t(y_t \mid x_t) p_t(x_t \mid x_{t-1})}{p_t(x_t \mid x_{t-1}, y_t)},$$

which is known as the locally optimal *Feynman–Kac representation of the state-space model (see, e.g., Doucet and Johansen, 2011) as it is optimal in the sense that it minimises the variance of the weights as in Section 2.2.3. Note how, in the bootstrap representation, the potential function $G_t(x_t, x_{t-1})$ is the likelihood of the observation y_t given the state x_t, while in the locally optimal representation, it may depend on both the state x_t and the previous state x_{t-1}.*

Whilst the usefulness of the Feynman–Kac formalism may not be immediately obvious, it will become apparent in Section 3.4 when we introduce sequential Monte Carlo methods. Another advantage of the Feynman–Kac formalism is that it allows us to define *predictive* representations of the Markovian model.

Definition 3.13 (Predictive distribution). *Let $(\pi_t)_{t=0}^T$ be a Markovian model given by a Feynman–Kac representation* (3.8). *The one-step* predictive distribution *of the state X_{t+1} is defined as*

$$\pi_t(\mathrm{d}x_{t+1}) = \int M_{t+1}(\mathrm{d}x_{t+1} \mid x_t)\pi_t(\mathrm{d}x_t). \tag{3.9}$$

When the representation considered is the bootstrap representation of Example 3.12, the predictive distribution is given by

$$\begin{aligned}\pi_t(\mathrm{d}x_{t+1}) &= \int p_{t+1}(\mathrm{d}x_{t+1} \mid x_t)\pi_t(\mathrm{d}x_t),\\ &= p(\mathrm{d}x_{t+1} \mid y_{0:t}).\end{aligned} \tag{3.10}$$

Several other predictive distributions can be defined, such as the k-step predictive distribution $\pi_t(\mathrm{d}x_{t+k})$ for $k \in \mathbb{N}$, or the *joint* predictive distribution $\pi_t(\mathrm{d}x_{t+1}, \mathrm{d}x_{t+2})$.

Remark 3.14. *Feynman–Kac representations form the basis or the justification of several publications included in this thesis, for instance, Publication I is partially interested in finding "good" representations automatically, while Publications V and VII use them in combination with 'auxiliary' variables, helping the design of the Feynman–Kac representation.*

3.2 Exact state inference in Gaussian Markovian models

In this section, we derive the filtering, smoothing, and pathwise smoothing distributions for linear Gaussian state-space models, a class of SSMs where the transition and observation models are affine and Gaussian.

3.2.1 Linear Gaussian state-space models

In this section, we focus on a particular class of Markovian models, known as linear Gaussian state-space models (LGSSMs). This class of models, albeit restrictive, is of particular interest as it forms the basic computational building block for inference in many other non-linear and non-Gaussian models.

Formally, a linear Gaussian state-space model is a state-space model as in Example 3.3, where the transition kernel $p_{t+1}(x_{t+1} \mid x_t)$ and the observation kernel $p_t(y_t \mid x_t)$ are affine and Gaussian distributions:

$$\begin{aligned} p_{t+1}(x_{t+1} \mid x_t) &= \mathcal{N}(x_{t+1}; F_{t+1}x_t + b_{t+1}, Q_{t+1}), \quad p_0(x_0) = \mathcal{N}(x_0; m_0, P_0),\\ p_t(y_t \mid x_t) &= \mathcal{N}(y_t; H_t x_t + d_t, R_t),\end{aligned} \tag{3.11}$$

where F_{t+1}, b_{t+1}, Q_{t+1} are the *transition matrix*, *transition offset*, and *transition covariance* respectively, and H_t, d_t, R_t are the *observation matrix*, *observation offset* and *observation covariance* respectively, and the initial state x_0 is assumed to be Gaussian with mean m_0 and covariance P_0. This corresponds to the following Markovian model:

$$\begin{aligned}\pi_t(x_{0:t}) &= p(x_{0:t} \mid y_{0:t}),\\ &\propto p(x_{0:t}, y_{0:t}),\\ &= \mathcal{N}(x_0; m_0, P_0) \prod_{s=1}^{t} \mathcal{N}(x_s; F_s x_{s-1} + b_s, Q_s) \prod_{s=0}^{t} \mathcal{N}(y_s; H_s x_s + d_s, R_s),\end{aligned} \tag{3.12}$$

for $t \in \{0, \dots, T\}$.

Under the joint distribution $p(x_{0:T}, y_{0:T})$, the states $x_{0:T}$ and the observations $y_{0:T}$ are jointly Gaussian, so that one may then use Gaussian algebra (Särkkä and Svensson, 2023, Appendix A.1) to compute the filtering, smoothing and pathwise smoothing distributions in closed form. However, doing so naively would result in a cubic time and space complexity in the number of time steps T, whereby one would need to form a covariance matrix of size $(T d_x d_y) \times (T d_x d_y)$, where d_x is the dimension of the state space X and d_y is the dimension of the observation space Y. Instead, it is possible to leverage the Markovian structure of the model to compute the filtering and (marginal) smoothing distributions in closed form using the Kalman filter and the Rauch–Tung–Striebel smoother, respectively, whilst only requiring a linear time and space complexity in T. We come back to this in Sections 3.2.2 and 3.2.3.

3.2.2 Kalman filter

We start by deriving the filtering distribution $\pi_t(\mathrm{d}x_t)$ recursively using the Kalman filter (Kalman, 1960). The procedure is split into two steps: a prediction step, where we compute the predictive distribution of the state x_{t+1} given the observations $y_{0:t}$, and an update step, where we compute the filtering distribution of the state x_{t+1} given the observations $y_{0:t+1}$. Recall that the predictive distribution of the state x_{t+1} given the observations $y_{0:t}$ is defined as

$$\pi_t(\mathrm{d}x_{t+1}) = p(x_{t+1} \mid y_{0:t}) = \int p(x_{t+1} \mid x_t) \pi_t(\mathrm{d}x_t). \tag{3.13}$$

Proposition 3.15 (Prediction step)**.** *Let π_t be given as in (3.11). Suppose that the filtering distribution $\pi_t(\mathrm{d}x_t)$ is given by a Gaussian distribution $\mathcal{N}(x_t; m_t^f, P_t^f)$. Then the one-step predictive distribution $\pi_t(\mathrm{d}x_{t+1})$ is given by $\mathcal{N}(x_{t+1}; m_{t+1}^p, P_{t+1}^p)$ with*

$$m_{t+1}^p = F_{t+1} m_t^f + b_{t+1}, \quad P_{t+1}^p = F_{t+1} P_t^f F_{t+1}^\top + Q_{t+1}.$$

Once the prediction step has been performed, we can now compute the filtering distribution $\pi_{t+1}(\mathrm{d}x_{t+1})$ using the update step.

Proposition 3.16 (Update step)**.** *Let π_t be given as in (3.11). Suppose that the one-step predictive distribution $\pi_t(\mathrm{d}x_{t+1})$ is given by a Gaussian distribution $\mathcal{N}(x_{t+1}; m^p_{t+1}, P^p_{t+1})$. Then the filtering distribution $\pi_{t+1}(\mathrm{d}x_{t+1})$ is given by $\mathcal{N}(x_{t+1}; m^f_{t+1}, P^f_{t+1})$ with*

$$m^f_{t+1} = m^p_{t+1} + K_{t+1}(y_{t+1} - H_{t+1} m^p_{t+1} - d_{t+1}), \quad P^f_{t+1} = (I - K_{t+1} H_{t+1}) P^p_{t+1},$$
$$K_{t+1} = P^p_{t+1} H^\top_{t+1} (H_{t+1} P^p_{t+1} H^\top_{t+1} + R_{t+1})^{-1},$$

where m^p_{t+1} and P^p_{t+1} are given by Proposition 3.15 and $m^p_0 = m_0$ and $P^p_0 = P_0$ by convention.

Proof. The proofs to both Propositions 3.15 and 3.16 follow from standard Gaussian algebra and can be found, for instance, in Särkkä and Svensson (2023, Chap. 6). □

The resulting algorithm is known as the Kalman filter, and is summarised in Algorithm 2. An important byproduct of the Kalman filter is that it allows us to

Algorithm 2: Kalman filter

```
input  : Initial distribution $p_0(\mathrm{d}x_0) \sim \mathcal{N}(x_0; m_0, P_0)$, transition and
         observation models $p_{t+1}(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1}; F_{t+1} x_t + b_{t+1}, Q_{t+1})$
         and $p_t(y_t \mid x_t) = \mathcal{N}(y_t; H_t x_t + d_t, R_t)$ for $t \in \{0, \dots, T\}$.
output : Filtering distributions $\pi_t(\mathrm{d}x_t) \sim \mathcal{N}(x_t; m^f_t, P^f_t)$ for $t \in \{0, \dots, T\}$.
1 for $t \in \{0, \dots, T\}$ do
2     ▷ Prediction step
3     if $t = 0$ then
4         Set $m^p_0 \leftarrow m_0$, $P^p_0 \leftarrow P_0$
5     else
6         Set $m^p_t \leftarrow F_t m^f_{t-1} + b_t$, $P^p_t \leftarrow F_t P^f_{t-1} F^\top_t + Q_t$
7     ▷ Update step
8     Set $K_t \leftarrow P^p_t H^\top_t (H_t P^p_t H^\top_t + R_t)^{-1}$
9     Set $m^f_t \leftarrow m^p_t + K_t(y_t - H_t m^p_t - d_t)$, $P^f_t \leftarrow (I - K_t H_t) P^p_t$
```

compute the marginal likelihood of the observations $p(y_{0:t})$ recursively.

Proposition 3.17 (Marginal likelihood)**.** *Let π_t be given as in (3.11). Suppose that the predictive distribution $p(x_{t+1} \mid y_{0:t})$ is given by a Gaussian distribution $\mathcal{N}(x_{t+1}; m^p_{t+1}, P^p_{t+1})$. Then the marginal likelihood of the observations up to time t is given by*

$$\frac{p(y_{0:t})}{p(y_{0:t-1})} = \mathcal{N}(y_t; H_t m^p_t + d_t, H_t P^p_t H^\top_t + R_t),$$

where $p(y_{0:t-1})$ is the marginal likelihood of the observations up to time $t-1$ and by convention $p(y_{0:-1}) = 1$.

Proof. The proof follows from noticing that

$$\begin{aligned}\frac{p(y_{0:t})}{p(y_{0:t-1})} &= p(y_t \mid y_{0:t-1}),\\ &= \int p(y_t \mid x_t) p(x_t \mid y_{0:t-1}) \mathrm{d}x_t,\\ &= \int \mathcal{N}(y_t; H_t x_t + d_t, R_t) \mathcal{N}(x_t; m_t^p, P_t^p) \mathrm{d}x_t.\end{aligned}$$

□

This result is crucial in the context of parameter estimation, as it allows us to compute the marginal likelihood of the observations $p(y_{0:t} \mid \theta)$ where θ is a parameter of the model, and then perform any form of inference on the parameter θ as per Chapter 2. We will come back to this in Section 3.4.4.

3.2.3 Pathwise sampling and marginal smoothing

In Section 3.2.2, we have seen how to compute the filtering distribution $p(x_t \mid y_{0:t})$ recursively using the Kalman filter, with a recursive application of Propositions 3.15 and 3.16. In this section, we turn to the pathwise smoothing problem, and derive a recursive procedure to sample from the *joint* smoothing distribution $p(\mathrm{d}x_{0:T} \mid y_{0:T})$. As a byproduct, we will also be able to compute the marginal smoothing distribution $p(x_t \mid y_{0:T})$ for all $t \in \{0, \dots, T\}$, recovering the classical Rauch–Tung–Striebel (RTS) smoother (Rauch et al., 1965).

To sample from $\pi_T(\mathrm{d}x_{0:T})$, it suffices to sample from $\pi_T(\mathrm{d}x_T)$, and then recursively sample from $\pi_T(\mathrm{d}x_{T-1} \mid x_T)$, $\pi_T(\mathrm{d}x_{T-2} \mid x_{T-1:T})$, and so on, until we reach $\pi_T(\mathrm{d}x_0 \mid x_{1:T})$. Noticing that the Markov property of the model implies that $\pi_T(\mathrm{d}x_t \mid x_{t+1:T}) = \pi_T(\mathrm{d}x_t \mid x_{t+1})$, we therefore only need to characterise the conditional distribution $\pi_T(\mathrm{d}x_t \mid x_{t+1})$ for all $t \in \{0, \dots, T-1\}$.

In order to do so, we again assume that we have already computed the filtering distributions $\pi_t(\mathrm{d}x_t) \sim \mathcal{N}(x_t; m_t^f, P_t^f)$ as well as the predictive ones $\pi_t(\mathrm{d}x_{t+1}) = \mathcal{N}(x_{t+1}; m_{t+1}^p, P_{t+1}^p)$, following Algorithm 2. Under this assumption, the following proposition gives the conditional distribution $\pi_T(\mathrm{d}x_t \mid x_{t+1})$.

Proposition 3.18. *Let π_t be given by (3.11), then*

$$\begin{aligned}\pi_T(x_t \mid x_{t+1}) &= p(x_t \mid x_{t+1}, y_{0:T}),\\ &= \mathcal{N}(x_t; L_t x_{t+1} + f_t, S_t),\end{aligned} \tag{3.14}$$

with

$$L_t = P_t^f F_{t+1}^\top (P_{t+1}^p)^{-1}, \quad f_t = m_t^f - L_t m_{t+1}^p, \quad S_t = P_t^f - L_t P_{t+1}^p L_t^\top, \tag{3.15}$$

where m_t^f, m_{t+1}^p, P_t^f, and P_{t+1}^p are given by Propositions 3.15 and 3.16, respectively.

Proof. This follows from noticing that x_{t+1} takes the role of an observation in the model π_t, and applying Proposition 3.16, replacing y_t by x_{t+1}, and H_t by F_{t+1}, d_t by b_{t+1} and R_t by Q_{t+1}. □

In turn, this allows us to derive the marginal smoothing distribution $\pi_T(\mathrm{d}x_t)$ by integrating out the state x_{t+1} from the conditional distribution $\pi_T(\mathrm{d}x_t \mid x_{t+1})$ with respect to the smoothing distribution $\pi_T(\mathrm{d}x_{t+1})$. This is given by the following proposition.

Proposition 3.19 (RTS smoothing)**.** *Let π_t be given as in (3.11), then the smoothing distribution $\pi_T(\mathrm{d}x_t)$ is also Gaussian $\mathcal{N}(x_t; m^s_t, P^s_t)$ with*

$$m^s_t = m^f_t + L_t(m^s_{t+1} - m^p_{t+1}), \quad P^s_t = P^f_t + L_t(P^s_{t+1} - P^p_{t+1})L_t^\top, \tag{3.16}$$

where m^f_t, m^p_{t+1}, P^f_t, and P^p_{t+1} are given by Propositions 3.15 and 3.16, respectively, and L_t is given by Proposition 3.18.

In both cases, the recursion is then started by noting that the last marginal smoothing distribution $\pi_T(\mathrm{d}x_T) \sim \mathcal{N}(x_T; m^f_T, P^f_T)$ is given by the filtering distribution at time T.

We summarise both the pathwise sampling and the RTS smoother in Algorithm 3.

Algorithm 3: LGSSM smoother

```
input :Transition models $p_{t+1}(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1}; F_{t+1}x_t + b_{t+1}, Q_{t+1})$,
       filtering distributions $p(x_t \mid y_{0:t}) \sim \mathcal{N}(x_t; m^f_t, P^f_t)$, and predictive
       distributions $p(x_{t+1} \mid y_{0:t}) = \mathcal{N}(x_{t+1}; m^p_{t+1}, P^p_{t+1})$ for $t \in \{0, \dots, T\}$.
output:Smoothing distributions $p(x_t \mid y_{0:T}) \sim \mathcal{N}(x_t; m^s_t, P^s_t)$ for
       $t \in \{0, \dots, T\}$, and pathwise smoothing trajectory
       $X_{0:T} \sim p(x_{0:T} \mid y_{0:T})$.
1 Set $m^s_T \leftarrow m^f_T$ and $P^s_T \leftarrow P^f_T$
2 Sample $X_T \sim \mathcal{N}(m^s_T, P^s_T)$
3 for $t \in \{T-1, \dots, 0\}$ do
4     Set $L_t \leftarrow P^f_t F^\top_{t+1}(P^p_{t+1})^{-1}$, $f_t \leftarrow m^f_t - L_t m^p_{t+1}$, and $S_t \leftarrow P^f_t - L_t P^p_{t+1} L^\top_t$
5     Set $m^s_t \leftarrow L_t m^s_{t+1} + f_t$ and $P^s_t \leftarrow S_t + L_t P^s_{t+1} L^\top_t$
6     Sample $X_t \sim \mathcal{N}(L_t X_{t+1} + f_t, S_t)$
```

3.2.4 An application: Gaussian process regression in linear time

In this section, we discuss an application of the Kalman filter and RTS smoother to machine learning, namely to the problem of Gaussian process regression. Gaussian processes are a class of statistical models that are widely used in machine learning (Rasmussen and Williams, 2006), and are particularly popular in the context of Bayesian optimisation (see, e.g., Frazier, 2018). They are defined as a probability distribution over functions $f \sim \mathrm{GP}(m, k)$, where $m(\cdot)$ is a mean function and $k(\cdot,\cdot)$ is a covariance function (also known as a kernel function). This distribution is characterised by its finite-dimensional distributions, which are jointly Gaussian distributions with mean and pairwise covariance given by

$$\mathbb{E}[f(t)] = m(t), \quad \mathbb{C}[f(t), f(t')] = k(t,t'), \tag{3.17}$$

where $t, t' \in \mathbb{R}$ are two arbitrary points. Given a data set $\mathscr{D} = \{(t_i, y_i)\}_{i=1}^n$ of n pairs of input-observations of the function f, where for all $i \in \{1,\ldots,n\}$, $y_i \sim \mathcal{N}(f(t_i), \sigma^2)$, the goal of Gaussian process regression is to compute the posterior distribution of the function f given the data set $\mathscr{D}$, i.e., to compute $p(f(s) \mid \mathscr{D})$ for a new query point $s \in \mathbb{R}$. Thankfully, this distribution is Gaussian and has mean and covariance given by

$$\mathbb{E}[f(s) \mid \mathscr{D}] = m(s) + k(s,\mathbf{t})(\mathbf{K} + \sigma_y^2 I)^{-1}(\mathbf{y} - m(\mathbf{t})), \tag{3.18}$$

$$\mathbb{C}[f(s), f(s') \mid \mathscr{D}] = k(s,s') - k(s,\mathbf{t})(\mathbf{K} + \sigma_y^2 I)^{-1} k(\mathbf{t}, s'), \tag{3.19}$$

where $\mathbf{t} = (t_1,\ldots,t_n)$, $\mathbf{Y} = (y_1,\ldots,y_n)$, $\mathbf{K} := (k(t_i,t_j))_{i,j=1}^n$, $k(s,\mathbf{X}) := (k(s,t_i))_{i=1}^n$, $k(\mathbf{X},z) := k(z,\mathbf{X})^\top$, I is the identity matrix of size $n \times n$, and z, z' are two arbitrary points in the input space X.

However, this typically incurs a cubic time and space complexity in the number of data points n, as one needs to solve systems involving the covariance matrix $\mathbf{K}$ which has size $n \times n$. Thankfully, when the covariance function k is isotropic, i.e., when $k(x,x') = k(\|x - x'\|)$ for some function $k : \mathbb{R} \to \mathbb{R}$, it is possible to approximate, and sometimes obtain an exact representation for, the Gaussian process prior distribution in terms of a stochastic differential equation (SDE, see, e.g., Särkkä and Solin, 2019). A typical example is the Matérn covariance function (Rasmussen and Williams, 2006; Stein, 2012), often used to represent functions with a predefined degree of smoothness and defined as

$$k^\nu(t,t';\sigma,\ell) := \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\|t-t'\|}{\ell}\right)^\nu K_\nu\left(\sqrt{2\nu}\frac{\|t-t'\|}{\ell}\right), \tag{3.20}$$

where $\nu > 0$ is the smoothness parameter, $\ell > 0$ is the lengthscale parameter, $\sigma^2 > 0$ is the variance parameter, and K_ν is the modified Bessel function of the second kind of order ν. Notably, when $\nu = m + \frac{1}{2}$ for some $m \in \mathbb{N}$, functions sampled from a Gaussian process with Matérn covariance function are m times differentiable almost surely and the covariance function simplifies. For instance, when $\nu = \frac{1}{2}$, the Matérn covariance function simplifies to

$$k^{\frac{1}{2}}(t,t';\sigma,\ell) = \sigma^2 \exp\left(-\frac{\|t-t'\|}{\ell}\right), \tag{3.21}$$

corresponding to the stationary covariance of the stochastic differential equation

$$\begin{aligned} \mathrm{d}x_t &= -\ell^{-1} x_t \mathrm{d}t + \sqrt{2}\sigma\ell^{-1/2}\mathrm{d}W_t, \\ p(x_0) &= \mathcal{N}(x_0; 0, \sigma^2), \end{aligned} \tag{3.22}$$

where W_t is a standard Brownian motion. This is known as the Ornstein–Uhlenbeck process (Uhlenbeck and Ornstein, 1930), and is a Markov process with continuous sample paths. Its transition density is given by

$$p(x_{t+h} \mid x_t) = \mathcal{N}\left(x_{t+h}; e^{-\frac{h}{\ell}} x_t, \sigma^2 \left(1 - e^{-\frac{2h}{\ell}}\right)\right) \tag{3.23}$$

for any $h > 0$, $t \geq 0$.

As a consequence, when the mean function $m \equiv 0$ is null, the problem of inference in a Gaussian process with Matérn covariance function for $\nu = 1/2$ can be reduced to the problem of inference in the continuous-time system with discrete-time observations

$$\begin{aligned} \mathrm{d}x_t &= -\ell^{-1} x_t \mathrm{d}t + \sqrt{2}\sigma\ell^{-1/2} \mathrm{d}W_t, \\ p(x_0) &= \mathcal{N}(x_0; 0, \sigma^2), \\ p(y_{t_k} \mid x_{t_k}) &= \mathcal{N}(y_{t_k}; x_{t_k}, \sigma_y^2), \quad k = 0, \dots, n, \end{aligned} \tag{3.24}$$

where $0 = t_0, t_1, \dots, t_n$ are the input points of the data set $\mathcal{D}$ and the y_{t_k} are the corresponding observations.

Given query points $s_1, \dots, s_m$, we can form the array of sorted time steps $0 = t'_0 \leq t'_1 \leq \dots \leq t'_{m+k}$, comprising the input points of the data set $\mathcal{D}$ and the query points. Writing $u_k = x_{t'_k}$, $v_k = y_{t_k}$ if $t'_k \in \{t_j; j = 0, \dots, n\}$, and $v_k = \emptyset$ otherwise, we can then consider the discrete-time system

$$\begin{aligned} \pi_{n+m}(u_{0:n+1}) \propto \mathcal{N}(u_0; 0, \sigma^2) \prod_{k=1}^{m+n} \mathcal{N}\left(u_k; e^{-\frac{1}{\ell}(t'_k - t'_{k-1})} u_{k-1}, \sigma^2 \left(1 - e^{-2\frac{t'_k - t'_{k-1}}{\ell}}\right)\right) \\ \times \prod_{k=0}^{n+m} p_k(v_k \mid u_k), \end{aligned} \tag{3.25}$$

where, by convention, $p_k(v_k \mid u_k) \equiv 1$ if $v_k = \emptyset$ and $p_k(v_k \mid u_k) = \mathcal{N}(v_k; u_k, \sigma_y^2)$ otherwise, corresponding to a LGSSM with missing observations.

The problem of Gaussian process regression can then be solved by using the Kalman filter and RTS smoother (presented in Sections 3.2.2 and 3.2.3, respectively) to compute the posterior distribution of the states $u_{0:n+m}$ which contain the states at the query points $s_1, \dots, s_m$.

For more details on constructing SDE representations for Gaussian processes, we refer the reader to Hartikainen and Särkkä (2010); Särkkä et al. (2013).

Finally, we note that this procedure can be further extended to (i) higher dimensions by considering spatio-temporal models, where the resulting dynamics are given by stochastic partial differential equations (SPDEs, see, e.g., Särkkä et al., 2013; Lindgren et al., 2022), (ii) as well as non Gaussian observations and non-stationary covariance functions by means of approximating (variational) methods (Wilkinson et al., 2023), such as those we discuss in the following section.

Remark 3.20. *This representation of Gaussian process regression as a linear Gaussian state-space model is the basis of Publication III, where we consider its parallelism, and efficient parameter estimation within these models.*

3.3 Gaussian approximated state-space models

Often, the state-space model of interest is not Gaussian, and the Kalman filter and RTS smoother of Section 3.2 cannot be applied directly. A natural approach is then to approximate the model $\pi_T(x_{0:T}) \propto p(x_{0:T}, y_{0:T})$ by an approximate LGSSM $\hat{\pi}_T(x_{0:T}) \propto \hat{p}(x_{0:T}, y_{0:T})$ to then apply the Kalman filter and Rauch–Tung–Striebel smoother to the approximated model.

3.3.1 Online approximation

Non-Gaussianity in the model π_T arises as soon as the weight function Γ_{t+1} in (3.1) is not quadratic in x_{t+1} and x_t jointly.

Example 3.21. *Consider a one-dimensional state-space model*

$$p_t(x_t \mid x_{t-1}) = \mathcal{N}(x_t; F_t x_{t-1} + b_t, Q_t),$$

$$p_t(y_t \mid x_t) = \mathcal{N}(y_t; |x_t| + c_t, R_t),$$

then Γ_{t+1} is only quadratic in x_t and not in x_{t+1}, making the model non-Gaussian.

Alternatively, consider a one-dimensional state-space model

$$p_t(x_t \mid x_{t-1}) = \mathcal{N}(x_t; x_{t-1}^2 + b_t, Q_t),$$

$$p_t(y_t \mid x_t) = \mathcal{N}(y_t; x_t + c_t, R_t),$$

then it is easy to see that there will exist a function μ, such that $p(x_{t+1} \mid y_t, x_t) \sim \mathcal{N}(x_{t+1}; \mu(x_t), W_{t+1})$, but μ will be non-linear. In other terms, Γ_{t+1} is only quadratic in x_{t+1} and not in x_t, making the model non-Gaussian too.

In order to perform (approximate) inference therein, it is therefore natural to try to approximate the transition and observation distributions by Gaussian distributions $\mathcal{N}(F_{t+1}x_t + b_{t+1}, Q_{t+1})$ and $\mathcal{N}(H_t x_t + d_t, R_t)$ respectively, where F_{t+1}, b_{t+1}, Q_{t+1}, H_t, d_t and R_t are parameters to be optimised. A natural approach to do so is to sequentially linearise the transition and observation models with respect to the current approximate filtering mean m_t^f and covariance P_t^f. This approach gives rise to the celebrated extended and unscented Kalman filter (Julier and Uhlmann, 2004) and can be achieved by applying a first-order Taylor expansion (respectively, an unscented transform, Julier et al., 2000) of the transition and observation models around the current approximate filtering mean m_t^f and covariance P_t^f following Section 2.3.4. We summarise the resulting (first-order) extended approximation for when the model has Gaussian additive noise:

$$\begin{aligned} p_{t+1}(x_{t+1} \mid x_t) &= \mathcal{N}(x_{t+1}; \mu_{t+1}^x(x_t), Q_{t+1}), \\ p_t(y_t \mid x_t) &= \mathcal{N}(y_t; \mu_t^y(x_t), R_t), \end{aligned} \tag{3.26}$$

in Algorithm 4 and refer to Särkkä and Svensson (2023, Chap. 7 and 8) for a detailed discussion of extensions to non-additive Gaussian noise models and other linearisation methods. A similar algorithm exists to compute the ap-

Algorithm 4: Extended Kalman filter

```
input  : Initial distribution $p_0(\mathrm{d}x_0) \sim \mathcal{N}(x_0; m_0, P_0)$, transition and
         observation mean functions $\mu_t^x$, $\mu_t^y$ and covariances $Q_t$, $R_t$.
output : Approximate distributions $\hat{\pi}_t(\mathrm{d}x_t) \sim \mathcal{N}(x_t; m_t^f, P_t^f)$ for
         $t \in \{0, \dots, T\}$.
1 for $t \in \{0, \dots, T\}$ do
2     ▷ Transition linearisation step
3     Set $F_t \leftarrow \nabla \mu_t^x(m_{t-1}^f)$ and $b_t \leftarrow \mu_t^x(m_{t-1}^f) - F_t m_{t-1}^f$
4     ▷ Prediction step
5     Apply Proposition 3.15 with $F_t$, $b_t$ and $Q_t$ to compute $m_t^p$ and $P_t^p$
6     ▷ Observation linearisation step
7     Set $H_t \leftarrow \nabla \mu_t^y(m_t^p)$ and $c_t \leftarrow \mu_t^y(m_t^p) - H_t m_t^p$
8     ▷ Update step
9     Apply Proposition 3.16 with $H_t$, $c_t$ and $R_t$ to compute $m_t^f$ and $P_t^f$
```

proximate smoothing distributions $\pi_T(\mathrm{d}x_t)$ for all $t \in \{0, \dots, T\}$, known as the extended Rauch–Tung–Striebel smoother (Cox, 1964). Additionally, while we have discussed the case of additive Gaussian noise models, this technique can be used to tackle non-additive Gaussian noise models, in a fashion similar to Section 2.3.4. We refer to Särkkä and Svensson (2023) for a detailed discussion of these techniques.

Similarly to Section 2.3.4, linearising the transition and observation models around the current approximate filtering mean m_t^f and covariance P_t^f may not be sufficient to obtain a good approximation of the model $\pi_t(\mathrm{d}x_t)$. Iterative linearisation methods can be used to improve the approximation, such as the iterated extended Kalman filter (IEKF, Bell and Cathey, 1993) and the iterated posterior linearisation filter (IPLF, García-Fernández et al., 2016), which recompute the linearisations of the transition and observation models (steps 3 and 7 in Algorithm 4) using the current approximation to filtering distribution $\hat{\pi}_t(\mathrm{d}x_t) \sim \mathcal{N}(x_t; m_t^f, P_t^f)\mathrm{d}x_t$ at each time step t. In the next section, we discuss how this iterative procedure can be used to obtain precise approximations of the smoothing distributions $p(x_t \mid y_{0:T})$ rather than the filtering distributions $p(x_t \mid y_{0:t})$.

3.3.2 Offline approximation

In Section 3.3.1, we have seen how to approximate the filtering distributions π_t for all $t \in \{0, \dots, T\}$ by Gaussian models $(\widetilde{\pi}_t)_{t=0}^T$ by linearising the transition and observation models around the current approximate filtering mean m_t^f. In this section, we turn to the problem of approximating the full smoothing distribution π_T for (3.26) by a Gaussian model $\widetilde{\pi}_T$. The method we present is known as the iterated extended Kalman smoother (IEKS, Bell, 1994) and was extended to more general models by Tronarp et al. (2018) under the name of iterated posterior linearisation smoothers (IPLS).

The idea of the IEKS is to express the MAP of the smoothing distribution π_T as the solution of a non-linear least-squares problem, and then to use the Gauss–Newton algorithm (see, e.g., Boyd et al., 2004, Chap. 4) to solve it. This is achieved by noticing that the MAP of the smoothing distribution π_T is given by

$$
\begin{aligned}
\operatorname*{arg\,max}_{x_{0:T}} \log \pi_T(x_{0:T}) &= \operatorname*{arg\,max}_{x_{0:T}} \log p(y_{0:T} \mid x_{0:T}) + \log p(x_{0:T}), \\
&= \operatorname*{arg\,max}_{x_{0:T}} \sum_{t=0}^{T} \log \mathcal{N}(y_t; \mu_t^y(x_t), R_t) + \log \mathcal{N}(x_t; \mu_t^x(x_{t-1}), Q_t),
\end{aligned}
\tag{3.27}
$$

where by convention we have taken $x_{-1} = \emptyset$ to be a "null" element, so that $\mu_0^x(x_{-1}) = m_0$, and $Q_0 = P_0$. Now the individual terms of the sum in (3.27) are Gaussian, and therefore we can rewrite the maximisation problem (3.27) as a non-linear least-squares problem

$$
\operatorname*{arg\,min}_{x_{0:T}} \sum_{t=0}^{T} \left\| y_t - \mu_t^y(x_t) \right\|_{R_t}^2 + \left\| x_t - \mu_t^x(x_{t-1}) \right\|_{Q_t}^2, \tag{3.28}
$$

where $\|\cdot\|_Q = \sqrt{\cdot^\top Q^{-1} \cdot}$ is the Mahalanobis distance associated with the covariance matrix Q. Now, write $M = \operatorname{diag}(Q_0^{-1/2}, R_0^{-1/2}, \dots, Q_T^{-1/2}, R_T^{-1/2})$ and

$$
\mu(x_{0:T}) = \begin{pmatrix} x_0 - \mu_0^x(x_{-1}) & y_0 - \mu_0^y(x_0) & \dots & x_T - \mu_T^x(x_{T-1}) & y_T - \mu_T^y(x_T) \end{pmatrix}^\top, \tag{3.29}
$$

then the least-squares problem (3.28) can be rewritten as

$$
\operatorname*{arg\,min}_{x_{0:T}} \left\| M \mu(x_{0:T}) \right\|_2^2. \tag{3.30}
$$

The Gauss–Newton algorithm provides a natural iterative procedure to solve this least-squares problem: suppose that we have a guess $x_{0:T}^{(k)}$ of the solution $x_{0:T}$ at iteration k, then we can linearise the function μ around $x_{0:T}^{(k)}$ and solve the resulting linear least-squares problem to obtain a new guess $x_{0:T}^{(k+1)}$. Formally, this corresponds to taking

$$
x_{0:T}^{(k+1)} = \operatorname*{arg\,min}_{x_{0:T}} \left\| M \mu(x_{0:T}^{(k)}) + M \nabla \mu(x_{0:T}^{(k)})^\top (x_{0:T} - x_{0:T}^{(k)}) \right\|_2^2. \tag{3.31}
$$

We can then note that this is the MAP of the following LGSSM distribution

$$
\begin{aligned}
\pi_T^k(\mathrm{d}x_{0:T}) \propto & \prod_{t=0}^{T} \mathcal{N}\left(x_t; \mu_t^x(x_{t-1}^{(k)}) + \nabla \mu_t^x(x_{t-1}^{(k)})^\top (x_t - x_t^{(k)}), Q_t\right), \\
& \times \prod_{t=0}^{T} \mathcal{N}\left(y_t; \mu_t^y(x_t^{(k)}) + \nabla \mu_t^y(x_t^{(k)})^\top (x_t - x_t^{(k)}), R_t\right).
\end{aligned} \tag{3.32}
$$

Now, because the mode and the mean of a Gaussian distribution coincide, we can rewrite the MAP of the smoothing distribution π_T^k as the posterior mean for the same Gaussian model, and therefore we can apply the Kalman filter of Section 3.2.2 to obtain a new guess $x_{0:T}^{(k+1)}$. When the algorithm has converged (for example, once the differences between successive MAP estimates $x_{0:T}^{(k+1)}$ are small enough), we can then use the final approximate smoothing distribution π_T^k as our approximation $\hat{\pi}_T$ of the true smoothing distribution π_T. This is summarised in Algorithm 5.

Algorithm 5: Iterated extended Kalman smoother

```
input  : Initial distribution $p_0(\mathrm{d}x_0) \sim \mathcal{N}(x_0; m_0, P_0)$, transition and
         observation mean functions $\mu_t^x$, $\mu_t^y$ and covariances $Q_t$, $R_t$, initial
         guess $x_{0:T}^{(0)}$.
output : Approximate distributions $\hat{\pi}_T(\mathrm{d}x_t) \sim \mathcal{N}(x_t; m_t^s, P_t^s)$ for
         $t \in \{0, \dots, T\}$.
1 while not converged do
2     Form the model $\pi_T^k$ by linearising the transition and observation
        models around $x_{0:T}^{(k)}$ following (3.32)
3     Apply the Kalman filter of Algorithm 2 to compute the approximate
        filtering distributions $\pi_t^k(\mathrm{d}x_t) \sim \mathcal{N}(x_t; m_t^f, P_t^f)$ for $t \in \{0, \dots, T\}$
4     Apply the RTS smoother of Algorithm 3 to compute the approximate
        smoothing distributions $\pi_T^k(\mathrm{d}x_t) \sim \mathcal{N}(x_t; m_t^s, P_t^s)$ for $t \in \{0, \dots, T\}$
5     Set $x_{0:T}^{(k+1)} \leftarrow m_{0:T}^s$ and $k \leftarrow k + 1$.
```

Because Gauss–Newton is known to be a first-order method (Nocedal and Wright, 1999, Chap. 10), the iterated extended Kalman smoother converges linearly fast to the MAP of the smoothing distribution π_T, provided that the initial guess $x_{0:T}^{(0)}$ is close enough to the true MAP of the smoothing distribution π_T. In practice, this is not always the case, and the IEKS may converge slowly or not at all; this has prompted the development of alternative methods, such as line-search methods in (Särkkä and Svensson, 2020).

Remark 3.22. *Iterated smoothing forms the basis of Publication II and its follow-up Yaghoobi et al. (2022), where we develop parallel extensions of the IPLS, as well as numerically stable versions thereof.*

3.3.3 Parameter identification

We end this section by discussion the problem of *pointwise* parameter identification in Gaussian approximated state-space models. Formally, we are given a state-space model $\pi_T(x_{0:T};\theta) \propto p(x_{0:T}, y_{0:T};\theta)$ depending on a parameter $\theta \in \Theta$ of interest, and we are interested in computing the MAP of the parameter θ given the observations $y_{0:T}$, i.e., the maximum of $\pi_T(\theta) = p(\theta \mid y_{0:T}) \propto p(y_{0:T} \mid \theta)p(\theta)$ for a prior distribution $p(\theta)$. When the model is Gaussian, this problem can be solved by employing, for example, expectation-maximisation (EM, Dempster et al., 1977; Meng and Van Dyk, 2002), provided that the parameter θ describes the full model, i.e., that $\theta = \{m_0, P_0\} \cup \{F_t, b_t, Q_t, H_t, c_t, R_t; t = 0, \dots, T\}$ or a subset thereof. However, when the model is non-Gaussian, or when θ is fed to the model in a hierarchical form, this approach is not applicable. This is the case for Gaussian models when F_t^θ and other parameters are functions of θ, such as in Section 3.2.4, where we can have $\theta = (\sigma_y, \sigma, \ell)$.

In this case, one needs to resort to numerical methods. Thankfully, the Kalman filter (Algorithm 2) returns the marginal likelihood of the observations $p(y_{0:T} \mid \theta)$ as a byproduct, and therefore one can use gradient-based methods to maximise $p(\theta \mid y_{0:T})$ or compute any variational approximation thereof. This is typically made easier by the use of automatic differentiation (see, e.g., Baydin et al., 2018, for a survey), which can seamlessly be applied to the Kalman filter to compute the gradient of the marginal likelihood $p(y_{0:T} \mid \theta)$ with respect to θ at the same computational cost as the Kalman filter itself. Furthermore, when the model is non-Gaussian, one can then iteratively

1. Compute a LGSSM approximation $\hat{\pi}_T(x_{0:T};\theta)$ of the model $\pi_T(x_{0:T};\theta)$ using the iterated extended Kalman smoother of Algorithm 5.
2. Compute the approximate normalising constant, corresponding to marginal likelihood $\hat{p}(y_{0:T} \mid \theta)$ with respect to θ under the approximation, and using automatic differentiation, take a gradient ascent step with respect to θ.

A caveat to this approach is that the approximate model $\hat{\pi}$ parameters implicitly depend on θ, and therefore the gradient of the approximate marginal likelihood $\hat{p}(y_{0:T} \mid \theta)$ with respect to θ naively needs to be propagated through the iterated extended Kalman smoother of Algorithm 5. Thankfully, the fixed-point structure of IEKS, as well as the implicit function theorem, allows one to bypass this issue (Christianson, 1994). We refer to Yaghoobi et al. (2022) for a detailed discussion of this approach within the context of iterated linearisation.

Remark 3.23. *In Publication III, we combine this, as well as some methods mentioned in Chapter 4, to develop a parallel and efficient method for Bayesian parameter estimation in Gaussian process regression models. Publication VI too is concerned with the problem of parameter estimation in state-space models, and develops a method based on the Wasserstein gradient flows.*

3.4 Monte Carlo approximations: particle filters and smoothers

In Section 3.2, we have seen how to compute the filtering and smoothing distributions for a class of Gaussian Markovian models and in Section 3.3, we have seen how non-Gaussian models could be approximated by Gaussian models to which the methods of Section 3.2 could be applied. In this section, we turn to a more general case, where we are not willing to make Gaussian approximations, and present Monte Carlo approximations to the filtering and smoothing distributions, known as particle filters and smoothers. Contrary to the rest of this section, we do not assume Markovianity of the model $(\pi_t)_{t=0}^T$, but instead, assume that it is given by the following recursive decomposition

$$\mathrm{d}\pi_t(x_{0:t}) \propto M_t(\mathrm{d}x_t \mid x_{0:t-1}) G_t(x_{0:t}) \mathrm{d}\pi_{t-1}(x_{0:t-1}) \tag{3.33}$$

and that we can sample from the transition kernels M_t and evaluate the potential functions G_t, respectively. Note that this recovers the formalism (3.8) for $G_t(x_{0:t}) = G_t(x_{t-1:t})$, and $M_t(\mathrm{d}x_t \mid x_{0:t-1}) = M_t(\mathrm{d}x_t \mid x_{t-1})$.

3.4.1 Sequential Monte Carlo

Sequential Monte Carlo methods are a class of samplers for (possibly non-Markovian) sequences of probability distributions π_t on $(\mathrm{X}^t, \mathscr{X}^t)$. Originally called sequential importance resamping methods (SIR), they were introduced by Gordon et al. (1993) to perform inference in highly non-linear multi-modal state-space models. They have since been extended to a wide range of applications, from their original state and parameter estimation purpose to standard Bayesian inference, by means of tempering (Del Moral et al., 2006), or optimisation (Gerber and Douc, 2021).

At heart, they rely on the importance sampling identity (2.13) to sample from the distribution π_t. Formally, suppose that $\pi_t^N(\mathrm{d}x_{0:t}) = \sum_{n=1}^N W_t^n \delta_{X_{0:t}^n}(\mathrm{d}x_{0:t})$ is an empirical approximation of the distribution π_t with N trajectories $X_{0:t}^n$ and weights W_t^n for $n \in \{1, \ldots, N\}$, then an approximation of the distribution π_{t+1} can be obtained by sampling $X_{0:t+1}^n \sim M_{t+1}(\mathrm{d}x_{t+1} \mid X_{0:t}^n) \delta_{X_{0:t}^n}(\mathrm{d}x_{0:t})$ and computing the weights

$$W_{t+1}^n = \frac{G_{t+1}(X_{0:t+1}^n) W_t^n}{\sum_{m=1}^N G_{t+1}(X_{0:t+1}^m) W_t^m}, \tag{3.34}$$

forming the empirical approximation $\pi_{t+1}^N(\mathrm{d}x_{0:t+1}) = \sum_{n=1}^N W_{t+1}^n \delta_{X_{0:t+1}^n}(\mathrm{d}x_{0:t+1})$. This procedure, known as sequential importance sampling (SIS), is consistent in the sense that π_{t+1}^N converges to π_{t+1} as $N \to \infty$ (see, e.g., Doucet and Johansen, 2011; Chopin and Papaspiliopoulos, 2020; Särkkä and Svensson, 2023). Furthermore, an important property of SIS is that $\sum_{n=1}^N W_{t-1}^n G_t(X_{0:t}^n)$ is an unbiased estimator of the incremental normalising constant $\int G_t(x_{0:t}) \mathrm{d}\pi_{t-1}(x_{0:t})$ (see, e.g., Doucet and Johansen, 2011).

However, it suffers from a degeneracy problem, where the weights W_t^n concentrate on a single trajectory $X_{0:t}^n$ as $t \to \infty$. This is due to the fact that the dimension of the space X^t is increasing with t, making importance sampling exponentially inefficient as t increases (Doucet and Johansen, 2011, Section 3), so that an exponentially large number of particles N is required to obtain a good approximation of the distribution π_t.

To alleviate this problem, Gordon et al. (1993) introduced the resampling step, which, originally, consists in sampling N trajectories with replacement from the empirical approximation π_t^N with probabilities proportional to the weights W_t^n prior to the propagation step. Intuitively, this procedure removes unsuccessful trajectories from the empirical approximation π_t^N and duplicates successful trajectories, thus preventing the weights W_t^n from concentrating on a single trajectory $X_{0:t}^n$. This procedure forms a genealogy of the trajectories $X_{0:t}^n$, whereby a particle X_t^n is associated with an ancestor $X_{t-1}^{A_{t-1}^n}$, itself associated with an ancestor $X_{t-2}^{A_{t-2}^{A_{t-1}^n}}$, and so on. Formally, this corresponds to the following recursive decomposition.

Definition 3.24 (Particles genealogy)**.** *Let A_{t-1}^n be the ancestor of the particle X_{t-1}^n at time $t-1$, then the genealogy of the particle X_t^n is given by*

$$X_{0:t}^{(n)} := \left(X_{0:t-1}^{(A_{t-1}^n)}, X_t^n\right). \tag{3.35}$$

This is the basis of the sequential importance resampling (SIR) algorithm, which is summarised in Algorithm 6.

Algorithm 6: Sequential importance resampling (SIR)

```
input  : Feynman–Kac model $(\pi_t)_{t=0}^T$ given by (3.33), number of particles
         $N$.
output : Approximate distributions $\pi_t^N(\mathrm{d}x_{0:t}) = \sum_{n=1}^N W_t^n \delta_{X_{0:t}^{(n)}}(\mathrm{d}x_{0:t})$ and
         normalising constant estimator $L_t^N$, for $t \in \{0,\dots,T\}$.
1 ▷ All operations involving the index $n$ need to be done for $n = 1,\dots,N$
2 Sample $X_0^n \sim M_0(\mathrm{d}x_0)$
3 Set $w_0^n \leftarrow G_0(X_{0:0}^n)$, $W_0^n \leftarrow \frac{w_0^n}{\sum_{m=1}^N w_0^m}$
4 Set $L_0^N \leftarrow \frac{1}{N}\sum_{m=1}^N w_0^m$
5 for $t \in \{1,\dots,T\}$ do
6     Sample $A_{t-1}^n \sim \mathrm{Multinomial}(W_{t-1}^{1:N})$          ▷ Multinomial resampling
7     Sample $X_t^n \sim M_t\left(\mathrm{d}x_t \mid X_{0:t-1}^{(A_{t-1}^n)}\right)$          ▷ Propagation
8     Set $w_t^n \leftarrow G_t(X_{0:t}^{(n)})$, $W_t^n \leftarrow \frac{w_t^n}{\sum_{m=1}^N w_t^m}$          ▷ Weighting
9     Set $L_t^N \leftarrow L_{t-1}^N \frac{1}{N}\sum_{m=1}^N w_t^m$          ▷ Normalising constant estimator
```

Consistency results for the SIR algorithm are available (see, e.g., Del Moral, 2004; Crisan and Doucet, 2002).

Proposition 3.25 (Unbiasedness of the normalising constant estimator)**.** *The normalising constant estimator L_t^N is an unbiased estimator of the normalising* *constant*

$$\int M_0(\mathrm{d}x_0)G_0(x_0)\prod_{s=1}^{t} G_s(x_{0:s})M_s(\mathrm{d}x_s \mid x_{0:s-1}). \tag{3.36}$$

Proposition 3.26 (Convergence of the empirical approximation)**.** *For any bounded and continuous function $\varphi\colon \mathsf{X}^{t+1} \to \mathbb{R}$, we have*

$$\mathbb{E}\left[\left(\sum_{n=1}^{N} W_t^n \varphi(X_{0:t}^{(n)}) - \pi_t(\varphi)\right)^2\right] \le C_t \frac{\|\varphi\|_\infty^2}{N}, \tag{3.37}$$

for some constant $C_t > 0$, and where $\|\varphi\|_\infty = \sup_{x_{0:t} \in \mathsf{X}^{t+1}} |\varphi(x_{0:t})|$.

We refer to Chopin and Papaspiliopoulos (2020) for a proof of these results and other convergence results such as an almost sure convergence, a central limit theorem, and the dependence of the constant C_t on t.

3.4.2 The resampling step

We now discuss the resampling step of the SIR algorithm, and some of its properties and variants.

Adaptive resampling. While we have written the SIR algorithm as applying the resampling step every time step t, it is possible to apply it only every k time steps, for some $k \in \{1, \dots, T\}$, as well as adaptively (Liu and Chen, 1995; Del Moral et al., 2012), for instance by using the effective sample size (Liu, 1996)

$$\mathrm{ESS}_t = \frac{1}{\sum_{n=1}^{N} (W_t^n)^2}, \tag{3.38}$$

where the resampling step is applied when $\mathrm{ESS}_t < \alpha N$ for a given threshold $\alpha \in (0,1)$ representing the number of "effective" particles that a non-importance sampling algorithm would need to achieve the same variance (see, e.g., Elvira et al., 2022, for a review). In this case, when there is no resampling, the weights appearing at step 8 in Algorithm 6 would simply need to be replaced with $W_t^n \propto W_{t-1}^n G_t(X_{0:t}^{(n)})$, the ancestors A_{t-1}^n to n, and the normalising constant estimator L_t^N would need to be updated accordingly. While no clear theoretical argument exists for their efficacy (however, see Section 5.2 for an analysis of a different algorithm, Del Moral et al., 2012), adaptive resampling schemes are known to empirically improve the performance of SIR algorithms (see, e.g., Chopin et al., 2022).

Unbiased resampling schemes. Moreover, the choice of the multinomial resampling in Algorithm 6, introduced by Gordon et al. (1993), is often suboptimal and introduces undesirable variance in expectation estimates by discarding particles unnecessarily. In order to alleviate this problem, other, low-variance, resampling schemes have been proposed, such as the systematic and stratified resampling (Bolić et al., 2004) techniques. These resampling schemes consist in implementing a *joint* discrete distribution over the indices A_{t-1}^n rather than sampling them independently. For the resampling to still allow SIR to return unbiased estimators of the normalising constant, it needs to verify the following property.

Definition 3.27 (Unbiased resampling)**.** *An unbiased resampling operation is a distribution* $\mathscr{R}(\cdot \mid (W^n)_{n=1}^N)$ *on* $\{1, \ldots N\}^N$ *verifying*

$$\mathbb{E}_{A \sim \mathscr{R}}\left[\frac{1}{N}\sum_{n=1}^{N}\varphi(X^{A^n})\right] = \sum_{n=1}^{N} W^n \varphi(X^n),$$

for all $(X^n)_{n=1}^N$ *and* φ.

Example 3.28. *The systematic resampling scheme is an unbiased resampling scheme, where the joint distribution* $\mathscr{R}^{\text{sys}}(\cdot \mid (W^n)_{n=1}^N)$ *is given by*

$$\begin{aligned} N^{-1}\mathscr{R}^{\text{sys}}(A^{1:N} \mid W^{1:N}) = \min&\left(1/N, C_{A^1} - \frac{0}{N}, C_{A^2} - \frac{1}{N}, \ldots, C_{A^N} - \frac{N-1}{N}\right) \\ &- \max\left(0, C_{A^1-1} - \frac{0}{N}, C_{A^2-1} - \frac{1}{N}, \ldots, C_{A^N-1} - \frac{N-1}{N}\right) \end{aligned}$$

where $C_n = \sum_{m=1}^n W^m$ *with* $C_0 = 0$. *This can be sampled from by first sampling* $U \sim \text{Uniform}\left(0, \frac{1}{N}\right)$ *and then setting* $A^n = \min\{m \in \{1, \ldots, N\} : U + \frac{n-1}{N} \leq C_m\}$ *for* $n \in \{1, \ldots, N\}$.

Another way to understand the unbiasedness of the resampling scheme is to look at the offspring distribution, rather than the ancestor distribution, that is, the number of copies of each particle X_{t-1}^n made by the resampling scheme.

Definition 3.29 (Offspring distribution)**.** *Suppose* $A^{1:N} \sim \mathscr{R}(\cdot \mid (W^n)_{n=1}^N)$*, the offspring of a particle with index m is defined as*

$$\mathscr{O}^m = \sum_{n=1}^{N} \mathbb{1}(A^n = m).$$

This provides an alternative characterisation of unbiasedness of the resampling scheme that is more flexible as it does not make assumptions on the specific position of the ancestors, and therefore allows for reordering of the particles.

Definition 3.30 (Unbiasedness of the resampling scheme)**.** *A resampling algorithm with offspring* $(\mathscr{O}^m)_{m=1}^N$ *is unbiased as soon as the following conditions are satisfied:*

- $\mathscr{O}^m \geq 0$ *a.s.,*
- $\sum_{m=1}^N \mathscr{O}^m = N$ *a.s.,*
- $\mathbb{E}[\mathscr{O}^m] = \sum_{n=0}^N n\mathbb{P}(\mathscr{O}^m = n) = NW^m$.

The second condition is related to the fact that all the particles are assigned an ancestor, and the third condition is related to the fact that the expected number of copies of a particle X_{t-1}^m is equal to the weight W_{t-1}^m. It is easy to see that if a resampling scheme is unbiased in the sense of Definition 3.27, then it is unbiased in the sense of Definition 3.30. On the other hand, if a resampling scheme is unbiased in the sense of Definition 3.30, then there exists a permutation thereof that is unbiased in the sense of Definition 3.27.

Unbiased resamplings are important because they allow one to obtain unbiased estimators of the normalising constant of the distribution π_t.

Proposition 3.31 (Unbiasedness of the normalising constant estimator)**.** *If the resampling scheme used in SIR is unbiased, then the normalising constant estimator L_t^N is an unbiased estimator of the normalising constant*

$$\int \prod_{s=0}^{t} G_s(x_{0:s}) M_s(\mathrm{d}x_s \mid x_{0:s-1}).$$

Proof. The proof of this result can be found, for example, in Chopin and Papaspiliopoulos (2020, Chap. 16). □

Remark 3.32. *The unbiasedness of the normalising constant estimator L_t^N is a desirable property, and will have important consequences in Chapter 4.*

Finally, a natural question is whether there exists an unbiased resampling scheme that is also optimal in some sense. This question has been answered in the positive only recently by Gerber et al. (2019) and Li et al. (2021) in the context of *ordered* resampling schemes, where the particles are ordered according to their position in the state space, and where the resampling scheme is required to preserve this ordering.

Biased resampling schemes. While unbiased resampling schemes are desirable as they provide unbiased estimators of the normalising constant, they may be suboptimal for other tasks, such as state estimation. Indeed, while optimal unbiased resampling schemes exist, providing the optimal asymptotic rate (in terms of N) of convergence of the empirical approximation π_t^N to the true distribution π_t (Gerber et al., 2019; Li et al., 2021), they may not be optimal in terms for a specific task, such as state estimation. This remark prompted the introduction of other, possibly biased, specialised resampling schemes.

An example is given by the ensemble transform resampling (ETR, Reich, 2013), which is a deterministic resampling scheme, where the weighted empirical distribution $\pi_t^N = \sum_{n=1}^N W_t^n \delta_{X_t^n}(\mathrm{d}x)$ is mapped to the closest unweighted empirical distribution $\widetilde{\pi}_t^N = \frac{1}{N}\sum_{n=1}^N \delta_{\widetilde{X}_t^n}(\mathrm{d}x)$ in the Wasserstein sense. This consists in minimising the expected mean squared error introduced by the resampling step, i.e., finding the set of resampled particles $\widetilde{X}_t^n$ solving

$$\arg\min_{\widetilde{X}_t^n} \mathcal{W}_2(\pi_t^N, \widetilde{\pi}_t^N), \tag{3.39}$$

where $\mathscr{W}_2$ is given in Definition 2.30. While this is a well-defined problem, and can be approached using gradient-flow techniques as in Section 2.3.5, it is not practical to solve in general, owing to the support of the target distribution not being fixed. Consequently, Reich (2013) proposed a practical approximation to solve it, which consists in computing the transport plan between the unweighted particles and the weighted particles and then using it to compute the resampled particles.

Formally, write $\hat{\pi}_t^N = \frac{1}{N}\sum_{n=1}^N \delta_{X_t^n}(\mathrm{d}x)$ to be the unweighted empirical distribution of the particles X_t^n, and consider the optimal transport problem

$$\arg\min_{\gamma \in \Gamma(\pi_t^N, \hat{\pi}_t{}^N)} \int_{\mathsf{X}^t \times \mathsf{X}^t} \|x - \hat{x}\|^2 \gamma(\mathrm{d}x, \mathrm{d}\hat{x}), \tag{3.40}$$

where $\Gamma(\pi_t^N, \hat{\pi}_t^N)$ is the set of joint distributions γ on $\mathsf{X}^t \times \mathsf{X}^t$ with marginals π_t^N and $\hat{\pi}_t{}^N$, which can be represented as the set of $N \times N$ matrices $\gamma = (\gamma_{ij})_{i,j=1}^N$ with $\gamma_{ij} \geq 0$ and $\sum_{j=1}^N \gamma_{ij} = W_t^i$ and $\sum_{i=1}^N \gamma_{ij} = \frac{1}{N}$. Because $\hat{\pi}_t^N$ represents the "prior" distribution of the particles, i.e., before applying importance weights, and π_t^N is the "posterior" distribution of the particles, i.e., after applying importance weights, the optimal transport problem (3.40) can be seen as finding an optimal (in the MSE sense) way to transport the prior to the posterior distribution.

Now, optimising (3.40) is equivalent (Hoffman et al., 1963) to solving the following linear program

$$\min_{\gamma \in \Gamma(W_t, \mathbb{1}/N)} \sum_{i,j=1}^N \gamma_{ij} \left\| x_i - x_j \right\|^2, \tag{3.41}$$

which can be solved in roughly $\mathscr{O}(N^3)$ operations (Pele and Werman, 2009). Once this has been done, for an optimal solution γ^*, the resampled particles $\widetilde{X}_t^n$ are given by

$$\widetilde{X}_t^n = N \sum_{m=1}^N \gamma_{mn}^* X_t^m, \tag{3.42}$$

corresponding to the barycentres of the particles X_t^m with weights γ_{mn}^*, or, in other words, corresponding to approximately transporting the unweighted distribution $\hat{\pi}_t^N$ to the weighted distribution π_t^N, using the optimal transport plan γ^*. This resampling scheme is biased, but has been shown to be asymptotically consistent in the sense that expectations under $\widetilde{\pi}_t^N$ converge to the true one as $N \to \infty$ (Reich, 2013).

Other resampling techniques, based on the same intuition that unbiasedness is not necessarily desirable, have been proposed. For example, Malik and Pitt (2011a,b) consider continuous but biased resampling schemes for one-dimensional state-space models; Murray (2012) introduce a biased but parallelisable resampling scheme based on Markov chain Monte Carlo methods (see Chapter 4); Acevedo et al. (2017) consider second-order accurate (i.e., which preserve the covariance estimator of the particle ensemble) of the ETR resampling scheme; and Kviman et al. (2024) seek to minimise the Kullback–Leibler divergence between the target empirical distribution π_t and the resampled distribution $\widetilde{\pi}_t^N$, which can be written as a recursive ELBO (see Section 2.3.3) maximisation problem.

Remark 3.33. *Publication I is largely based on the ensemble transform resampling scheme, although its main focus is on differentiability, while the focus of Reich (2013) was motivated by improving the expressivity of the particle filter.*

3.4.3 Sequential Monte Carlo for smoothing

In Section 3.4.1, we have seen how to sample from the running smoothing distributions $\pi_t(\mathrm{d}x_{0:t})$ for all $t \in \{0, \dots, T\}$, which also recover the filtering distributions $\pi_t(\mathrm{d}x_t)$ for all $t \in \{0, \dots, T\}$ by marginalisation. While we have made this implicit, Algorithm 6 in fact directly provides an approximation of the smoothing distributions $\pi_T^N(\mathrm{d}x_{0:T}) = \sum_{n=1}^N W_T^n \delta_{X_{0:T}^{(n)}}(\mathrm{d}x_{0:T})$ as well. This was first remarked by Kitagawa (1994), who also proposed an alternative algorithm for state-space models, known as the two-filter smoother, for computing the smoothing distributions π_T. In this section, we discuss a more general approach, originally due to Godsill et al. (2004), which can be used to regenerate the smoothing distributions π_T from the filtering distributions π_t for $t \in \{0, \dots, T\}$, known as the backward sampling algorithm.

The method relies on the following proposition, which is a direct consequence of the recursive model structure (3.33).

Proposition 3.34 (Backward simulation)**.** *Suppose that the model $(\pi_t)_{t=0}^T$ is given by (3.33), and assume that all the π_t have a density with respect to a common product measure, then, we have*

$$\mathrm{d}\pi_T(x_{0:t} \mid x_{t+1:T}) \propto \frac{\pi_T(x_{0:T})}{\pi_t(x_{0:t})} \mathrm{d}\pi_t(x_{0:t}). \tag{3.43}$$

Proof. We have

$$\begin{aligned}\pi_T(x_{0:t} \mid x_{t+1:T}) &\propto \pi_T(x_{0:t}, x_{t+1:T}) \\ &= \frac{\pi_T(x_{0:T})}{\pi_t(x_{0:t})} \pi_t(x_{0:t}).\end{aligned}$$

□

Namely, suppose that we have an approximation $\sum_{n=1}^N W_t^n \delta_{X_{0:t}^n}(\mathrm{d}x_{0:t})$ of the filtering distribution $\pi_t(\mathrm{d}x_{0:t})$ for $t \in \{0, \dots, T\}$, then we can form an approximation of the conditional smoothing distribution $\pi_T(\mathrm{d}x_{0:t-1} \mid x_{t:T})$ as

$$\pi_T^N(\mathrm{d}x_{0:t} \mid x_{t+1:T}) = \sum_{n=1}^N \widetilde{W}_t^n \delta_{X_{0:t}^n}(\mathrm{d}x_{0:t}), \tag{3.44}$$

where the weights $\widetilde{W}_t^n$ are given by

$$\begin{aligned} \widetilde{w}_t^n &= \frac{\pi_T(X_{0:T}^n)}{\pi_t(X_{0:t}^n)} W_t^n, \\ \widetilde{W}_t^n &= \frac{\widetilde{w}_t^n}{\sum_{m=1}^N \widetilde{w}_t^m}, \end{aligned} \tag{3.45}$$

noting that the ratio $W_{t:T}^n$ does not depend on the normalising constants of the distributions π_t and π_T. This procedure, known as backward simulation, is summarised in Algorithm 7.

Algorithm 7: Backward simulation

```
input  : Feynman–Kac model $(\pi_t)_{t=0}^T$ given by (3.33), filtering
         approximations coming from Algorithm 6 $(\pi_t^N)_{t=0}^T$, number of
         smoothing samples $M$.
output: Approximate distribution $\pi_T^M(\mathrm{d}x_{0:T}) = \sum_{m=1}^M W_{0:T}^m \delta_{Z_{0:T}^m}(\mathrm{d}x_{0:T})$.
1 ▷ All operations involving the index $m$ need to be done for $m = 1,\dots,M$,
    and all operations involving the index $n$ need to be done for $n = 1,\dots,N$
2 Sample $Z_{T:T}^m \sim \pi_T(\mathrm{d}x_T)$
3 for $t \in \{T-1,\dots,0\}$ do
4     Set $\widetilde{w}_t^{m,n} \leftarrow \frac{\pi_T(X_{0:t}^{(n)}, Z_{t+1:T}^m)}{\pi_t(X_{0:t}^{(n)})} W_t^m$
5     Set $\widetilde{W}_t^{m,n} \leftarrow \frac{\widetilde{w}_t^{m,n}}{\sum_{k=1}^N \widetilde{w}_t^{m,k}}$
6     Sample $B_t^m \sim \mathrm{Multinomial}(\widetilde{W}_t^{m,1:N})$
7     Set $Z_{t:T}^m \leftarrow [X_t^{B_t^m}, Z_{t+1:T}^m]$
```

For N filtering and M smoothing particles, respectively, Algorithm 7 presents a complexity of $\mathcal{O}(TMN)$, which is prohibitive for large N. However, it is possible to reduce the complexity of the algorithm to $\mathcal{O}(TN)$ by using either rejection sampling, or Markov moves. For a detailed discussion of these methods, as well as general consistency results for backward sampling algorithms, we refer to Dau and Chopin (2023).

Example 3.35. *In this section, we have described the backward simulation algorithm for a general, non-Markovian model $(\pi_t)_{t=0}^T$. When the model is Markovian however, i.e., when*

$$\mathrm{d}\pi_t(x_{0:t}) \propto M_t(\mathrm{d}x_t \mid x_{t-1}) G_t(x_{t-1:t}) \mathrm{d}\pi_{t-1}(x_{0:t-1}) \tag{3.46}$$

following (3.8)*, then the expression* (3.43) *simplifies to*

$$\begin{aligned}\pi_T(\mathrm{d}x_{0:t} \mid x_{t+1:T}) &\propto \frac{\pi_T(x_{0:T})}{\pi_t(x_{0:t})} \mathrm{d}\pi_t(x_{0:t}),\\ &\propto \frac{\pi_{t+1}(x_{0:t+1})}{\pi_t(x_{0:t})} \mathrm{d}\pi_t(x_{0:t}),\\ &\propto M_{t+1}(x_{t+1} \mid x_t) G_{t+1}(x_{t:t+1}) \mathrm{d}\pi_t(x_{0:t}),\end{aligned} \tag{3.47}$$

when seen as a function of x_{t+1} and x_t. This further recovers the backwards simulation weights $M_{t+1}(x_{t+1} \mid x_t)$ when G_{t+1} does not depend on x_t, for example, when the model is a state-space model (3.2).

3.4.4 Model identification

Similar to when we used Gaussian-based methods in Section 3.3.3, $\pi_T(\mathrm{d}x_{0:T};\theta)$ may depend on a parameter that one may wish to infer. Thankfully, the SIR of Algorithm 6, with or without backward sampling 7 provides consistent estimators of the smoothing distribution π_T as well as unbiased estimators of the normalising constant $L_T(\theta)$ of $\pi_T(\mathrm{d}x_{0:T};\theta)$.

Perhaps the most natural way to find an approximation to the maximum likelihood estimator of θ consists in computing the gradient of the estimator L_T^N with respect to θ (for example using automatic differentiation), and then taking a gradient ascent step with respect to θ, hopefully converging to a maximum of the normalising constant $L_T(\theta)$. While this method is attractive, it is in fact incorrect: due to the use of resampling steps in the SIR algorithm, the normalising constant estimator L_T^N is not differentiable with respect to θ (it is in fact non-continuous), and therefore, one cannot differentiate *through* the algorithm. This issue was, to the best of our knowledge, first pointed out by Malik and Pitt (2011a,b), who then suggested to replace the resampling step with a stochastic interpolation, making the algorithm piecewise differentiable with respect to θ, provided that the state is one-dimensional. However, this method is not applicable in general, and in particular, when the state is multi-dimensional, and one must then resort to other methods.

Remark 3.36. *In some sense, Publication I is a direct response to this issue, as it provides a differentiable particle filter algorithm, which can be used to obtain differentiable estimators of the normalising constant L_T with respect to θ. It can also be thought of as a generalisation of the interpolation method of Malik* *and Pitt (2011a,b) to multi-dimensional state-spaces as the technique it relies on, entropy-regularised optimal transport (Cuturi, 2013), can too be seen as an interpolation method.*

While, L_T^N, and therefore the more numerically amenable $\log L_T^N$ is not differentiable, its expectation $L_T = \mathbb{E}[L_T^N]$ is. As a consequence, it is natural to try and obtain approximations of the gradient of $\log L_T$ with respect to θ directly rather than as a by-product of the SIR algorithm. This can be done by using the following identity, called Fisher's identity (Poyiadjis et al., 2011; Kantas et al., 2015).

Proposition 3.37 (Fisher's identity). *Suppose that the model $(\pi_t)_{t=0}^T$ given by (3.33) corresponds to the un-normalised density*

$$\pi_T(x_{0:T};\theta) =: \frac{1}{Z_T(\theta)} \Gamma_{0:T}(x_{0:T};\theta), \tag{3.48}$$

where $L_T(\theta) = \int \Gamma_{0:T}(x_{0:T};\theta) \mathrm{d}x_{0:T}$ is the normalising constant. Then, we have

$$\frac{\partial}{\partial\theta} \log L_T = \mathbb{E}_{\pi_T}\left[\frac{\partial}{\partial\theta} \log \Gamma_{0:T}(x_{0:T};\theta)\right]. \tag{3.49}$$

In other terms, the gradient of the log-normalising constant L_T with respect to θ can be obtained by taking the expectation of the gradient of the log-density of the un-normalised density of the smoothing distribution π_T with respect to θ.

Example 3.38. *Let*

$$\pi_T(\mathrm{d}x_{0:T};\theta) \propto p_0(\mathrm{d}x_0) p_0(y_0 \mid x_0;\theta) \prod_{t=0}^{T-1} p_{t+1}(\mathrm{d}x_{t+1} \mid x_t) p_{t+1}(y_{t+1} \mid x_{t+1};\theta) \tag{3.50}$$

be a state-space model of the form (3.52) depending on a parameter θ only through the observation model $p_t(y_t \mid x_t;\theta)$. Then, we have

$$\frac{\partial}{\partial\theta} \log L_T(\theta) = \mathbb{E}_{\pi_T}\left[\sum_{t=0}^{T} \frac{\partial}{\partial\theta} \log p_t(y_t \mid x_t;\theta)\right]. \tag{3.51}$$

As a consequence, one can obtain a consistent estimator of the gradient of $\log L_T$ with respect to θ by computing (3.49) under samples from the SIR of Algorithm 6, with or without backward sampling (Algorithm 7). This estimator may then be used in an approximate gradient ascent algorithm to maximise the normalising constant L_T with respect to θ. Related to this approach, one can also implement stochastic expectation-maximisation (EM) algorithms (Cappé and Moulines, 2009) for the model (3.33) by using the same identity. Finally, we note that other, online, methods for maximising the normalising constant L_T with respect to θ are available, whereby one does not need to compute the smoothing distribution π_T but instead, only the filtering distributions π_t for $t \in \{0,\ldots,T\}$, requiring less memory and computation when Markovianity is assumed. We refer to Kantas et al. (2015) for a detailed discussion of these methods.

3.4.5 The choice of the Feynman–Kac representation

In the previous sections, we have described how to sample from the filtering and smoothing distributions π_t and π_T for a general model $(\pi_t)_{t=0}^T$ given by (3.33). The question then arises, when one is given, for instance, a state-space model of the form

$$\mathrm{d}\pi_T(x_{0:T}) \propto p_0(\mathrm{d}x_0) \prod_{s=0}^{T} p_{s+1}(\mathrm{d}x_{s+1} \mid x_s) \prod_{s=0}^{T} p_s(y_s \mid x_s), \tag{3.52}$$

of the best sequence of distributions π_t to use for the SIR of Algorithm 6 to sample from π_T. Said otherwise, we want to find an arbitrary sequence of distributions $(\pi_t)_{t=0}^T$ such that π_T is given by (3.52) and the degeneracy problem of the SIR algorithm is as mild as possible. In this section, we focus on the case of state-space models of the form (3.52) and discuss two possible methods for designing such a sequence of distributions $(\pi_t)_{t=0}^T$.

Local approximations. The first method consists in considering local approximations to the model (3.52), i.e., approximations to the transition and observation models for $t \in \{0, \dots, T\}$. The intuition behind this method is to get an importance sampling estimator for the normalising constant L_t with minimal variance following Proposition 2.11 for a choice of integrand $f \equiv 1$. In other terms, we wish, as discussed in Example 3.12, for a given x_{t-1}, to sample from $p(\mathrm{d}x_t \mid x_{t-1}, y_t) \propto p_t(\mathrm{d}x_t \mid x_{t-1}) p_t(y_t \mid x_t)$. This proposal is usually described as the "locally optimal proposal" (Doucet and Johansen, 2011) in the sense that it minimises the variance of the importance sampling estimator for the normalising constant $p(y_t \mid y_{0:t-1})$.

However, it is rarely possible to sample from $p(\mathrm{d}x_t \mid x_{t-1}, y_t)$, and instead, one needs to resort to approximations of this distribution. A popular choice, used for example by Van Der Merwe et al. (2000) is to use the Gaussian approximations of Section 3.3, i.e., to approximate $p(\mathrm{d}x_t \mid x_{t-1}, y_t)$ by a Gaussian distribution $\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1}) \sim \mathcal{N}(x_t; \mu_t(x_{t-1}, y_t), \Sigma_t(x_{t-1}, y_t))$ using any of the methods described in Section 3.3.

Once an approximation has been chosen, the SIR of Algorithm 6 can be applied to sample from the sequence of distributions $(\pi_t)_{t=0}^T$ given by the following recursive decomposition

$$\mathrm{d}\pi_t(x_{0:t}) \propto \widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1}) \underbrace{\frac{p_t(\mathrm{d}x_t \mid x_{t-1}) p_t(y_t \mid x_t)}{\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1})}}_{\widetilde{G}_t(x_{t-1:t})} \mathrm{d}\pi_{t-1}(x_{0:t-1}), \tag{3.53}$$

which is well defined as soon as the Radon–Nikodym derivative $\frac{p_t(\mathrm{d}x_t \mid x_{t-1})}{M_t(\mathrm{d}x_t \mid x_{t-1})}$ is well defined. Such approximations are however *ad hoc*, and also do not fully alleviate the degeneracy problem of the SIR algorithm, as the dimension of the space X^t is still increasing with t making these locally optimal proposals still exponentially inefficient as t increases.

Global approximations. Let us assume for a moment that we have access to a sampler $\widetilde{M}_t(\mathrm{d}x_t \mid x_{0:t-1}) = \pi_T(\mathrm{d}x_t \mid x_{0:t-1})$, for all t that is, such that sampling from $\pi_T(\mathrm{d}x_{0:T})$ is possible by simple recursion. Further define the sequence of target distributions as $\pi_T(\mathrm{d}x_{0:t})$, for $t \in \{0,\dots,T\}$, corresponding to the partial smoothing distributions of the model (3.52). Then the decomposition (3.33) is satisfied for the choice of potential functions $G_t \equiv 1$ and the resulting weights W_t^n appearing in Algorithm 6 are uniformly $1/N$ so that no resampling is required and the variance of the normalising constant estimator L_t^N is 0.[1] This idea, known in the literature as *twisting* (Whiteley and Lee, 2014), is the basis behind constructing global approximations to the model (3.33) by means of parameterisations of the *globally* optimal proposal distributions.

Example 3.39. *When π_T is given by (3.52), then the* globally *optimal proposal distribution $\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1})$ is given by*

$$\begin{aligned}\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1}) &\propto p(\mathrm{d}x_t \mid x_{t-1}) p(y_{t:T} \mid x_t),\\ &\propto p(\mathrm{d}x_t \mid x_{t-1}) p(y_t \mid x_t) \int \prod_{s=t+1}^{T} p(y_s \mid x_s) p(\mathrm{d}x_s \mid x_{s-1}),\end{aligned} \tag{3.54}$$

corresponding to a Markovian proposal distribution.

When the model is a state-space model, Example 3.39 suggests that we could find the optimal proposal distribution $\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1})$ as a parameterisation of a family of Markovian kernels:

$$\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1}; \phi), \tag{3.55}$$

for a parameter $\phi \in \Phi$. The goal of the statistician is then to find a tractable optimisation problem for finding the parameter ϕ such that the approximation $\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1}; \phi)$ is as close as possible to the true conditional smoothing distribution $\pi_T(\mathrm{d}x_t \mid x_{t-1})$.

A natural way to do so is then to try and minimise the variance of the importance weights *after the fact*. This is the approach taken by the iterated auxiliary particle filter (IAPF) of Guarniero et al. (2017) and the controlled SMC algorithm of Heng et al. (2020), which both consider different update rules for the parameter ϕ, both based on the same idea of making the normalising constant as deterministic as possible.

Remark 3.40. *In Publication V, we use a locally optimal proposal for an auxiliary model to improve the quality of the sampling method proposed therein, while, in Publication VII, we use these too, but also extend it to a twisted approximation (that, in our case, we can compute in closed form) of the model to obtain even better results.*

[1]Note that in the local approximation case, the variance was minimised for a *given* x_{t-1} so that under samples from π_{t-1}, the variance of the normalising constant estimator L_t^N would not be 0.

3.4.6 Variational SMC

In this section, we describe a different approach to the problem of finding a good sequence of proposal distributions M_t for a given model $(\pi_t)_{t=0}^T$. As noted in Section 3.4.4, the approximate normalising constant L_T^N is an unbiased estimator of the normalising constant L_T of the model (3.33). Using Jensen's inequality, we can then obtain a lower bound on the log-normalising constant $\log L_T$ as $\mathbb{E}[\log L_T^N]$, which can be computed using the SIR of Algorithm 6. This lower bound (see, e.g., Maddison et al., 2017) is a form of *evidence lower bound* (ELBO), described in Section 2.3.3, which presents several properties that make it attractive for inference listed in the following remark.

Remark 3.41 (Monte Carlo objectives)**.** *The quantity* $\mathbb{E}[\log L_T^N]$ *is a lower bound on the log-normalising constant* $\log L_T$ *and the lower bound* $\mathbb{E}[\log L_T^N]$ *converges to the log-normalising constant* $\log L_T$ *as* $N \to \infty$*. In other terms,* $\mathbb{E}[\log L_T^N]$ *is an ELBO, and is consistent in the limit* $N \to \infty$*.*

An intuitive way to relate this lower bound to the approaches of the previous section is to note that the lower bound $\mathbb{E}[\log L_T^N]$ is tight when L_T^N is deterministic, that is, when and only when the variance of the importance weights W_T^n is 0. Maximising the lower bound $\mathbb{E}[\log L_T^N]$ can then be seen as a different way to find a sequence of *globally optimal* importance distributions M_t for the model (3.33). This idea is the basis behind the variational SMC (VSMC) algorithms of Naesseth et al. (2018); Maddison et al. (2017); Le et al. (2018), where the expectation $\mathbb{E}[\log L_T^N]$ is approximated using M independent realisations from the SIR of Algorithm 6, and the resulting approximation is differentiated with respect to the parameters of the proposal distributions $M_t(\cdot;\phi)$ by means of backpropagation. The VSMC algorithm is summarised in Algorithm 8.

Algorithm 8: Variational SMC (VSMC)

```
input : Feynman–Kac model $(\pi_t)_{t=0}^T$ given by (3.33), initial distribution,
        number of particles $N$, variational family parameter $\phi$, number of
        iterations $K$, number of independent SIR algorithms $J$,
        optimisation step-sizes $\alpha_k$.
output: Optimal proposal distribution $M_t(\mathrm{d}x_t \mid x_{t-1};\phi^*)$ for $t \in \{0,\dots,T\}$.
1 Initialise $\phi \leftarrow \phi_0$
2 for $k \in \{1,\dots,K\}$ do
3     Sample $L_T^{N,j}(\phi)$, $j = 1,\dots,J$ using Algorithm 6
4     Compute $\nabla_\phi \frac{1}{J}\sum_{j=1}^J \log L_T^{N,j}(\phi)$ using backpropagation
5     Update $\phi \leftarrow \phi + \alpha_k \nabla_\phi \frac{1}{J}\sum_{m=1}^J \log L_T^{N,j}(\phi)$
```

While this algorithm is attractive, a crucial limitation is that it requires the ability to backpropagate through the SIR of Algorithm 6, which as we mentioned in Section 3.4.4, is not possible due to the use of resampling steps. Following

Example 2.26 of Section 2.3.3, it is nonetheless possible to derive an estimator of the gradient by means of stacking several score function estimators throughout the differentiation procedure. However, doing so empirically results in very high variance estimators (Maddison et al., 2017; Le et al., 2018; Naesseth et al., 2018), and the resulting algorithm is not practical. This observation has prompted proponents of VSMC to simply propagate the gradients through all the steps of the algorithm, to the exception of the resampling step, which is simply ignored, resulting in a non-consistent estimator of the gradient for the lower bound $\mathbb{E}[\log L_T^N]$. While this approach is not theoretically justified, it has been shown to work well in practice.

Remark 3.42. *Maximising this lower bound is a central question of Publication I, and is one of the motivations for the development of a differentiable particle filter algorithm. For a recent review of the topic, including a discussion of Publication I, we refer to Chen and Li (2023).*

4. Markov chain Monte Carlo methods

In Chapter 2, we have introduced the general problem of inference in Bayesian statistical models, and we have discussed how these methods could be applied to perform efficient inference in Markovian models, such as state-space models, in Chapter 3. These methods were either based on a form of variational inference (such as linearisation in state-space models), or on variations on importance sampling (such as particle filtering). An issue shared by these two methods is that they both consist of *global* approximations to the posterior distribution, in the sense that they are based on approximating the posterior distribution as a whole, which is often an arduous task when the dimension of the space is large. An easier question, that we try to answer in this chapter, is the following one:

If X_k is a sample from π, how can we generate another new sample X_{k+1} from π?

In other terms, how can we form a chain of samples $X_0, X_1, \dots$ such that $X_k \sim \pi$ for all k?

In this chapter, we review the general framework of Markov chain Monte Carlo (MCMC) methods (Brooks et al., 2011), which are a class of algorithms that aim at solving this problem. Special attention will be given to the class of *auxiliary samplers*, following the denomination of Higdon (1998), which are particularly well suited to introducing statistically relevant, albeit computationally simple, information in the design of the chain of samples. We will then discuss a particular class of MCMC methods, called particle MCMC methods (Andrieu et al., 2010), which are well suited to the problem of inference in state-space models and their link with the auxiliary sampler framework.

This chapter is central to Publications V and VII, which are interested in designing efficient MCMC methods for state-space models (see Chapter 3).

4.1 Markov chain Monte Carlo

In this section, we assume that we are given a probability distribution π that has a density (also denoted π) with respect to a reference measure, i.e., $\pi(\mathrm{d}x) = \pi(x)\mathrm{d}x$

which we will assume to be the Lebesgue measure, the counting measure, or a combination of these. Furthermore, we assume that $\pi(x) \propto f(x)$ is known up to a constant of proportionality, where f is a function on X.

Markov chain Monte Carlo (MCMC) methods are a simulation based approach to the problem of sampling from a given probability distribution $\pi(\mathrm{d}x)$ that, at heart, are based on the idea of implementing easy-to-sample Markov chains (see, e.g., Douc et al., 2018) which verify LLNs of the form

$$\frac{1}{T}\sum_{t=1}^{T} f(X_t) \to \int f(x)\pi(\mathrm{d}x), \tag{4.1}$$

almost surely as $T \to \infty$ and for all regular enough functions f.

4.1.1 Computing expectations and ergodic averages of stationary Markov chains

In order to build such Markov chains, the key idea is to build a Markov kernel on X which is invariant with respect to π.

Definition 4.1 (Invariance)**.** *A Markov kernel K on* X *is said to be invariant with respect to π if*

$$\pi(\mathrm{d}y) = \int K(\mathrm{d}y \mid x)\pi(\mathrm{d}x). \tag{4.2}$$

In other terms, if $X \sim \pi$, and $Y \sim K(\cdot \mid X)$, then $Y \sim \pi$. When the kernel K is invariant with respect to π, we also say that K is π-invariant, or that the resulting chain $X_1, X_2, \ldots, X_T$ is π-invariant. While stationarity is not necessary to build Markov chains verifying a LLN, and one may consider non-homogeneous chains that *eventually* stabilise (see, e.g., Andrieu and Thoms, 2008, for adaptive methods), it forms the basis of classical MCMC methods (Meyn and Tweedie, 2009, Chap. 10), and we will therefore focus on these in this chapter.

A sufficient condition for a Markov kernel K to be π-invariant is that of reversibility.

Definition 4.2 (Reversibility)**.** *A Markov kernel K on* X *is said to be reversible with respect to π if*

$$\pi(\mathrm{d}x)K(\mathrm{d}y \mid x) = \pi(\mathrm{d}y)K(\mathrm{d}x \mid y) \tag{4.3}$$

for all $x, y \in$ X*.*

This condition, which is also known as the *detailed balance* condition, implies the invariance with respect to π for K, as can be seen by integrating both sides of (4.3) with respect to x. Verifying reversibility is often a much simpler task than verifying invariance (Andrieu et al., 2020), and most MCMC methods have historically been designed to verify reversibility[1].

[1]Note however that, in recent years, a number of MCMC methods have been proposed that do not verify reversibility, but still verify invariance (Chen et al., 1999; Diaconis et al., 2000; Bierkens et al., 2019; Bouchard-Côté et al., 2018; Sherlock and Thiery, 2022), often outperforming reversible alternatives at the cost of a more complex simulation procedure.

Under some technical conditions on the Markov kernel K and the target distribution π, which are hard to prove in practice, it is possible to show that LLNs of the form (4.1) hold for all regular enough (including bounded) functions φ.

Theorem 4.3 (Birkhoff's ergodic theorem)**.** *Let K be a π-invariant Markov kernel on $\mathbb{X}$ and let $X_0, X_1, \dots$ be a Markov chain with transition kernel K and initial distribution π_0, with $\pi_0 \ll \pi$. Then, for all regular enough functions φ (which include bounded functions) on $\mathbb{X}$, and under some technical conditions, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\varphi(X_t) \to \int \varphi(x)\pi(\mathrm{d}x), \tag{4.4}$$

π-a.s. as $T \to \infty$.

The general form of this theorem is given, for example in Meyn and Tweedie (2009, Chap. 17), and is a generalisation of the strong law of large numbers (LLN, see Chapter 2) to dynamical systems.

Under similar conditions as for Theorem 4.3, the Markov chain $X_0, X_1, \dots$ converges to π in total variation (see, e.g., Meyn and Tweedie, 2009, Chap. 13).

Theorem 4.4 (Convergence in total variation)**.** *Let K be a π-invariant Markov kernel on $\mathbb{X}$ and let $X_0, X_1, \dots$ be a Markov chain with transition kernel K and initial distribution δ_x, for some $x \in \mathbb{X}$. Under technical conditions, we have*

$$\|\mathbb{P}(X_t \in \cdot) - \pi(\cdot)\|_{\mathrm{TV}} \to 0 \tag{4.5}$$

as $t \to \infty$.

Intuitively, this means that the samples from the chain $X_0, X_1, \dots$ eventually are indistinguishable from samples from π.

Additionally, under stronger conditions relating to the existence of higher order moments, it is possible to prove that the Markov chain $X_0, X_1, \dots$ follows a *central limit theorem* (CLT) at stationarity (Meyn and Tweedie, 2009, Chap. 17).

Theorem 4.5 (Central limit theorem)**.** *Let K be a π-invariant Markov kernel on $\mathbb{X}$ and let $X_0, X_1, \dots$ be a Markov chain with transition kernel K and initial distribution π. Then, for all regular enough functions φ from $\mathbb{X}$ to $\mathbb{R}$, and under some technical conditions, we have*

$$\sqrt{T}\left(\frac{1}{T}\sum_{t=1}^{T}\varphi(X_t) - \int \varphi(x)\pi(\mathrm{d}x)\right) \to \mathcal{N}(0, \sigma^2 + 2\sum_{k=1}^{\infty}\mathbb{C}[\varphi(X_0), \varphi(X_k)]), \tag{4.6}$$

in distribution as $T \to \infty$ and where $\sigma^2 = \mathbb{C}_\pi[\varphi(X_0)]$.

In other terms, the asymptotic variance of the CLT is given by the variance of the integrand under π and the sum of all the covariances between the integrand and its shifted self, known as the *auto-covariance function* of the Markov chain

$\varphi(X_k)$. This quantity is often replaced by the *integrated autocorrelation time* (IAT) of the Markov chain $\varphi(X_k)$, defined as

$$\tau_{\text{int}}(\varphi) = 1 + 2\sum_{k=1}^{\infty} \frac{\mathbb{C}[\varphi(X_0), \varphi(X_k)]}{\mathbb{C}_\pi[\varphi(X_0)]}, \tag{4.7}$$

which serves as a measure of the efficiency of the Markov chain $X_0, X_1, \dots$ to compute the expectation $\int \varphi(x)\pi(\mathrm{d}x)$ compared to a perfect Monte Carlo method (then corresponding to an IAT equal to 1), and is often used in practice to compare the efficiency of different MCMC methods (Vehtari et al., 2021).

Indeed, the smaller the IAT, the more efficient the chain in terms of asymptotic variance in the CLT. Consequently, the goal of the statistician when designing a new MCMC method is to design a Markov kernel K that has a small IAT, and that is easy to simulate. These two goals are often at odds with each other, and the design of MCMC methods is often a trade-off between these two principles: statistical quality and computational efficiency.

4.1.2 Metropolis–Hastings methods

Perhaps the most important class of MCMC methods are the algorithms based, in one form or another, on the Metropolis–Hastings [2] procedure (Metropolis et al., 1953; Hastings, 1970). These algorithms are based on the idea of constructing a reversible Markov kernel as a composition of two steps: a proposal step, and an acceptance step.

Formally, let $Q(\mathrm{d}y \mid x)$ be a Markov kernel on X, which is not necessarily reversible with respect to π, and let $\alpha(x, y)$ be a function on $\mathsf{X} \times \mathsf{X}$. Metropolis–Hastings kernels are then defined as the Markov kernels K on X of the form

$$K(\mathrm{d}y \mid x) = Q(\mathrm{d}y \mid x)\alpha(x, y) + \delta_x(\mathrm{d}y)\left(1 - \int \alpha(x, z)Q(\mathrm{d}z \mid x)\right). \tag{4.8}$$

In other terms, given the current state of the chain X, the sampling from $K(\cdot \mid X)$ is obtained by first sampling a proposal $Y \sim Q(\cdot \mid X)$, and then accepting it with probability $\alpha(X, Y)$, or rejecting it and staying at X with the complementary probability. The function α is called the *acceptance probability* of the kernel K. For the kernel K to be reversible with respect to π, it is sufficient that α verifies

$$\alpha(x, y)\pi(x)Q(y \mid x) = \alpha(y, x)\pi(y)Q(x \mid y) \tag{4.9}$$

for all $x, y \in \mathsf{X}$.

[2]The algorithm has recently been called Metropolis–Rosenbluth–Teller–Hastings in several works, including some of ours, to credit Arianna Rosenbluth, Marshall Rosenbluth, Edward Teller, and Augusta Teller. Their important scientific contribution to the method (Rosenbluth, 2003) went unnoticed due to alphabetical ordering of authors. Here we elect instead to keep the usual denomination, but will denote their specific implementation of the acceptance step, defined hereafter, as the Rosenbluth–Teller acceptance step.

The most well known choice for α is *Rosenbluth–Teller's acceptance ratio* (Hastings, 1970)

$$\alpha(x,y) = \min\left\{1, \frac{\pi(y)Q(x \mid y)}{\pi(x)Q(y \mid x)}\right\}, \tag{4.10}$$

however, it is not the only possible choice for α. Another popular choice is *Barker's acceptance probability* (Barker, 1965)

$$\alpha(x,y) = \frac{\pi(y)Q(x \mid y)}{\pi(x)Q(y \mid x) + \pi(y)Q(x \mid y)}. \tag{4.11}$$

In both cases, the acceptance probability is unchanged if we multiply $\pi(x)$ by a constant $\lambda\pi(x)$ for $\lambda > 0$, so that the method can be used even if π is only known up to a constant of proportionality. In practice, when it can be used, Rosenbluth–Teller's acceptance probability is preferred to Barker's acceptance probability, as it is more efficient, in the sense that it leads to lower variance estimators of the expectations of interest (Łatuszyński and Roberts, 2013). However, Barker's acceptance probability is more flexible as it presents a form of symmetry between x and y[3], which is absent from Rosenbluth–Teller's acceptance probability. We will make use of this flexibility[4] in Section 4.3, see also Zanella (2020); Vats et al. (2021); Livingstone and Zanella (2022) for recent applications of Barker's acceptance probability, and related ideas.

4.1.3 Independent Metropolis–Hastings

A special case of Metropolis–Hastings kernels is that of *independent Metropolis–Hastings* (IMH) kernels, which are kernels of the form

$$K(\mathrm{d}y \mid x) = Q(\mathrm{d}y)\alpha(x,y) + \delta_x(\mathrm{d}y)\left(1 - \int Q(\mathrm{d}z)\alpha(x,z)\right), \tag{4.12}$$

where $Q(\mathrm{d}y \mid x) \equiv Q(\mathrm{d}y)$ is a probability distribution on X that does not depend on x. A typical choice for Q, when $\pi(\mathrm{d}x) \propto f(x)\pi_0(\mathrm{d}x)$ for some prior probability distribution π_0 on X and some likelihood function f on X, is to take $Q(\mathrm{d}y) = \pi_0(\mathrm{d}y)$, in which case Rosenbluth–Teller's acceptance probability (4.10) simplifies to

$$\alpha(x,y) = \min\left\{1, \frac{f(y)}{f(x)}\right\}, \tag{4.13}$$

and Barker's acceptance probability (4.11) simplifies to

$$\alpha(x,y) = \frac{f(y)}{f(x) + f(y)}. \tag{4.14}$$

IMH is usually considered a poor choice for MCMC as it does not make use of the information brought by the previous state X_k to propose the next state X_{k+1}, which was the motivation for the development of MCMC methods in the

[3]Owing to the fact that then $\alpha(x,y) = 1 - \alpha(y,x)$.

[4]It is not strictly necessary (and in Publication VII we make use of Rosenbluth–Teller's acceptance probability) but often simplifies arguments.

first place. In fact, it can often be replaced fully by importance sampling, with lower variance estimators of the expectations of interest (see, e.g., Liu, 1996, for a comparison). A notable exception is given by its extension to Markovian models, which is the topic of Section 4.3.

4.1.4 Random walk Metropolis–Hastings

A popular choice for Q is that of a *random walk* kernel, i.e., a kernel of the form $Q(y \mid x) = q(y - x)$ for some probability distribution q on $\mathbb{X}$ often taken to be a centered Gaussian distribution with a given covariance matrix Σ when $\mathbb{X} = \mathbb{R}^D$. This choice was in fact the original choice of Metropolis–Hastings (Metropolis et al., 1953), and is still widely used in practice. In this case, the resulting kernel is called a *random walk Metropolis–Hastings* (RWM) kernel. When α is chosen to be Rosenbluth–Teller's acceptance probability (4.10), the resulting acceptance probability simplifies to

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y) q(x - y)}{\pi(x) q(y - x)}\right\} = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}, \tag{4.15}$$

so that the acceptance probability does not depend on the density of the proposal kernel q and similarly for Barker's acceptance probability (4.11).

The choice of the proposal scale Σ is crucial to the performance of the algorithm and has been the subject of a large body of research (Roberts et al., 1997; Haario et al., 2006; Andrieu and Thoms, 2008; Yang et al., 2020) and is still an active area of research. When $\pi \propto \prod_{d=1}^{D} f(x_d)$ is a separable probability density on $\mathbb{R}^D$ and q is a centered Gaussian distribution with covariance matrix $\delta^2 I$, the optimal choice for δ is given by $\delta^2 = 2.38^2/D$, in the sense that this choice maximises the convergence speed[5] of the chain to the target distribution (Roberts et al., 1997) as $D \to \infty$. This in turn corresponds to an asymptotic average acceptance rate $\alpha^* = \int \alpha(x, y) \pi(\mathrm{d}x) Q(\mathrm{d}y \mid x)$ of approximately 23.4%, often used as a rule of thumb for the choice of δ. Some methods also exist to choose full covariance matrices Σ adaptively, where the purpose is to choose it so that the acceptance rate of the chain is close to a given target acceptance rate α^*, but also such that the scaled target distribution $\pi^{\Sigma}(\mathrm{d}x)$ corresponding to the law of $\Sigma^{-1/2} X$, for $X \sim \pi$, is approximately isotropic. In other terms, to choose Σ such that the density of $\pi^{\Sigma}(x)$ can approximately be written as being proportional to $f^{\Sigma}(\|x\|)$ for some function f^{Σ} (see, e.g., Haario et al., 2006; Andrieu and Thoms, 2008). This corresponds to choosing a transformation of the space $\mathbb{X}$ under which symmetric proposals are close to being optimal, and, when the target is Gaussian, can heuristically be understood as scaling the space by the covariance matrix Σ of the target distribution π.

While RWM kernels make use of some information brought by the previous state X_k to propose the next state X_{k+1}, they do not take into account the

[5]More specifically, when $\delta^2 \propto D^{-1}$, the first component of the chain converges to a limiting Langevin diffusion for f, as $D \to \infty$, the speed of which is maximised for this choice of the proportionality constant.

geometry of the target distribution π around X_k. In particular, this means that they are as likely to propose a sample Y that has (locally) low density under π as they are to propose a sample Y that has high density under π. This results in an ineffective exploration of the space X as the resulting Y is likely to be rejected by the acceptance step, reflecting the low value for its optimal acceptance probability α^*.

4.1.5 Langevin-based Markov chain Monte Carlo

In order to address the issue of the poor exploration of the space X by RWM kernels, a number of methods have been proposed that make use of gradient information from the target distribution π. The most popular of these methods are based on using Langevin dynamics (Besag, 1994) associated to the density π:

$$\mathrm{d}X_t = \frac{1}{2}\nabla \log \pi(X_t)\mathrm{d}t + \mathrm{d}W_t, \tag{4.16}$$

where W_t is a standard Brownian motion on $\mathbb{R}^D$. Dynamics of (4.16) are invariant with respect to π, i.e., if $X_0 \sim \pi$, then $X_t \sim \pi$ for all $t \geq 0$.

Of course the process X_t cannot in general be simulated exactly, and discretisations, for example Euler–Maruyama (see, e.g., Särkkä and Solin, 2019, Chap. 8), of (4.16) are used instead, giving rise to Markov kernels $Q^\delta(\mathrm{d}y \mid x)$ that are approximations of the true discretised dynamics (4.16). These can then be used *as is* to build Metropolis–Hastings kernels, giving rise to the so-called class of unadjusted Langevin algorithms (ULA, see, e.g., Roberts and Tweedie, 1996). These kernels are not stationary with respect to π but rather to an approximation of π^δ that depends on the discretisation step δ and on the choice of discretisation scheme. Under some regularity conditions on π, it can however be shown that expectations computed with respect to π^δ are good approximations of expectations computed with respect to π (see, e.g., Durmus and Moulines, 2019). This approach can be corrected by adding an MH acceptance step to the Q^δ, giving rise to the so-called Metropolis-adjusted Langevin algorithm (MALA, Besag, 1994), which is often combined with an Euler–Maruyama discretisation of (4.16), giving

$$K^\delta(\mathrm{d}y \mid x) = Q^\delta(\mathrm{d}y \mid x)\alpha(x,y) + \delta_x(\mathrm{d}y)\left(1 - \int Q^\delta(\mathrm{d}z \mid x)\alpha(x,z)\right), \tag{4.17}$$

for $Q^\delta(\mathrm{d}y \mid x) \sim \mathcal{N}\left(x + \frac{\delta}{2}\nabla \log \pi(x), \delta I\right)$ and α given by (4.10).

Finally, it is worth noting that the dynamics (4.16) are *reversible* with respect to π, i.e., under no discretisation error, their transition kernel $Q^\delta_{\text{true}}(\mathrm{d}y \mid x)$ verifies (4.3) for all $x, y \in \mathbb{R}^D$ and any time increment $\delta > 0$. This means that dynamics (4.16) exhibit diffusive behaviour, so that exploring the space X with these dynamics is akin to exploring it with a Brownian motion: travelling a distance L in the space X takes a time of order L^2. Such diffusive behaviour is often undesirable, as it leads to slow convergence of the chain to the target distribution π. In order to address this issue, it is often proposed to add a *momentum* variable to the dynamics (4.16), considering the augmented distribution

$$\pi(\mathrm{d}x, \mathrm{d}v) = \pi(\mathrm{d}x)\mathcal{N}(v \mid 0, M^{-1})\mathrm{d}v, \tag{4.18}$$

where M is a preconditioning matrix, called the *mass matrix*. This gives rise to the celebrated Hamiltonian Monte Carlo (HMC) method (Duane et al., 1987), which is based on the idea of following lines of constant energy in the augmented space $\mathsf{X} \times \mathsf{V}$ and then using a Metropolis–Hastings acceptance step to correct for the discretisation error. We do not discuss HMC further in this chapter, but refer the reader to Brooks et al. (2011, Chap. 5) and Betancourt (2018) for detailed introductions to the method and its extensions.

4.1.6 Gibbs sampling

Another popular class of MCMC methods are Gibbs samplers (Geman and Geman, 1984; Casella and George, 1992), which are based on the idea of constructing a Markov kernel K invariant with respect to π by composing a number of *conditional* kernels K_k on X that are invariant with respect to π. Assume that $\pi(x) = \pi(u,v)$ for some $x = (u,v) \in \mathsf{X}_1 \times \mathsf{X}_2 = \mathsf{X}$. We can form a Markov chain $(U_0, V_0) = X_0, X_1, \ldots$ by alternatively sampling

$$U_{k+1} \sim \pi(\mathrm{d}u \mid V_k), \quad V_{k+1} \sim \pi(\mathrm{d}v \mid U_{k+1}) \tag{4.19}$$

for all $k \geq 0$. Provided that $X_k = (U_k, V_k)$ is distributed according to π, we can see that X_{k+1} is then distributed according to π as well, so that the chain $X_0, X_1, \ldots$ is π-invariant.

Example 4.6. *Let $\pi(u,v) \propto \exp\left(-\frac{1}{2}\left(u^\top u \cdot v^\top v + u^\top u + v^\top v\right)\right)$, $u,v \in \mathbb{R}^d$. Directly sampling from π is difficult as it involves terms of order 4 in the exponential. However, $\pi(u \mid v) = \mathcal{N}\left(u; 0, \frac{1}{1+v^\top v} I\right)$ and $\pi(v \mid u) = \mathcal{N}\left(v; 0, \frac{1}{1+u^\top u} I\right)$ are Gaussian, so that we can sample from them easily. The Gibbs sampler (4.19) can then be used to sample from π.*

This is the basis of *Gibbs sampling* methods, which are often used when the conditional distributions $\pi(\mathrm{d}u \mid v)$ and $\pi(\mathrm{d}v \mid u)$ are easy to sample from, but the joint distribution $\pi(\mathrm{d}u, \mathrm{d}v)$ is not. Moreover, Gibbs sampling can be generalised to the case where π is a distribution on X for which the conditional distributions $\pi(\mathrm{d}x_k \mid x_{-k})$, where $x_{-k} = (x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_D)$, are available.

When one of the distributions $\pi(\mathrm{d}x_k \mid x_{-k})$ is not easy to sample from, it is also possible to use a Metropolis–Hastings kernel K_k to sample from it instead, giving rise to the *Hastings-within-Gibbs* algorithm (HwG, Diggle et al., 1998), which, in the case of bivariate distributions, can be written as

1. sample $U_{k+1} \sim \pi(\mathrm{d}u \mid V_k)$;
2. sample $V_{k+1} \sim K(\mathrm{d}v \mid V_k)$, where K is an MCMC kernel on X_2 with invariant distribution $\pi(\mathrm{d}v \mid U_{k+1})$.

In that case, if $X_k = (U_k, V_k)$ is distributed according to π, then following step 1, U_{k+1}, V_k is distributed according to π as well, and following step 2, U_{k+1}, V_{k+1} is therefore distributed according to π as well, so that the chain $X_0, X_1, \dots$ is π-invariant. We will make use of this idea in Sections 4.2 and 4.3.4.

4.2 Auxiliary samplers

A general strategy employed to design MCMC kernels that are invariant with respect to $\pi(\mathrm{d}x)$ is to consider an augmented space $\mathrm{X} \times \mathrm{U}$ and an augmented distribution $\pi(\mathrm{d}x, \mathrm{d}u) = \pi(\mathrm{d}x)\pi(\mathrm{d}u \mid x)$ on $\mathrm{X} \times \mathrm{U}$ such that the marginal $\pi(x) = \int \pi(x,u)\pi(\mathrm{d}u \mid x)$ is the target distribution of interest (Higdon, 1998). This was quickly alluded to in Section 4.1.5, where we considered the augmented target distribution $\pi(\mathrm{d}x, \mathrm{d}v) = \pi(\mathrm{d}x)\mathcal{N}(v; 0, M^{-1})\mathrm{d}v$ of HMC. In this section, we discuss three other examples of such augmented methods, the first one, called *pseudo-marginal MCMC* (Andrieu and Roberts, 2009), allows us to consider models with intractable densities; the second one, due to (Titsias and Papaspiliopoulos, 2018a) and called *auxiliary gradient-based samplers*, allows us to incorporate gradient and prior information into the model and generalises MALA; and the third one, which we call *ensemble MCMC* (Tjelmeland, 2004; Calderhead, 2014), allows us to use several samples at once to propose the next state of the chain.

4.2.1 Pseudo-marginal MCMC

Suppose that we are interested in sampling from a probability distribution $\pi(x) = \frac{f(x)}{Z}$ on X, only known up to a proportionality constant, but for which we further only have access to an unbiased estimator $\hat{f}(x)$ of the (unnormalised) density $f(x)$ via a relationship $f(x) = \int f(x,u)q(\mathrm{d}u \mid x)$ for all $x \in \mathrm{X}$, with $q(\mathrm{d}u \mid x)$ being a (conditional) probability distribution on U that we know how to sample from. This is typically the case of latent variable models such as state-space models, where the target $\pi(\theta) \propto p(y \mid \theta)p(\theta)$ is the posterior distribution of a parameter $\theta \in \Theta$ given some observations $y \in \mathrm{Y}$, and the marginal likelihood $p(y \mid \theta) = \int p(y \mid x, \theta)p(\mathrm{d}x \mid \theta)$ is given as the marginalisation over all possible values of the latent variables $x \in \mathrm{X}$.

In that case, a natural idea would be to use the estimator $\hat{f}(x) = f(x, U)$, $U \sim q(\mathrm{d}u \mid x)$ within a Metropolis–Hastings routine. This suggests targeting the augmented distribution $\pi(\mathrm{d}x, \mathrm{d}u) = q(\mathrm{d}u \mid x)\frac{f(x,u)}{Z}\mathrm{d}x$ on $\mathrm{X} \times \mathrm{U}$ which marginally corresponds to the target distribution $\pi(\mathrm{d}x)$ on X:

$$\int_{\mathrm{U}} \pi(\mathrm{d}x, \mathrm{d}u) = \int_{\mathrm{U}} q(\mathrm{d}u \mid x)\frac{f(x,u)}{Z}\mathrm{d}x = \frac{1}{Z}\int f(x,u)q(\mathrm{d}u \mid x)\mathrm{d}x = \frac{1}{Z}f(x)\mathrm{d}x = \pi(\mathrm{d}x). \tag{4.20}$$

Suppose now that we are given a proposal kernel $Q(\cdot \mid x)$ on X, which we would ideally want to use as part of the Metropolis–Hastings procedure for $\pi(x)$. We can augment it to a proposal kernel $Q(\mathrm{d}y, \mathrm{d}v \mid x, u)$ on $\mathsf{X} \times \mathsf{U}$ by setting

$$Q(\mathrm{d}y, \mathrm{d}v \mid x, u) = Q(\mathrm{d}y \mid x) q(\mathrm{d}v \mid y). \tag{4.21}$$

Putting everything together, we obtain the pseudo-marginal MCMC kernel

$$\begin{aligned} K(\mathrm{d}y, \mathrm{d}v \mid x, u) &= Q(\mathrm{d}y, \mathrm{d}v \mid x, u)\alpha(x, y, u, v) \\ &\quad + \delta_x(\mathrm{d}y)\delta_u(\mathrm{d}v)\left(1 - \int \alpha(x, z, u, w) Q(\mathrm{d}z, \mathrm{d}w \mid x, u)\right), \end{aligned} \tag{4.22}$$

for the acceptance probability $\alpha(x, y, u, v) = \min(1, \widetilde{\alpha}(x, y, u, v))$[6] with

$$\begin{aligned} \widetilde{\alpha}(x, y, u, v) &= \frac{\pi(y, v) Q(x, u \mid y, v)}{\pi(x, u) Q(y, v \mid x, u)}, \\ &= \frac{Z f(y, v) q(v \mid y) Q(x \mid y) q(u \mid x)}{Z f(x, u) q(u \mid x) Q(y \mid x) q(v \mid y)}, \\ &= \frac{f(y, v) Q(x \mid y)}{f(x, u) Q(y \mid x)}, \end{aligned} \tag{4.23}$$

which does not depend on the normalising constant Z nor on the sampling distribution $q(\mathrm{d}u \mid x)$.

Proposition 4.7. *Let K be the pseudo-marginal MCMC kernel* (4.22) *with acceptance probability* (4.23)*. Then K is $\pi(\mathrm{d}x, \mathrm{d}u)$-invariant.*

Proof. The proof follows from the detailed balance condition (4.3) and the contruction of the kernel K.

[6]We can also use Barker's acceptance.

Example 4.8. *Consider the following model for a random variable X on $\mathbb{R}^D$:*

$$\begin{aligned} U^d &\sim q^d(\cdot \mid X), \quad d = 1,\dots,D, \\ Y^d &\sim p^d(\cdot \mid U^d), \quad d = 1,\dots,D, \end{aligned} \tag{4.24}$$

where X is a parameter of interest with prior distribution $p(x)$ and the observations Y are assumed to be conditionally independent given X. This type of model is known as a random effect model in statistics, and is an instance of hierarchical models (see, e.g., Papaspiliopoulos et al., 2007, for a detailed study of these). Heuristically, the random variable X represents the common cause of the observations Y, while the random variable U represents the idiosyncratic cause of the observations Y.

Now, when given a sample $Y = y$, an unbiased estimator of the likelihood function $p(y \mid x) = \prod_{d=1}^D \int p^d(y^d \mid u^d) q^d(\mathrm{d}u^d \mid x)$ is given by

$$\hat{p}(y \mid x) \propto \prod_{d=1}^{D} \frac{1}{N} \sum_{n=1}^{N} p^d(y^d \mid U_n^d), \quad U_n^d \sim q^d(\cdot \mid x), \quad d = 1,\dots,D, n = 1,\dots,N, \tag{4.25}$$

for any $N \geq 1$. For instance, for $N = 1$ pseudo-marginal MCMC kernel K of (4.22) then corresponds to equating the following terms:

1. *$\pi(\mathrm{d}u \mid x) \leftarrow \prod_{d=1}^D q^d(\mathrm{d}u^d \mid x)$, and*
2. *$f(x,u) \leftarrow \prod_{d=1}^D p^d(y^d \mid u^d) p(x)$.*

In practice, it is not necessary to explicitly manipulate the auxiliary variables U in the pseudo-marginal MCMC kernel K and, instead, one may simply retain the resulting likelihood function estimate $\hat{f}$, resulting in a chain on the space $X, \hat{f}$ instead. This gives rise to Algorithm 9, which is the pseudo-marginal MCMC algorithm of Andrieu and Roberts (2009). It is worth noting nonetheless

Algorithm 9: Pseudo-marginal MCMC algorithm

```
input  : $X, \hat{f}(X)$
output : The next state $X, \hat{f}(X)$ of the chain.
1 Propose $Y \sim Q(\cdot \mid X)$
2 Compute an unbiased estimator $\hat{f}(Y)$ of $f(Y)$
3 Accept $Y, \hat{f}(Y)$ with the corresponding probability $\alpha$, otherwise retain
    $X, \hat{f}(X)$
```

that, as a side product, this algorithm also provides a Markov chain for the auxiliary variables U if desired.

Several improvements of pseudo-marginal MCMC methods have been proposed in the literature, in particular with respect to allowing for a more efficient exploration of the auxiliary space U, which is often necessary to mitigate the variance in the acceptance step induced by the use of the unbiased estimator. For example, when $q(u \mid x) = \mathcal{N}(u; 0, I)$ is Gaussian and does not depend on x, Deligiannidis et al. (2018) consider correlating the auxiliary variables U throughout the chain, making use of a kernel $Q(\mathrm{d}v \mid u)$ rather than simply sampling from $q(\mathrm{d}u)$, and show that, under a proper choice for Q, this leads to a non-degenerate algorithm in the limit of the number of auxiliary variables going to infinity. These methods, however, have the drawback of not using the information brought by the previous state X_k to propose the next auxiliary state U_{k+1}, as well as not using the information brought by the previous auxiliary state U_k to propose the next state X_{k+1}. In order to solve this, Alenlöv et al. (2021) consider using Hamiltonian dynamics on both the target space X and the auxiliary space U, and show that this leads to a more efficient exploration of the space X. Fundamental to these methods is the idea that, when $p^d(y^d \mid u^d)$ is a smooth function of u^d, small changes to u^d will result in small changes to $p^d(y^d \mid u^d)$, and therefore to the likelihood function estimate, so that the acceptance probability α of the Metropolis–Hastings kernel K will be dominated by its X component, which is the component of interest. Other approaches are given by Dahlin et al. (2015); Murray and Graham (2016); Tran et al. (2017); Nemeth et al. (2019).

4.2.2 Ensemble samplers

Another class of auxiliary MCMC methods are the *ensemble samplers* (Tjelmeland, 2004; Calderhead, 2014; Martino, 2018), which are based on the idea of using $N-1 > 1$ samples at once, rather than a single one, to propose the next state of the chain. Formally, rather than a proposal kernel $Q(\mathrm{d}y \mid x)$ on X, we consider a proposal distribution $Q(\mathrm{d}x^{-n} \mid x^n)$ on X^{N-1}, where $x^{-n} = (x^1, \dots, x^{n-1}, x^{n+1}, \dots, x^N) \in \mathsf{X}^{N-1}$ and then select one of the $N-1$ samples as the next state of the chain (or reject them all). In order to do so, we can consider the augmented distribution

$$\pi^N(\mathrm{d}x^{1:N}, k) = \frac{1}{N}\pi(\mathrm{d}x^k)Q(\mathrm{d}x^{-k} \mid x^k) \tag{4.26}$$

on $\mathsf{X}^N \times \{1, \dots, N\}$, where $x^{1:N}$ is a shorthand for $x^1, \dots, x^N$. Here k is an auxiliary variable that indicates which of the N samples corresponds to the state of the marginal chain for π. Clearly, the conditional $\pi^N(\mathrm{d}x^n \mid k = n) = \pi(\mathrm{d}x^n)$ is the target distribution of interest, and $\pi^N(k) = 1/N$ is marginally uniform. As a consequence, sampling from the target $\pi(\mathrm{d}x)$ can be done by forming a Markov kernel that is invariant with respect to π^N, and simply selecting the k-th component of the resulting N samples as the next state of the chain.

On the other hand, conditionally on the $x^{1:N}$, the variable k is then distributed as according to a categorical distribution:

$$\pi^N(k \mid x^{1:N}) = \frac{\pi(x^k)Q(x^{-k} \mid x^k)}{\sum_{n=1}^N \pi(x^n)Q(x^{-n} \mid x^n)} =: W^k, \tag{4.27}$$

where we have assumed that we could compute the density $Q(x^{-n} \mid x^n)$ of the proposal.

We can therefore sample from $\pi^N(\mathrm{d}x^{1:N}, k)$ by using the following Gibbs procedure. Suppose that $x^{1:N}, k$ is the current state of the chain, then

1. sample $x^{-k} \sim Q(\mathrm{d}x^{-k} \mid x^k)$ and form $x^{1:N}$;
2. sample $k \sim \pi^N(\mathrm{d}k \mid x^{1:N})$;
3. set $x^{1:N}, k$ as the next state of the chain.

Proposition 4.9. *The Gibbs procedure above is $\pi(\mathrm{d}x^{1:N}, \mathrm{d}k)$-invariant, namely, if $x^{1:N}, k$ is distributed according to $\pi(\mathrm{d}x^{1:N}, \mathrm{d}k)$, then the next state $x^{1:N}, k$ is still distributed according to $\pi(\mathrm{d}x^{1:N}, \mathrm{d}k)$.*

Proof. This is a direct consequence of Gibbs sampling being invariant with respect to the joint distribution of the variables being sampled from.

Remark 4.10. *Instead of sampling from Step 2 as a categorical distribution, it is also possible to implement a Metropolis–Hastings step there to "force" picking a different index l than k if possible. This corresponds to proposing a new index l on $\{1, \dots, N\} \setminus k$ with probability $W^l/(1 - W^k)$, and accepting it with probability*

$$\begin{aligned} \alpha &= \min\left(1, \frac{W^l W^k/(1-W^l)}{W^k W^l/(1-W^k)}\right), \\ &= \min\left(1, \frac{1-W^k}{1-W^l}\right). \end{aligned} \tag{4.28}$$

This is the so-called forced move strategy of Chopin and Singh (2015a) that we make use of in Publication VII.

Example 4.11. *In the case when $N = 2$, and when using the "forced move" strategy of Remark 4.10, the resulting procedure K recovers the Metropolis–Hastings kernel K on $\mathbb{R}^D$: in this specific instance,*

$$\begin{aligned} \frac{1-W^k}{1-W^l} &= \frac{\pi(x^l) Q(x^{-l} \mid x^l)}{\pi(x^k) Q(x^{-k} \mid x^k)}, \\ &= \frac{\pi(x^l) Q(x^k \mid x^l)}{\pi(x^k) Q(x^l \mid x^k)}. \end{aligned} \tag{4.29}$$

This idea is particularly fruitful when parallelisation is possible (Calderhead, 2014), but also forms the basis of the Particle-MCMC methods of Section 4.3.

On the other hand, a natural criticism of ensemble samplers is that they typically incur a computational cost that is quadratic in the number of samples N, as the weights appearing in Step 2 involve N quantities of the form $Q(x^{-n} \mid x^n)$, each of which typically will need to be computed at a cost of $\mathcal{O}(N)$, which altogether will then scale as $\mathcal{O}(N^2)$. However, this cost can be reduced when Q is an exchangeable kernel, i.e. when $Q(x^{-n} \mid x^n)$ corresponds to a density $Q(x^{1:N})$

that is exchangeable with respect to the order of the samples $x^1, \dots, x^N$: for any permutation σ of $\{1, \dots, N\}$, we have

$$Q(x^{\sigma(1)}, \dots, x^{\sigma(N)}) = Q(x^1, \dots, x^N). \tag{4.30}$$

In that case, the weights can be computed in a computationally efficient manner, as we have

$$\begin{aligned}\frac{\pi(x^l)Q(x^{-l} \mid x^l)}{\sum_{m=1}^N \pi(x^m)Q(x^{-m} \mid x^m)} &= \frac{\pi(x^l)Q(x^{-l}, x^l)/Q(x^l)}{\sum_{m=1}^N \pi(x^m)Q(x^{-m}, x^m)/Q(x^m)}, \\ &= \frac{\pi(x^l)/Q(x^l)}{\sum_{m=1}^N \pi(x^m)/Q(x^m)},\end{aligned} \tag{4.31}$$

which only requires evaluating the marginal densities $Q(x^m)$ for all m. While this gives a general principle, coming up with an exchangeable kernel Q with known marginal densities is not always straightforward.

Instead, using De Finetti's theorem (see, e.g., Diaconis and Freedman, 1980), we know that the proposal is exchangeable when $Q(x^{1:n}) = \int \prod_{n=1}^N Q(x^n \mid u)Q(\mathrm{d}u)$ for some probability distribution $Q(\mathrm{d}u)$ on U and some conditional probability distribution $Q(x \mid u)$ on X.

Example 4.12. *Informally, we see that taking $Q(\mathrm{d}u) \equiv 1$ to be an improper prior and $Q(\mathrm{d}y \mid u) \sim \mathcal{N}\left(y; u, \frac{\delta}{2}I\right)$, we obtain $Q(\mathrm{d}u \mid y) \sim \mathcal{N}\left(y, \frac{\delta}{2}I\right)$, and therefore an exchangeable kernel $Q(\mathrm{d}x^{1:N}) = \int \prod_{n=1}^N Q(\mathrm{d}x^n \mid u)Q(\mathrm{d}u)$. Note that this does not result in a well-defined joint distribution $Q(\mathrm{d}x^{1:N})$ but results in well-defined proposal kernels $Q(\mathrm{d}y \mid x) \sim \mathcal{N}(y; x, \delta I)$, recovering the Random Walk Metropolis kernel.*

Following Example 4.12, we can define exchangeable proposals using conditionals $Q(\mathrm{d}x \mid u)$ and $Q(\mathrm{d}u \mid x)$, for symmetric kernels $Q(x \mid u) = Q(u \mid x)$, and use these within the ensemble sampler to obtain a computationally efficient algorithm.

This corresponds to augmenting the space X with an auxiliary variable U and considering the augmented distribution (again!)

$$\pi^N(\mathrm{d}x^{1:N}, \mathrm{d}u, k) = \frac{1}{N}\pi(\mathrm{d}x^k)R(\mathrm{d}u \mid x^k)\prod_{i \neq k} Q(\mathrm{d}x^i \mid u), \tag{4.32}$$

for a chosen $R(\mathrm{d}u \mid x)$ which may or may not be equal to $Q(\mathrm{d}u \mid x)$. Now, conditionally on $x^{1:N}$ and u, the variable k is then distributed as a categorical distribution

$$\pi^N(k \mid x^{1:N}, u) = \frac{\pi(x^k)R(u \mid x^k)/Q(x^k \mid u)}{\sum_{n=1}^N \pi(x^n)R(u \mid x^n)/Q(x^n \mid u)} =: W^k. \tag{4.33}$$

We can therefore implement the following Gibbs procedure to sample from $\pi^N(\mathrm{d}x^{1:N}, \mathrm{d}u, k)$. Given the current state of the augmented chain $x^{1:N}, u, k$, the next state of the chain is formed by the following steps:

1. sample $u \sim R(\mathrm{d}u \mid x^k)$;
2. for all $i \neq k$, sample $x^i \sim Q(\mathrm{d}x^i \mid u)$;
3. sample $k \sim \pi^N(\mathrm{d}k \mid x^{1:N}, u)$;
4. set $x^{1:N}, u, k$ as the next state of the chain.

Remark 4.13. *We are not aware of a similar presentation of ensemble samplers with exchangeable proposals in the literature, but we believe that this is a natural way to think about them. In particular, when Q is symmetric, and when $R(u \mid x) = Q(u \mid x)$, the Gibbs procedure above is equivalent to the ensemble sampler of Tjelmeland (2004) and the auxiliary variable does not appear in the acceptance probability. However, our presentation allows us to consider more general choices of $R(u \mid x)$ and $Q(x \mid u)$, which we make use of in Publication V and even more so in Publication VII by considering the decompositions given in Section 4.2.3. We believe it may be of interest to consider even more general choices of these but leave this for future work.*

4.2.3 MALA as an auxiliary sampler and beyond

In this section, we present MALA as an auxiliary sampler, and discuss how this can be generalised to other gradient-based MCMC methods.

Remark 4.14. *The presentation follows Titsias and Papaspiliopoulos (2018a) with the exception that we consider a different (but marginally equivalent) definition of the auxiliary distribution $\pi(x,v)$, which presents the benefit of being more amenable to ensemble samplers. This choice follows the use we make of this perspective in Publication VII (to the detriment of the one we make in Publication V).*

Consider first the augmented space $\mathrm{X} \times \mathrm{U}$ and augmented distribution $\pi(x,u) = \pi(x)\mathcal{N}\left(u; x, \frac{\delta}{2} I\right)$ on $\mathrm{X} \times \mathrm{U}$, where $\delta > 0$ is a given parameter. Suppose that the state of the chain is X_t, U_t; similar to the previous section, a natural way to interpret the RWM kernel K on $\mathbb{R}^D$ is then in terms of a Hastings-within-Gibbs kernel targeting $\pi(x,u)$:

1. sample $U_{t+1} \sim \mathcal{N}\left(X_t, \frac{\delta}{2} I\right)$;
2. sample X_{t+1} from an MH kernel targeting $\pi(\mathrm{d}x \mid U_{t+1})$, with a proposal $Q(x \mid U_{t+1}) = \mathcal{N}\left(x; U_{t+1}, \frac{\delta}{2} I\right)$.

The acceptance probability of Step 2 (for Barker's acceptance ratio) is then given as

$$\alpha(x,y) = \frac{\pi(y,u)\mathcal{N}\left(x; u, \frac{\delta}{2} I\right)}{\pi(y,u)\mathcal{N}\left(x; u, \frac{\delta}{2} I\right) + \pi(x,u)\mathcal{N}\left(y; u, \frac{\delta}{2} I\right)}, \tag{4.34}$$

which simplifies to $\frac{\pi(y)}{\pi(y)+\pi(x)}$, and does not depend on the auxiliary variable u. Note that this structure is still valid when the acceptance probability is Rosenbluth–Teller's, but we do not give the details here.

Remark 4.15. *An important property of this method is the fact that (conditionally on u), the proposal kernel $Q(\mathrm{d}x \mid x, u)$ does not depend on the previous state x, so that we can easily define a kernel $Q(\mathrm{d}u, \mathrm{d}x^{-n} \mid x^n) = Q(\mathrm{d}u \mid x^n) \prod_{m \neq n} Q(\mathrm{d}x^m \mid u)$ to cheaply use ensemble samplers to propose the next state of the chain. This structure is shared with the rest of the algorithms presented in this section.*

This construction can be generalised to gradient-based MCMC methods, such as MALA, as follows. Consider the augmented space $\mathsf{X} \times \mathsf{U}$ and augmented distribution $\pi(x, u) = \pi(x) \mathcal{N}\left(u; x + \frac{\delta}{2} \nabla \log \pi(x), \frac{\delta}{2} I\right)$ on $\mathsf{X} \times \mathsf{U}$, where $\delta > 0$ is a given parameter. Suppose that the state of the chain is X_t, V_t; we can write an auxiliary MALA kernel K on $\mathbb{R}^D$ as a Hastings-within-Gibbs kernel targeting $\pi(x, u)$, given by

1. Sample $U_{t+1} \sim \mathcal{N}\left(X_t + \frac{\delta}{2} \nabla \log \pi(X_t), \frac{\delta}{2} I\right)$.
2. Sample X_{t+1} from an MH kernel targeting $\pi(x \mid U_{t+1})$, with a proposal $Q(\mathrm{d}x \mid U_{t+1}) \sim \mathcal{N}\left(U_{t+1}, \frac{\delta}{2} I\right)$.

The acceptance probability of Step 2 (for Barker's acceptance ratio) is then given as

$$\alpha(x, y, u) = \frac{\pi(y, u) \mathcal{N}\left(x; u, \frac{\delta}{2} I\right)}{\pi(y, u) \mathcal{N}\left(x; u, \frac{\delta}{2} I\right) + \pi(x, u) \mathcal{N}\left(y; u, \frac{\delta}{2} I\right)}, \tag{4.35}$$

which now depends on the auxiliary variable u. This algorithm can be proven to be equivalent to MALA after marginalising out the auxiliary variable u in closed form, hence its designation as *auxiliary MALA*.

Pre-conditioned versions can also be formulated, such as the pre-conditioned Crank–Nicolson–Langevin (pCNL) kernel (Cotter et al., 2013), which we do not discuss further in this thesis[7]. Instead, we focus on the "aGrad" algorithm of Titsias and Papaspiliopoulos (2018a, note that we specifically consider their "aGrad-z" version), which was shown to be empirically more efficient than pCNL. In order to derive aGrad, we consider the case when the target distribution $\pi(x) \propto \mathcal{N}(x; 0, C) \exp(f(x))$ corresponds to a density with a Gaussian prior $\mathcal{N}(x; 0, C)$.[8] We can then consider the augmented space $\mathsf{X} \times \mathsf{U}$ and distribution $\pi(x, u) = \pi(x) \mathcal{N}\left(u; x + \frac{\delta}{2} \nabla f(x), \frac{\delta}{2} I\right)$ on $\mathsf{X} \times \mathsf{U}$, where $\delta > 0$ is a given parameter. The goal then is to find an efficient way to sample from $\pi(x \mid u)$. To do so, let X_t be the previous state of the chain, and $U_{t+1} \sim \mathcal{N}\left(X_t + \frac{\delta}{2} \nabla f(X_t), \frac{\delta}{2} I\right)$ be the next auxiliary state. Under the assumption of a small δ, we can approximate $\pi(x \mid u)$ by

$$\begin{aligned} \pi(x \mid u) &\approx \mathcal{N}\left(u; x, \frac{\delta}{2} I\right) \mathcal{N}(x; 0, C), \\ &= \mathcal{N}\left(x; \frac{2}{\delta} A^\delta u, A^\delta\right), \\ &=: Q(x \mid u), \end{aligned} \tag{4.36}$$

[7]We discuss this construction in the Appendix of Publication VII.

[8]Note that this can also be applied to a non-zero mean m by considering the change of likelihood $f(x) \leftarrow f(x) + x^\top C^{-1} m$.

where $A^\delta = \frac{\delta}{2}\left(\frac{\delta}{2}I + C\right)^{-1}C$. Again, this proposal kernel q does not depend on the previous state X_t, and can therefore be used in ensemble samplers. Whilst not critical in the context of this section, the proposal (4.36) presents the desirable property that its covariance matrix A^δ can be updated in a computationally efficient manner. By this, we mean that the cost of modifying δ, for example in the context of an adaptive MCMC algorithm as discussed above, is quadratic in the dimension D instead of the cubic cost that would be incurred by a naive implementation recomputing the Cholesky decomposition of A^δ at each iteration. This is achieved by noting that the eigenvectors of A^δ are the same as those of C, and that the resulting eigenvalues are easily obtained, see Titsias and Papaspiliopoulos (2018a, Supplementary material) for details.

Remark 4.16. *The auxiliary method presented in this section is closely related to proximal samplers (Chen et al., 2022), which, too, consider the same, or similar, augmented distribution $\pi(x,u)$. Such proximal algorithms have recently received some attention in the MCMC literature (see, e.g., Durmus et al., 2018), where they are used to design efficient approximate or exact MCMC kernels that make use of local gradient information. The link between these and Titsias and Papaspiliopoulos (2018a) aGrad algorithm is surprisingly under-utilised, and would deserve to be explored further.*

4.3 Particle MCMC

In this section, we discuss a class of MCMC methods that are specifically designed to sample from distributions of Markovian models, as given in Chapter 3. While the methods of Section 4.1 are general-purpose, they are typically ill-suited to sample from structured distributions, where coordinates of the state X show strong dependencies, in our case temporal dependencies. In order to address this, a natural way to proceed is to try and modify particle filters to implement MCMC kernels that are invariant with respect to the target distribution π_T. Throughout this section, for ease of reference, we follow the notations of Section 3.4 and consider a target distribution $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T}) = \pi_T(\mathrm{d}x_{0:T} \mid \theta)p(\mathrm{d}\theta)$ on $\Theta \times \mathrm{X}_0 \times \cdots \times \mathrm{X}_T$ that can be defined recursively as

$$\pi_{t+1}(\mathrm{d}x_{0:t+1} \mid \theta) \propto M_{t+1}(x_{t+1} \mid x_{0:t})G_{t+1}(x_{0:t+1})\pi_t(\mathrm{d}x_{0:t} \mid \theta), \tag{4.37}$$

for some transition kernels M_t and potential functions G_t which may depend on the parameter θ. To avoid notational clutter[9], we do not make this dependence explicit in the following, but the reader should keep in mind that it is always present. The case of Markovian models is recovered by taking the case when M_t depends on $x_{0:t-1}$ only via x_{t-1}, i.e., when $M_t(\mathrm{d}x_t \mid x_{0:t-1}) = M_t(\mathrm{d}x_t \mid x_{t-1})$, and when G_t depends on $x_{0:t}$ only via x_t and x_{t-1}, i.e., when $G_t(x_{0:t}) = G_t(x_t, x_{t-1})$.

[9]Particle MCMC methods are already unkind enough in this respect.

While this choice does not have *methodological* consequences, it has *computational* consequences: when the proposals and potentials are Markovian, the algorithms in this section can be implemented using $\mathcal{O}(T)$ operations, whereas the general case requires $\mathcal{O}(T^2)$ operations.

The methods presented in this section can be seen as a generalisation of the auxiliary MCMC methods of Section 4.2 to Markovian models:

1. particle marginal Metropolis–Hastings (PMMH) and particle independent Metropolis–Hastings (PIMH, Andrieu et al., 2010) can be seen as a generalisation of pseudo-marginal MCMC (see Section 4.2.1);
2. conditional sequential Monte Carlo (Andrieu et al., 2010, CSMC,) can be seen as a generalisation of ensemble samplers with independent proposals (see Section 4.2.2); and
3. random walk CSMC (RW-CSMC, Finke and Thiery, 2023) can be seen as a generalisation of ensemble samplers with auxiliary RWM proposals (see Sections 4.2.2 and 4.2.3).

4.3.1 Particle marginal Metropolis–Hastings

In Section 4.2.1, we have described how an MCMC chain could be constructed for a target distribution π when one has access to an unbiased estimator $\hat{Z}$ of the normalising constant Z of the model π. Thankfully, and as discussed in Section 3.4.1 sequential Monte Carlo methods provide such an estimator L_T^N of $L_T = \int \prod_{t=0}^T G_t(x_{0:t})\pi_t(\mathrm{d}x_t \mid x_{0:t-1})$ as a by-product of the algorithm. When the model further depends on a parameter $\theta \in \Theta$, and given a proposal mechanism for θ, one can then form a Markov chain on the space $\Theta \times \mathrm{U}$ of the form

$$K(\mathrm{d}\theta', \mathrm{d}u \mid \theta, u) = Q(\mathrm{d}\theta' \mid \theta)\alpha(\theta, \theta', u, v)\pi(\mathrm{d}v \mid \theta') + \delta_\theta(\mathrm{d}\theta')\delta_u(\mathrm{d}v)\left(1 - \rho(\theta, u)\right), \tag{4.38}$$

where $\rho(\theta, u)$ is the rejection probability of the algorithm and where the variable u represents *all the random variables* used in the SIR algorithm, which we now make explicit.

Proposition 4.17 (Distribution of all variables generated in SIR)**.** *Let $x_{0:T}^{1:N}$ and $a_{0:T-1}^{1:N}$ be the particles and ancestors generated by the SIR algorithm, respectively, and write $x_{0:t}^{(n)} = [x_{0:t-1}^{(a_{t-1}^n)}, x_t^n]$ for the genealogy of the n-th particle. The joint distribution of all the random variables generated in the SIR algorithm is given by*

$$q_T^N(\mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}) := \prod_{n=1}^N M_0(\mathrm{d}x_0^n) \prod_{t=1}^T \left\{ \mathscr{R}(\mathrm{d}a_{t-1}^{1:N} \mid W_{t-1}^{1:N}) \prod_{n=1}^N M_t(\mathrm{d}x_t^n \mid x_{0:t-1}^{(a_{t-1}^n)}) \right\}, \tag{4.39}$$

where $W_t^n = \frac{G_t(x_{0:t}^{(n)})}{\sum_{m=1}^N G_t(x_{0:t}^{(m)})}$ is the normalised weight of the n-th trajectory at time t and where we have omitted the dependency on θ for ease of notation.

Proof. The proof follows by tracking the genealogy of the particles, and noting that the ancestors only depend on the particles through the weights. See Naesseth et al. (2019); Chopin and Papaspiliopoulos (2020) for details. □

Example 4.18. *When multinomial resampling is used, the ancestors are conditionally independent, and their distribution is given by*

$$\mathscr{R}_t(a_t^{1:N} \mid W_t^{1:N}) = \prod_{n=1}^{N} W_t^{a_t^n}. \tag{4.40}$$

Following Section 4.2.1, and interpreting the particles and ancestors as latent, we can then define the augmented target distribution

$$\pi_T^N(\mathrm{d}\theta, \mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}) = p(\mathrm{d}\theta) q_T^N(\mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N} \mid \theta) \frac{L_T^N(\theta)}{L_T(\theta)}, \tag{4.41}$$

which marginally recovers $\pi_T(\mathrm{d}\theta)$ because $\mathbb{E}_{q_T^N}[L_T^N \mid \theta] = L_T(\theta)$.

As a consequence, we can leverage the method presented in Section 4.2.1 to construct an MCMC kernel for the parameters θ of the model $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T})$. Finally, we note that the method can be extended to marginally target the joint density $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T})$ by means of further introduction of auxiliary variables, which is how the method was originally presented in Andrieu and Roberts (2009); Andrieu et al. (2010). The discussion of this extension is largely similar to the one presented in Section 4.2.1 and we therefore omit it here.

4.3.2 Conditional SMC and particle Gibbs

In Section 4.3.1 we have discussed how SIR could be used to construct a pseudo-marginal MCMC kernel for the parameters θ of a model $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T})$. As mentioned there, this can be extended to also target the joint distribution $\pi_T(\mathrm{d}x_{0:T}, \mathrm{d}\theta)$ by further considering additional auxiliary variables. This nonetheless requires the choice of a proposal mechanism for the parameters θ *with no dependency on the trajectory*. This may be seen as a limitation, as it is often the case that the distribution $\pi_T(\mathrm{d}\theta \mid x_{0:T})$ is sometimes known in closed form, or is more easily approximated than the distribution $\pi_T(\mathrm{d}\theta)$ directly[10].

In this section, we describe an alternative approach that is based on the idea of using SIR to construct a Metropolis-within-Gibbs-like sampler for a model $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T})$, essentially using a generalisation of ensemble samplers to sample from the state given the parameters. To do so, we further assume that $\mathscr{R}$ is the multinomial resampling method, see (4.40) but note that this also can

[10]Additionally, even in the case when the model does not depend on a parameter θ, PIMH is less performant than the conditional SMC method presented in this section, in the sense that its mixing properties do not improve as the number of particles N increases, which is not the case for CSMC (Andrieu et al., 2018, Appendix F) and further justifies introducing the method.

be achieved for other resampling mechanisms, see Chopin and Singh (2015b); Karppinen et al. (2023) for details. This method was first introduced in Andrieu et al. (2010), and is known as conditional SMC (CSMC).

To do so, consider the target distribution $\pi_T^N(\mathrm{d}\theta, \mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N})$ of PMMH and augment it with an additional auxiliary variable $k \in \{1, \dots, N\}$ that indicates which of the N particles genealogies will be used to propose the next state of the chain.

$$\pi_T^N(\mathrm{d}\theta, \mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}, k) = \pi_T^N(\mathrm{d}\theta, \mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}) W_T^k, \tag{4.42}$$

where $W_T^k = \frac{G_T(x_{0:T}^{(k)})}{\sum_{m=1}^N G_T(x_{0:T}^{(m)})}$ is the normalised weight of the k-th trajectory at time T. We can then show, limiting ourselves for simplicity to the case when $\mathscr{R}(a_t^{1:N} \mid W_t^{1:N}) = \prod_{n=1}^N W_t^{a_t^n}$ is the multinomial resampling mechanism (see Karppinen et al., 2023, Theorem 2, for a proof in a more general case), that the following proposition holds.

Proposition 4.19 (Conditional SMC decomposition)**.** *We have*

$$\pi_T^N(\mathrm{d}\theta, \mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}, k) = \prod_{n \neq l_0^k}^N M_0(\mathrm{d}x_0^n) \prod_{t=1}^T \prod_{n \neq l_t^k}^N W_{t-1}^{a_{t-1}^n} M_t(\mathrm{d}x_t^n \mid x_{0:t-1}^{(a_{t-1}^n)}) \tag{4.43a}$$

$$\times \frac{1}{N^{T+1}} p(\mathrm{d}\theta) \pi_T(\mathrm{d}x_{0:T}^{(k)} \mid \theta), \tag{4.43b}$$

where l_t^k is the ancestor of the k-th trajectory at time t: $l_T^k = k$, and $l_t^k = a_t^{l_{t+1}^k}$ for $t = T-2, \dots, 0$. Furthermore, we have $\pi_T^N(k \mid \theta, x_{0:T}^{1:N}, a_{0:T-1}^{1:N}) = W_T^k$ and $\pi_T^N(k) = \frac{1}{N}$.

Proof. We omit θ for ease of notation. By reverse induction on T, we have

$$\begin{aligned}
\pi_T(\mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}, k) &= q_T^N(\mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}) \frac{L_T^N}{L_T} W_T^k \\
&= q_{T-1}^N(\mathrm{d}x_{0:T-1}^{1:N}, \mathrm{d}a_{0:T-2}^{1:N}) \prod_{n=1}^N W_{T-1}^{a_{T-1}^n} M_T(\mathrm{d}x_T^n \mid x_{0:T-1}^{(a_{T-1}^n)}) \\
&\quad \times \frac{1}{N} \frac{L_{T-1}^N \sum_{n=1}^N G_T(x_{0:T}^{(n)})}{L_T} \frac{G_T(x_{0:T}^{(k)})}{\sum_{n=1}^N G_T(x_{0:T}^{(n)})} \\
&= \pi_{T-1}^N(\mathrm{d}x_{0:T-1}^{1:N}, \mathrm{d}a_{0:T-2}^{1:N}) \prod_{n=1}^N W_{T-1}^{a_{T-1}^n} M_T(\mathrm{d}x_T^n \mid x_{0:T-1}^{(a_{T-1}^n)}) \\
&\quad \times \frac{1}{N} \frac{L_{T-1}^N}{L_{T-1}} \frac{L_{T-1}}{L_T} G_T(x_{0:T}^{(k)}) \\
&= q_{T-1}^N(\mathrm{d}x_{0:T-1}^{1:N}, \mathrm{d}a_{0:T-2}^{1:N}) \frac{L_{T-1}^N}{L_{T-1}} W_{T-1}^{l_{T-1}^k} \qquad (4.44) \\
&\quad \times \prod_{n \neq l_T^k} W_{T-1}^{a_{T-1}^n} M_T(\mathrm{d}x_T^n \mid x_{0:T-1}^{(a_{T-1}^n)})
\end{aligned}$$

$$
\begin{aligned}
&\times \frac{1}{N}\frac{L_{T-1}}{L_T} M_T(\mathrm{d}x_T^k \mid x_{0:T-1}^{(a_{T-1}^k)}) G_T(x_{0:T}^{(k)}) \\
&\vdots \\
&= \prod_{n \neq l_0^k}^N M_0(\mathrm{d}x_0^n) \prod_{t=1}^T \prod_{n \neq l_t^k}^N W_{t-1}^{a_{t-1}^n} M_t(\mathrm{d}x_t^n \mid x_{0:t-1}^{(a_{t-1}^n)}) \\
&\times \frac{1}{N^{T+1}} \pi_T(\mathrm{d}x_{0:T}^{(k)}).
\end{aligned}
$$

The induction then applies to (4.44) by unpacking elements from q_{T-1}^N into (4.43a) and (4.43b) one at a time. The proof of the other claims is straightforward. □

The distribution (4.43b) recovers the target distribution $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T})$, while the distribution (4.43a), is known as the conditional SMC distribution, and is the distribution of the particles and ancestors generated by the SIR algorithm conditional on the kth trajectory being fixed: informally,

$$
(4.43a) = \pi_T^N(\mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N} \mid \theta, k, x_{0:T}^{(k)}). \tag{4.45}
$$

Following Section 4.2.2, we can use this to construct valid MCMC kernel targeting $\pi_T^N(\mathrm{d}\theta, \mathrm{d}x_{0:T}^{1:N}, \mathrm{d}a_{0:T-1}^{1:N}, k)$. Suppose that $\theta, x_{0:T}^{1:N}, a_{0:T-1}^{1:N}, k$ is the current state of the chain (note that this implies that $\theta, x_{0:T}^{(k)}$ is then distributed according to $\pi_T(\mathrm{d}\theta, \mathrm{d}x_{0:T})$), then the next state of the chain is obtained by the following steps:

1. sample $\theta \sim \pi_T(\mathrm{d}\theta \mid x_{0:T}^{(k)})$: this can be done in closed form or using an MCMC kernel targeting $\pi_T(\mathrm{d}\theta \mid x_{0:T}^{(k)})$;
2. sample from (4.43a) to obtain $x_{0:T}^{1:N}, a_{0:T-1}^{1:N}$ (conditionally on k, θ and $x_{0:T}^{(k)}$);
3. sample $k \sim \pi_T^N(k \mid \theta, x_{0:T}^{1:N}, a_{0:T-1}^{1:N})$.

This algorithm is known as Particle Gibbs, and its version without θ as conditional SMC (CSMC, Andrieu et al., 2010), and is a generalisation of ensemble samplers with independent proposals to the case of Markovian models.

Remark 4.20. *While explicitly writing the method in terms of the full joint distribution* (4.42) *is useful for the purpose of justifying the method, analogously to the case of ensemble samplers, the method can likewise be implemented by only keeping track of* θ, k, *and* $x_{0:T}^{(k)}$.

The CSMC algorithm, corresponding to the case when no parameter θ is present, is known to be ergodic under mild assumptions and its efficiency is now well-studied (Chopin and Singh, 2015b; Lindsten et al., 2015; Andrieu et al., 2018; Lee et al., 2020). However, the algorithm still suffers from genealogical degeneracy, which was the motivation for the development of backward simulation methods as discussed in Section 3.4.3. Thankfully, it can be shown that instead of simply tracing back the lineage of k: $l_t^k = a_t^{l_{t+1}^k}$ for $t = T-2, \ldots, 0$ to form the conditional SMC distribution (4.43a), which corresponds to Kitagawa

(1994) smoother, one can instead use the backward simulation algorithm of Section 3.4.3 to mitigate the genealogical degeneracy of the algorithm (for a complete description of the backward sampling[11] step in general models, we refer to Lindsten and Schön 2013; see also Lindsten et al. 2014 for an alternative approach, statistically equivalent for Markovian models). This algorithm, known as CSMC with backward sampling, was first proposed in Whiteley (2010), and is known to be mixing in $\mathcal{O}(\log T)$ steps (Karjalainen et al., 2023)[12], for any fixed number of particles N, an improvement over the previous $\mathcal{O}(T)$ bound (Lee et al., 2020). This is in contrast with the original CSMC algorithm, which is known to be mixing in $\mathcal{O}(\exp(T))$ steps (see, e.g., Andrieu et al., 2018; Lee et al., 2020, for quantitative bounds). Finally, and similar to Section 3.4.3, we can reduce the computational cost of the backward sampling step by using rejection sampling (Cardoso et al., 2023) routines. However, we believe that using it *as is* is generally a bad idea, owing to the underlying mechanism having a random run time with infinite expectation (Dau and Chopin, 2023). Nonetheless, it seems plausible that the improvements proposed in Dau and Chopin (2023) for unconditional backward sampling would also apply to the conditional case. We leave this as an open question for future research.

The only question remaining is how to sample from the distribution (4.43a) used in step 2 of the Particle Gibbs procedure. In practice this simply corresponds to running the SIR algorithm, but with the additional constraint that the k-th trajectory is fixed. This latter condition is easy in the case of the multinomial resampling mechanism (4.40), as the resampling is then independent for all particles, but requires more care in the case of other resampling mechanisms, see Chopin and Singh (2015b); Karppinen et al. (2023) for details. We provide an implementation of CSMC in Algorithm 10 with or without backward sampling, where we have omitted the dependency on θ for ease of notation.

Remark 4.21. *In Publication IV, we offer a construction of a conditional SMC algorithm within the context of divide-and-conquer SMC, itself presented in Section 5.3.*

4.3.3 Why does CSMC work?

In Section 4.1.3 we ascertained that Metropolis–Hastings algorithms with an independent proposal mechanism where usually inefficient, owing to the fact that they do not use the information contained in the past states of the chain to propose the next state. This is not improved by the use of the ensemble sampler presented in Section 4.2.2, and it can be shown (this is a special case of Finke and Thiery, 2023, Section 2.3) that as the dimension of the latent state space

[11]A description is also available in Chopin and Papaspiliopoulos (2020, Chap. 16), but with the caveat that it is incorrect for multivariate potentials G_t, see the errata.
[12]Informally this means that approximately only $\log T$ iterations of the Markov chain are needed to obtain a "true" sample from the target distribution.

Algorithm 10: Conditional Sequential Monte Carlo (CSMC)

```
input  : Feynman–Kac model $(\pi_t)_{t=0}^T$ given by (4.37), number of particles
         $N$, reference trajectory $x_{0:T}$, ancestry lineage $l_{0:T}$.
output : New reference trajectory $x_{0:T}$ and ancestry lineage $l_{0:T}$.
1  Sample $X_0^n \sim M_0(\mathrm{d}x_0)$, $n \neq l_0$ and set $X_{0:0}^{l_0} \leftarrow x_0$
2  Set $w_0^n \leftarrow G_0(X_{0:0}^n)$, $W_0^n \leftarrow \frac{w_0^n}{\sum_{m=1}^N w_0^m}$, $n = 1,\dots,N$
3  for $t \in \{1,\dots,T\}$ do
4      Sample $A_{t-1}^n \sim \mathrm{Multinomial}(W_{t-1}^{1:N})$, $n \neq l_t$, and set $A_{t-1}^{l_t} \leftarrow l_{t-1}$
5      Sample $X_t^n \sim M_t(\mathrm{d}x_t \mid X_{0:t-1}^{(A_{t-1}^n)})$, $n \neq l_t$, and set $X_t^{l_t} \leftarrow x_t$
6      Set $w_t^n \leftarrow G_t(X_{0:t}^{(n)})$, $W_t^n \leftarrow \frac{w_t^n}{\sum_{m=1}^N w_t^m}$
7  Sample $l_T \sim \mathrm{Multinomial}(W_T^{1:N})$[13]
8  Set $x_T = X_T^{l_T}$
9  if Backward sampling then
10     ▷ All operations involving the index $n$ need to be done for $n = 1,\dots,N$
11     for $t \in \{T-1,\dots,0\}$ do
12         Set $\widetilde{w}_t^n \leftarrow \frac{\pi_T(X_{0:t}^{(n)}, x_{t+1:T})}{\pi_t(X_{0:t}^{(n)})} W_t^n$
13         Set $\widetilde{W}_t^n \leftarrow \frac{\widetilde{w}_t^n}{\sum_{m=1}^N \widetilde{w}_t^m}$
14         Sample $l_t \sim \mathrm{Multinomial}(\widetilde{W}_t^{1:N})$
15         Set $x_{t:T} \leftarrow [X_t^{l_t}, x_{t+1:T}]$
16 else
17     for $t \in \{T-1,\dots,0\}$ do
18         Set $l_{t-1} \leftarrow A_{t-1}^{l_t}$
19         Set $x_{t:T} \leftarrow [X_t^{l_t}, x_{t+1:T}]$
```

increases, the number of samples required to obtain a reasonable MCMC method increases exponentially.

Nonetheless, we have presented CSMC, a generalisation of ensemble samplers with independent proposals[14] to models of the form (4.37), and have claimed that, when backward sampling is employed, it is efficient even for a small number of particles N. It may at first seem like we are contradicting ourselves given that the dimension of the space is $\mathcal{O}(T)$, where T is the number of time steps. However, the key difference between CSMC and the ensemble sampler is that using N particles in the former heuristically results in N^{T+1} possible trajectories to propose from (see also Deligiannidis et al., 2020, for an algorithm with a similar intuition) rather than just N.

While this is not a complete explanation, and comes with numerous caveats (mostly owing to the resampling step making particles not independent), it is a good starting point to understand why these algorithms are often efficient despite their independent proposal mechanism. On the other hand, the independence of the proposal mechanism cannot be counterbalanced in the same way for the dimension D of the state space itself, and other methods need to be used to mitigate the degeneracy of the algorithm when it is large. We present one such method in the next section.

4.3.4 Particle-RWM

While CSMC with backward sampling is a valid MCMC kernel, it selects its next state from particles generated during the filtering step of SIR. This implies that it suffers from the same curse of dimensionality as SIR, itself inherited from importance sampling, whereby the effective sample size (see Section 3.4.2) collapses exponentially fast with the dimension of the state space. To obtain a reasonable approximation of the target distribution, we would consequently have to use an exponentially large number of particles (Finke and Thiery, 2023, Section 2.3).

Of course this is not a satisfactory solution, and we would like to find a way to sample from the target distribution $\pi(\mathrm{d}x_{0:T}, \mathrm{d}\theta)$ without having to resort to an unreasonable number of particles. Thankfully, the kinship between CSMC and ensemble samplers suggests a solution: we can use the same idea as in Section 4.2.2 and generalise random walk Metropolis proposals to CSMC the same way we used them in the context of ensemble samplers in Section 4.2.2. This idea was first introduced in Finke and Thiery (2023), and is known as particle-RWM (random-walk CSMC therein, see also Malory, 2021, for a similar approach) which we now describe. For simplicity, we do not consider the case when the model depends on a parameter θ, which can be handled by adding an

[13]Similar to Section 4.2.2, we can use a forced-move step here (Chopin and Singh, 2015a)

[14]Note that we do not mean here that X_t^n is independent of the previous state $X_{0:t-1}^{A_{t-1}^n}$ but rather that the proposals do not depend on the reference trajectory $x_{0:T}$.

intermediate step to the algorithm, as in the case of Particle Gibbs.

Particle-RWM is a generalisation of ensemble samplers with auxiliary RWM proposals to the case of Markovian models, whereby it targets the marginal distribution $\pi(\mathrm{d}x_{0:T}, \mathrm{d}u_{0:T}) = \pi(\mathrm{d}x_{0:T}) \prod_{t=0}^{T} \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2} I\right) \mathrm{d}u_t$, with $\delta_t > 0$ being a given sequence of hyperparameters. This augmented distribution can then be sampled from using the particle Gibbs procedure of Section 4.3.2, where $u_{0:T}$ serves as the parameter of the model for an improper prior $p(\mathrm{d}u_{0:T}) \equiv 1$. Informally, the resulting procedure can then be summarised as follows: if the current state of the chain is $(x_{0:T}, u_{0:T})$ then the next state is obtained by the following steps:

1. sample $u_t \sim \mathcal{N}\left(x_t, \frac{\delta_t}{2} I\right)$, $t = 0, \dots, T$;
2. sample $x_{0:T}$ from a CSMC kernel targeting $\pi(\mathrm{d}x_{0:T} \mid u_{0:T})$.

At first sight, this algorithm seems to be *worse* than CSMC, as it incorporates additional observations $u_{0:T}$, possibly with very high informativeness (if δ_t is small), which would make the algorithm degenerate even faster. However, it is possible to use these to construct a proposal kernel that is more efficient than the original one: write

$$\widetilde{M}_t(\mathrm{d}x_t \mid x_{0:t-1}, u_{0:T}) = \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2} I\right) \mathrm{d}x_t \tag{4.46}$$

and that, as soon as M_t has a density, which we identify with the measure itself,

$$\begin{aligned} \pi_T(\mathrm{d}x_{0:T} \mid u_{0:T}) &\propto \prod_{t=0}^{T} M_t(\mathrm{d}x_t \mid x_{0:t-1}) G_t(x_{0:t}) \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2} I\right), \\ &= \prod_{t=0}^{T} \widetilde{M}_t(\mathrm{d}x_t \mid x_{0:t-1}, u_{0:T}) \left\{ \underbrace{G_t(x_{0:t}) M_t(x_t \mid x_{0:t-1})}_{\widetilde{G}_t(x_{0:t})} \right\}, \end{aligned} \tag{4.47}$$

so that we can sample from $\pi_T(\mathrm{d}x_{0:T}, \mathrm{d}u_{0:T})$ by running the CSMC algorithm of Section 4.3.2 with a proposal kernel $\widetilde{M}_t$ centred around u_t, and a potential function $\widetilde{G}_t$ that does not depend on $u_{0:T}$. This is the idea behind particle-RWM: we use the additional observations $u_{0:T}$ to localise the model, and then use the CSMC algorithm with the modified proposal kernels to sample from the resulting distribution[15].

Finke and Thiery (2023) show that, when the target $\pi_T(\mathrm{d}x_{0:T}, \mathrm{d}u_{0:T})$ is regular enough, the particle-RWM algorithm recovers the classical D^{-1} scaling of the random walk Metropolis algorithm, where D is the dimension of the state space. That is, when $\delta_t = O(1/D)$, and independently of the number of particles, the resulting Markov kernel is stable and does not suffer from the

[15]This is not the way the construction was originally introduced in Finke and Thiery (2023), but comes from Publication V. Because it is dramatically simpler (in our opinion) than the former, and that it allows to understand Publications V and VII more easily, we elect to present the algorithm here this way.

curse of dimensionality. Finally, we note that similar ideas have been used in the context of particle MCMC (Fearnhead and Meligkotsidou, 2016; Karppinen and Vihola, 2021) to construct efficient MCMC kernels for the state, but also for the parameters of the model. However, these works have mostly been interested in the converse problem of having to deal with uninformative observations, and therefore the augmentation was meant as a way to increase the informativeness of the observations, rather than decrease it as in the case of particle-RWM.

Remark 4.22. *In Publications V,VII we make explicit use of the auxiliary variable $u_{0:T}$ to generalise the particle-RWM algorithm to include both prior and gradient information, and show that the resulting algorithms improve upon the performance of the original particle-RWM algorithm.*

5. Topics in parallel inference

In this chapter, we introduce several topics related to parallel inference. This is a rather broad field, and we therefore limit ourselves to a few topics that are of particular interest to us, and that should be of help in understanding the rest of the thesis. We start by providing a brief overview of what we *mean* by parallel inference, and then discuss several instances of parallelism in the context of state-space models inference.

The topics of this chapter are of particular interest for Publications II,III,IV,V.

5.1 Introduction

In this chapter, we provide a brief overview of the general state of the literature. Our main focus is on two types of parallelism: computational parallelism, where the parallelism happens at the level of the computations, keeping the method unchanged, and statistical parallelism, where the parallelism happens at the level of the method, allowing for more efficient computations often at the cost of statistical efficiency.

5.1.1 Computational parallelism

Computational parallelism is probably the most common form of parallelism in statistics or numerical computation in general, and is often the first type of parallelism that is considered when developing a new algorithm. This can be understood as a *method first* approach, where the statistician starts with a given procedure they want to implement, and then considers which parts of the procedure are amenable to a form of parallelism or another.

Opportunities for parallelism are often found in the form of *embarrassingly parallel* computations, where the computations are independent of each other, and can therefore be run in parallel with no communication required between them. A typical example of this is the computation of the mean of a large dataset, where the data is split into several chunks, and the mean of each chunk is computed independently, and then only combined to produce the final result.

This type of parallelism is frequent in the context of data parallelism, where the data is split into several parts that are then processed independently, and has led to the development of several frameworks for distributed computing, such as Apache Spark (Zaharia et al., 2016) or OpenMP (Chandra et al., 2001). Another example of computational parallelism is the standard Monte Carlo integration, where the integral is approximated by a sample average:

$$\int f(x)\pi(\mathrm{d}x) \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{5.1}$$

where $x_1, \dots, x_N$ are independent samples from π (see Chapter 2). Then it is possible to split the sum in (5.1) into several parts, and compute each part independently, and then combine the results to produce the final result. This is the most common type of parallelism in the context of Monte Carlo integration, where the samples are generated independently, and is often what is meant when one says Monte Carlo simulation is *embarrassingly parallel*.

Additionally to these, and thanks to the advances of parallel hardware such as general-purpose graphics processing units (GPUs), other forms of parallelism make it possible to speed up computations that were previously considered too slow. For example, standard inference in Gaussian processes (see, e.g., Rasmussen and Williams, 2006) is now routinely sped up by using GPU matrix-matrix multiplication and matrix inversion routines, which were previously considered too slow to be used in practice beyond a few hundred data points. This use of fast matrix multiplication routines is also at the centre of the success of deep learning (see, e.g., Goodfellow et al., 2016), where the use of GPUs has allowed for the efficient training of neural networks on large datasets.

Remark 5.1. *We, too, make use of these accelerated routines specifically in Publication I to speed up the computation of an optimal transport plan (see Section 3.4.2). They are also used implicitly, but less critically, in our other publications treating of parallelism: Publications II,III,IV,V.*

5.1.2 Statistical parallelism

Statistical parallelism is a different paradigm altogether, where the statistician starts from the very principle of parallelism compatibility and then builds a method for their problem under this constraint. This can be understood as a *hardware first* approach, sometimes sacrificing statistical efficiency on the altar of computational efficiency.

Typical examples of this paradigm are the use of debiasing techniques such as perfect simulation, simulated tempering, or coupling of Markov chains or multi-level Monte Carlo. Fundamentally, these methods consist in taking biased but efficient procedures, and then debiasing them to produce an unbiased estimator of the quantity of interest.

Perfect simulation: Perfect simulation (Huber, 2016) consists in sampling exactly from a target distribution π, which is often intractable. Several methods have been developed to solve this problem, the most famous one being coupling from the past (Propp and Wilson, 1996), and its extensions (Corcoran and Tweedie, 2002). These methods consist in running several dependent Markov chains in parallel, which are designed so that they eventually coalesce into a single chain that is then distributed according to π. This method then allows for the exact sampling from π, and therefore embarrassingly parallel Monte Carlo simulation at the cost of a potentially large number of iterations before the chains coalesce.

Simulated tempering: Simulated tempering or annealing (Marinari and Parisi, 1992; Geyer and Thompson, 1995) is a method for sampling from a target distribution π by introducing an auxiliary variable β that controls the temperature of the system. The chain is started at temperature $\beta_0 = 0$, and samples from the distribution are achieved at $\beta_L = 1$. While not often used in practice, due to the difficulty of choosing the sequence of temperatures $\beta_0, \ldots, \beta_L$, simulated tempering allows for rejuvenation of the chain when it is at temperature β_0 by sampling from a reference distribution π_0. This means that the chain can be simulated independently for each cycle going from $\beta_0 = 0$ back to $\beta_0 = 0$, and then combined to produce the final result.

Unbiased MCMC with couplings: Unbiased MCMC with couplings (Glynn and Rhee, 2014; Jacob et al., 2020b) is a method for debiasing MCMC estimators by means of telescoping sums

$$\varphi(X_0) + \sum_{k=1}^{\infty} \varphi(X_k) - \varphi(Y_{k-1}), \tag{5.2}$$

where $(X_k)_{k=0}^{\infty}$ and $(Y_k)_{k=0}^{\infty}$ are identically distributed Markov chains with stationary distribution π, and are coupled such that $X_k = Y_{k-1}$ for all $k > \tau$, where τ is a random variable. When τ is finite, this estimator of $\pi(\varphi)$ can be computed and, under additional conditions is unbiased, thereby allowing for parallel computation by running independent realisations of the sum. This method has also been used in the context of multilevel Monte Carlo (MLMC, Vihola, 2018; Rhee and Glynn, 2015; Chada et al., 2021) and SMC (Jacob et al., 2020a; Middleton et al., 2019).

All the examples above allow for "infinite" parallelism, in the sense that given an infinite number of computers, the performance of the algorithm can be improved indefinitely. This is not the case, for example, of parallel tempering (Geyer, 1991), which, similarly to simulated tempering, introduces an auxiliary variable β that controls the temperature of the system. In contrast to simulated tempering, rather than a single chain, parallel tempering runs several chains in parallel, each at a different temperature, and then exchanges the states (or equivalently the temperatures) of the chains at regular intervals. This specific algorithm is sometimes referred to as *replica exchange* (Swendsen and Wang, 1986), and differs from the other methods in that it improves the mixing of the chain (in particular for multi-modal posterior distributions, Neal, 1996), and therefore improves the statistical efficiency of the estimator rather than just its computational efficiency. This is a key difference, and is the reason why parallel tempering is more often used in practice than other alternatives (see, e.g., The Event Horizon Telescope Collaboration, 2019, for a well-known application to cosmology). On the other hand, parallel tempering suffers from asymptotic inefficiency, in the sense that its efficiency plateaus (and sometimes even degrades) as the number of parallel chains increases, and therefore requires a careful tuning of the number of chains to be used. Some recent developments in the field of parallel tempering include the use of asynchronous parallel tempering (Marie d'Avigneau et al., 2021), which allows for the independent chains to be run at different *computational* speed, and therefore allows using MCMC kernels with random runtimes (such as Hoffman et al., 2014) within; and the use of non-reversible moves on the temperature ladder (Syed et al., 2022), which allows for the use of a larger number of chains, and therefore improves the efficiency of the algorithm. Finally, we note that some interesting efforts have been made to combine parallel tempering with the debiasing techniques described above (Biswas et al., 2019).

5.2 Prefix-sums and parallel inference in state-space models

In this section, we present an instance of *computational parallelism* in the context of state-space models. Two cases are considered: (i) the parallelisation of inference along the time dimension, and (ii) the parallelisation of inference across simulations in SIR. The main idea behind this parallelisation is the use of prefix-sums, which is a common technique in parallel computing and which we introduce in Section 5.2.1.

5.2.1 Prefix-sums

Prefix-sums are a common technique in parallel computing Blelloch (1990); Hillis and Steele Jr (1986) for obtaining the running sum of a sequence of numbers. Formally, suppose that we are given an associative binary operator $\oplus$ and a sequence of elements $x_1, \dots, x_N$. Then the prefix-sums of the sequence $x_1, \dots, x_N$ are defined as the sequence $y_1, \dots, y_N$ where

$$y_k = \bigoplus_{i=1}^{k} x_i. \tag{5.3}$$

When the hardware at hand is a CPU, the prefix-sums can be computed in $\mathcal{O}(N)$ time by means of a simple for-loop. However, when the hardware at hand allows

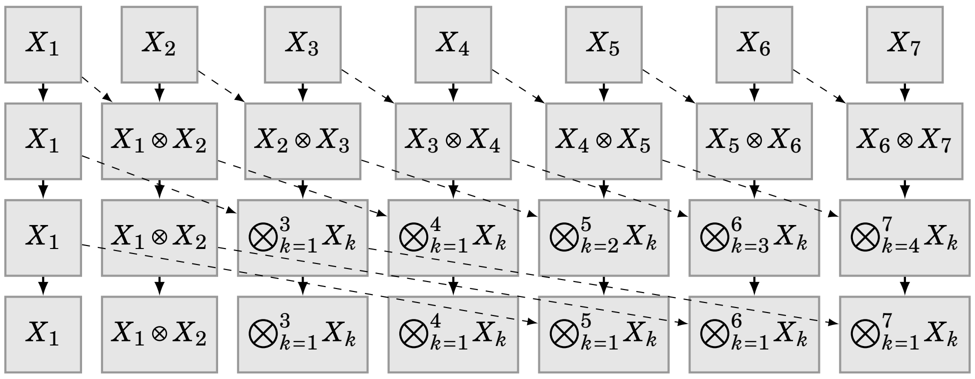

X_1
X_2
X_3
X_4
X_5
X_6
X_7
$X_1 \otimes X_2$
$X_2 \otimes X_3$
$X_3 \otimes X_4$
$X_4 \otimes X_5$
$X_5 \otimes X_6$
$X_6 \otimes X_7$
$\bigotimes_{k=1}^{3} X_k$
$\bigotimes_{k=1}^{4} X_k$
$\bigotimes_{k=2}^{5} X_k$
$\bigotimes_{k=3}^{6} X_k$
$\bigotimes_{k=4}^{7} X_k$
$\bigotimes_{k=1}^{5} X_k$
$\bigotimes_{k=1}^{6} X_k$
$\bigotimes_{k=1}^{7} X_k$

Figure 5.1. Illustration of prefix-sums.

for parallel computation and memory access, the prefix-sums can be computed in $\mathcal{O}(\log N)$. A procedure to compute the prefix-sums in $\mathcal{O}(\log N)$ is illustrated in Figure 5.1. In this figure, the prefix-sum is computed in place, where the array containing the elements $x_1, \dots, x_N$ is modified at each step. The algorithm illustrated therein is known as the *Hillis–Steele* algorithm (Hillis and Steele Jr, 1986), and is one of the simplest algorithms to compute the prefix-sums. It proceeds by combining the elements of the array in pairs, and then repeating the process until the prefix-sums are computed, pairing elements of doubling distance at each step, resulting in a total of $\log_2 N$ steps.

This algorithm can be easily parallelised, where each pairwise operation is computed in parallel, and then the results are recursively combined to produce the final result. This ensures that the total number of *embarrassingly parallel* step required to compute the prefix-sums is $\log_2 N$, and therefore that the algorithm has *span complexity* $\mathcal{O}(\log N)$, where the span complexity, defined as the total number of non-parallel steps is named after the span of a tree, which is the longest path from the root to a leaf. Other algorithms exist to compute the prefix-sums, such as the *Blelloch* algorithm (Blelloch, 1989), which has span complexity $\mathcal{O}(\log N)$ and work complexity $\mathcal{O}(N)$, but offers less parallelism than the Hillis-Steele algorithm. For a more detailed overview of the different algorithms to compute the prefix-sums and their practical implications, we refer the reader to Pibiri and Venturini (2021).

Because this concept is central to the rest of this chapter, we provide an illustrative pseudocode for the Hillis-Steele procedure in Algorithm 11.

5.2.2 Parallel resampling in the particle filter

Perhaps the earliest and easiest application of prefix-sums in the context of state-space models is the parallelisation of the resampling step in the particle filter. This question emerged in the literature around the same time as early works considering the use of GPUs for Bayesian inference (Manavski and Valle, 2008; Suchard and Rambaut, 2009), and was for example considered in Maskell

Algorithm 11: Illustrative implementation of the Hillis-Steele algorithm.

```
input  : Elements $x_1, \dots, x_N$, operator $\oplus$.
output : Prefix-sums $\bigoplus_{i=1}^{n} x_i$, for $n = 1, \dots, N$.
1 ▷ The routine modifies the input array in place.
2 for $d \leftarrow 0$ to $\lfloor \log_2 N \rfloor$ do
3     for $i \leftarrow N-1$ to $0$ do
4         ▷ For each $i$, in parallel.
5         if $i - 2^d \geq 0$ then
6             $x_i \leftarrow x_{i-2^d} \oplus x_i$
```

et al. (2006); Lee et al. (2010); Murray (2012, see also references within), where they are used to speed up particle filtering in its entirety. Most of the steps in SIR (aside from the time dimension) are in fact embarrassingly parallel, where all particles are handled independently, and therefore can be run in parallel. However, the resampling step is not embarrassingly parallel as it

1. requires the normalisation of the weights $W_t^n = \frac{G_t(x_{0:t}^{(n)})}{\sum_{m=1}^N G_t(x_{0:t}^{(m)})}$, which requires the computation of the sum $\sum_{n=1}^N G_t(x_{0:t}^{(n)})$;
2. requires the sampling from the joint distribution of the ancestors indices conditionally on the weights;
3. and finally requires the duplication of the particles according to the sampled indices.

In order to understand how the resampling step can be parallelised, let us come back to its definition, which we had written in terms of its probability mass function in Chapter 3. We first define the inverse CDF of the categorical distribution.

Definition 5.2 (Inverse CDF of a categorical distribution)**.** *Let*

$$\eta^N(\mathrm{d}v) = \sum_{n=1}^{N} W_t^n \mathbb{1}_n(\mathrm{d}v)$$

be a categorical distribution on $\{1, \dots, N\}$*, with its inverse CDF defined as*

$$\mathscr{F}^-(u) = \inf\left\{ j \geq 1 \mid \sum_{n=1}^{j} W_t^n \geq u \right\}.$$

The following proposition links the inverse CDF of the categorical distribution to unbiased resampling schemes.

Proposition 5.3 (Resamplings as a pushforward of the inverse CDF)**.** *Let* $U_{1:N} \sim \mathscr{S}$ *be a joint distribution over* $[0,1]^N$ *such that a randomly chosen* U_K*, with* $K \sim \mathscr{U}(\{1, \dots, N\})$*, is marginally uniform on* $[0,1]$*, then* $\mathscr{R}(\cdot \mid (W_t^n)_{n=1}^N) = \mathscr{F}_\#^- \mathscr{S}$ *is an unbiased resampling scheme.*

Proof. This follows from checking that the offspring distribution of $\mathscr{R}$ verifies the conditions of Definition 3.30.

$$\mathscr{O}^m = \sum_{n=1}^{N} \mathbb{1}\left(\sum_{j=1}^{m} W_t^j \geq U_n \geq \sum_{j=1}^{m-1} W_t^j\right),$$

so that, by summing disjoint sets, $\sum_{m=1}^{N} \mathscr{O}^m = N$, and, for $K \sim \mathscr{U}(\{1,\dots,N\})$,

$$\begin{aligned}\mathbb{E}\left[\mathscr{O}^m\right] &= N\mathbb{E}_K\left[\mathbb{1}\left(\sum_{j=1}^{m} W_t^j \geq U_K \geq \sum_{j=1}^{m-1} W_t^j\right)\right],\\ &= NW_t^m.\end{aligned}$$

□

For example, standard resamplings are obtained as the following distribution $\mathscr{S}$ over $[0,1]^N$:

Stratified: $\mathscr{S}(u_{1:N}) = \prod_{n=1}^{N} \mathscr{U}\left(u_n; \left[\frac{n-1}{N}, \frac{n}{N}\right]\right)$,

Systematic: $\mathscr{S}(\mathrm{d}u_{1:N}) = \mathscr{U}(u_1; [0, 1/N]) \prod_{n=2}^{N} \delta_{u_1+(n-1)/N}(\mathrm{d}u_n)$.

This formulation of the resampling step allows for the use of prefix-sums to compute the resampling step. This is illustrated in Algorithm 12. In this

Algorithm 12: Resampling

```
input  : Normalised weights $W_t^n \propto G_t(x_{0:t}^{(n)})$ for $n = 1,\dots,N$, $U^{1:N} \sim \mathscr{S}$ for
         an unbiased resampling scheme $\mathscr{S}$.
output : Resampled indices $A_t^n$ for $n = 1,\dots,N$.
1 Compute the prefix-sum $s^n = \sum_{m=1}^{n} W_t^m$, $n = 1,\dots,N$
2 Set $S^n = s^n / s^N$ for $n = 1,\dots,N$
3 for $n = 1,\dots,N$ do
4     ▷ This can be done in parallel
5     Set $A_t^n = \textsc{BinarySearch}(U^n, S^{1:N})$
```

algorithm, the first step is to compute the prefix-sums of the weights W_t^n, which can be done in $\mathscr{O}(\log N)$ on a parallel hardware. Then, the prefix-sums are normalised to produce the cumulative distribution function of the weights $S^n = \sum_{m=1}^{n} W_t^m$, which can be done in $\mathscr{O}(1)$ in parallel. The indices A_t^n are sampled from the inverse CDF of the categorical distribution, by doing, for example, independent binary searches for each U^n, each of which can be done in $\mathscr{O}(\log N)$ in the worst case, fully in parallel.

Numerical stability issues can arise when computing the prefix-sums, especially when a large number of particles, thereby making the weights small. While this can be alleviated by working with the logarithm of the weights, this issue is inherent to the use of floating point numbers, and is not specific to the use of prefix-sums. This remark prompted the development of alternative resampling schemes that do not require the normalisation of the weights (Murray, 2012; Murray et al., 2016) but instead define alternative sampling schemes that only rely on comparing ratios of weights, such as MCMC chains on the space of ancestors. Finally, we note that, over the course of the past decade, several other parallel resampling schemes have been developed, some of them more amenable to *distributed* architectures (Lee and Whiteley, 2016; Varsi et al., 2021), and some others tackling asynchronous parallelism (typically by using Poisson particle birthing processes, see, e.g., Tomasz Cakala and Niemiro, 2021). However, these are beyond the scope of this thesis, and we refer the reader to the original papers for more details.

Remark 5.4. *These concepts, and subsequent improvements in Murray et al. (2016), are used in Publication IV to perform efficient resampling for large matrices $N \times N$ (see Section 5.3).*

5.2.3 Associative filtering and smoothing

We now turn to the parallelisation of inference in state-space models along the time dimension. This is a largely different problem from the parallelisation across simulations in SIR, which maintained a sequential dependency in the time dimension. In contrast, the parallelisation along the time dimension means to remove this sequential dependency altogether by formulating the whole inference procedure, presented in Chapter 3, as a prefix-sum. This is the approach taken in Särkkä and García-Fernández (2021); Hassan et al. (2021), which we present in this section. For the sake of generality, but also for coherence with the rest of the thesis, we take a more general perspective than the one presented in the articles we base our work on, and consider the abstract formulation of the filtering and smoothing for Markovian models given as in (3.8):

$$\mathrm{d}\pi_t(x_{0:t}) \propto M_0(\mathrm{d}x_0) G_0(x_0) \prod_{s=0}^{t-1} M_{s+1}(\mathrm{d}x_{s+1} \mid x_s) \prod_{s=1}^{t} G_s(x_s, x_{s-1}). \tag{5.4}$$

We can now consider the family $\mathscr{F}$ of pairs (F, H) such that

$$F : \begin{cases} \mathrm{X} \longrightarrow \mathscr{P}(\mathscr{Y}) \\ x \longmapsto F(\mathrm{d}y \mid x) \end{cases} \qquad H : \begin{cases} \mathrm{X} \longrightarrow \mathbb{R}^+ \\ x \longmapsto H(x) \end{cases} \tag{5.5}$$

where $F(\mathrm{d}y \mid x)$ represents Markov kernels on X, and $H(x)$ is a positive function on X, representing normalising constants of the Markov kernels $F(\mathrm{d}y \mid x)$.

Definition 5.5 (Filtering operator)**.** *Let $\mathscr{F}$ be the family of pairs (F, H) as in (5.5). Then the filtering operator $\otimes$ is defined on $\mathscr{F}$ by $(F_i, H_i) \otimes (F_j, H_j) := (F_{ij}, H_{ij})$*

with

$$F_{ij}(\mathrm{d}z \mid x) := \frac{\int_{y \in \mathsf{X}} H_j(y) F_j(\mathrm{d}z \mid y) F_i(\mathrm{d}y \mid x)}{\int_{y \in \mathsf{X}} H_j(y) F_i(\mathrm{d}y \mid x)}, \tag{5.6a}$$

$$H_{ij}(z) := H_i(z) \int_{y \in \mathsf{X}} H_j(y) F_i(\mathrm{d}y \mid z). \tag{5.6b}$$

Proposition 5.6. *The operator $\otimes$ is associative.*

Proof. Let (F_i, H_i), (F_j, H_j) and (F_k, H_k) be three elements of $\mathscr{F}$. Then

$$\begin{aligned} H_{ijk}(z) &= H_{ij}(z) \int_{y \in \mathsf{X}} H_k(y) F_{ij}(\mathrm{d}y \mid z) \\ &= \left\{ H_i(z) \int_{y \in \mathsf{X}} H_j(y) F_i(\mathrm{d}y \mid z) \right\} \int_{y \in \mathsf{X}} H_k(y) \frac{\int_{y' \in \mathsf{X}} H_j(y') F_j(\mathrm{d}y \mid y') F_i(\mathrm{d}y' \mid z)}{\int_{y' \in \mathsf{X}} H_j(y') F_i(\mathrm{d}y' \mid z)} \\ &= H_i(z) \int_{y' \in \mathsf{X}} \left\{ F_i(\mathrm{d}y' \mid z) H_j(y') \int_{y \in \mathsf{X}} F_j(\mathrm{d}y \mid y') H_k(y) \right\} \\ &= H_i(z) \int_{y' \in \mathsf{X}} H_{jk}(y') F_i(\mathrm{d}y' \mid z), \end{aligned}$$

proving associativity of the operator $\otimes$ for the H part. The proof for the F part is similar. □

Remark 5.7. *A natural way to interpret the associativity of the operator $\otimes$ is to read it as a form weighted Chapman–Kolmogorov equation (see,e.g., Douc et al., 2018, Chap. 1), where the weights are given by the normalising functions H_i. Indeed, in the case where the H_i terms are constant equal to 1, the operator $\otimes$ reduces to the usual Chapman–Kolmogorov equation which is clearly associative.*

It now remains to define the filtering operation as a specific instance of the family $\mathscr{F}$. Thankfully, this is straightforward, and the filtering operation is given by the following proposition.

Proposition 5.8 (Filtering as an associative operator)**.** *Consider the family*

$$H_t(x_{t-1}) := \int G_t(x_{t-1:t}) M_t(\mathrm{d}x_t \mid x_{t-1}), \tag{5.7}$$

$$F_t(\mathrm{d}x_t \mid x_{t-1}) := \frac{M_t(\mathrm{d}x_t \mid x_{t-1}) G_t(x_{t-1:t})}{H_t(x_{t-1})}, \tag{5.8}$$

with by convention

$$H_0(x_{-1}) := \int G_0(x_0) M_0(\mathrm{d}x_0), \tag{5.9}$$

$$F_0(\mathrm{d}x_0 \mid x_{-1}) := \frac{G_0(x_0) M_0(\mathrm{d}x_0)}{H_0(x_{-1})}. \tag{5.10}$$

Then the pair $\pi_t(\mathrm{d}x_t), L_t$ is given by

$$\left(\pi_t(\mathrm{d}x_t) \quad L_t\right) = \bigotimes_{s=0}^{t} (F_s, H_s). \tag{5.11}$$

where the left hand side has to be understood as a shorthand for

$$\pi_t(\mathrm{d}x_t \mid x_{-1}), \quad L_t(x_{-1}), \tag{5.12}$$

which do not depend on $x_{-1} = \emptyset$.

Proof. Given that the operator $\otimes$ is associative, it suffices to prove that the sequential application of the operator $\otimes$ to the family $(F_s, H_s)_{s=0}^t$ is equal to the filtering operation. For $t = 0$, the result is true. If this is true up to time $t - 1 \geq 0$, then write

$$\bigotimes_{s=0}^{t} (F_s, H_s) = \left(\widetilde{F}(\mathrm{d}x_t \mid \alpha), \quad \widetilde{H}(\beta)\right), \tag{5.13}$$

where,

$$\begin{aligned}
\widetilde{H}(\beta) &= L_{t-1} \int_{x_{t-1} \in \mathrm{X}} H_t(x_{t-1}) \pi_{t-1}(\mathrm{d}x_{t-1}) \\
&= L_{t-1} \int_{x_{t-1} \in \mathrm{X}} \int_{x_t \in \mathrm{X}} G_t(x_{t-1:t}) M_t(\mathrm{d}x_t \mid x_{t-1}) \pi_{t-1}(\mathrm{d}x_{t-1}) \\
&= L_{t-1} \iint_{\mathrm{X} \times \mathrm{X}} G_t(x_{t-1:t}) M_t(\mathrm{d}x_t \mid x_{t-1}) \pi_{t-1}(\mathrm{d}x_{t-1}) \\
&= L_{t-1} \iint_{\mathrm{X} \times \mathrm{X}} G_t(x_{t-1:t}) \pi_{t-1}(\mathrm{d}x_{t-1:t}) \quad \text{(Extension)} \\
&= L_{t-1} \ell_t = L_t.
\end{aligned}$$

and we have, noting that α disappears from the expression,

$$\begin{aligned}
\widetilde{F}(\mathrm{d}x_t \mid \alpha) &= \frac{\int_{x_{t-1} \in \mathrm{X}} H_t(x_{t-1}) F_t(\mathrm{d}x_t \mid x_{t-1}) \pi_{t-1}(\mathrm{d}x_{t-1})}{\int_{x_{t-1} \in \mathrm{X}} H_t(x_{t-1}) \pi_{t-1}(\mathrm{d}x_{t-1})} \\
&\propto \int_{x_{t-1} \in \mathrm{X}} M_t(\mathrm{d}x_t \mid x_{t-1}) G_t(x_{t-1:t}) \pi_{t-1}(\mathrm{d}x_{t-1}) \\
&= \int_{x_{t-1} \in \mathrm{X}} G_t(x_{t-1:t}) \pi_{t-1}(\mathrm{d}x_{t-1:t}) \quad \text{(Extension)} \\
&\propto \int_{x_{t-1} \in \mathrm{X}} \pi_t(\mathrm{d}x_{t-1:t}) \quad \text{(Change of measure)} \\
&= \pi_t(\mathrm{d}x_t) \quad \text{(Marginalisation).}
\end{aligned}$$

And the result follows by induction. This presentation differs from the one in Särkkä and García-Fernández (2021), in particular the proof therein is done by considering the more complicated partial prefix-sums $\bigotimes_{t=r}^{s}(F_t,H_t)$, which may seem more general but is in fact equivalent to the one presented here. We also consider the case of bivariate potentials, slightly more general than the likelihood terms considered in Särkkä and García-Fernández (2021). □

Proposition 5.8 allows for the parallelisation of the filtering step, by means of the use of prefix-sums, however, it is not clear how to actually implement this in practice:

1. it is not possible to represent arbitrary Markov kernels and functions on X in a computer, and therefore the full family $\mathscr{F}$ is not computable;
2. even if the family $\mathscr{F}$ was computable, the integrations in (5.6a) and (5.6b) are not tractable in general, and therefore the filtering operator ⊗ cannot be computed in practice.

Nonetheless, at least two special cases of the filtering operator ⊗ are tractable and can be computed in practice, namely the case where the model (5.4) is a state-space model, linear and Gaussian, and the case where the model (5.4) is a hidden Markov model (i.e., when X is finite).

Hidden Markov models: $\mathbb{X} = \{1,\dots,K\}$**.** In this case, the operands of the filtering operator ⊗ are given by

$$\begin{aligned}
H_t(i) &:= \sum_{j=1}^{K} G_t(x_t = j, x_{t-1} = i) M_t(x_t = j \mid x_{t-1} = i), \\
F_t(j \mid i), &:= \frac{M_t(x_t = j \mid x_{t-1} = i) G_t(x_t = j, x_{t-1} = i)}{H_t(i)}, \\
H_0(i) &:= \sum_{j=1}^{K} G_0(x_0 = i) M_0(x_0 = i), \\
F_0(i \mid j) &:= \frac{1}{H_0(j)} p(x_0) p(y_0 \mid x_0 = j),
\end{aligned} \tag{5.14}$$

and the filtering operator ⊗ is given by

$$\begin{aligned}
(F_i,H_i) \otimes (F_j,H_j) =: (F_{ij},H_{ij}), \qquad F_{ij}(m \mid k) &:= \frac{\sum_{l=1}^{K} H_j(l) F_i(m \mid l) F_j(l \mid m)}{\sum_{l=1}^{K} H_j(l) F_i(l \mid k)}, \\
H_{ij}(k) &:= H_i(k) \sum_{l=1}^{K} H_j(l) F_i(l \mid k),
\end{aligned} \tag{5.15}$$

so that these can be represented and computed exactly. This forms the basis of Hassan et al. (2021); Särkkä and García-Fernández (2023).

Linear Gaussian state-space models. In the case where the model (5.4) is a state-space model, linear and Gaussian, the operands of the filtering operator $\otimes$ are given by

$$\begin{aligned} H_t(x_{t-1}) &= p(y_t \mid x_{t-1}) \propto \mathscr{N}_I(x_{t-1}; \eta_t, J_t), \\ F_t(x_t \mid x_{t-1}) &= \mathscr{N}(x_t, A_t x_{t-1} + u_t, C_t), \\ p(x_0 \mid x_{-1}) &= \mathscr{N}(x_0, I m_{-1} + 0, 0), \end{aligned} \tag{5.16}$$

where $\mathscr{N}_I(\eta_t, J_t)$ is the Gaussian distribution given in its information form: its precision vector is η_t and its precision matrix J_t. The coefficients A_t, u_t, C_t, η_t, and J_t can all be computed in closed form from the parameters of the model, using standard Gaussian algebra, so that the filtering operands have a finite-dimensional representation and can be stored in a computer.

The filtering operator $\otimes$ is given by

$$(F_i, H_i) \otimes (F_j, H_j) =: (F_{ij}, H_{ij}), \qquad \begin{aligned} F_{ij}(z \mid x) &:= \mathscr{N}(z; A_{ij} x + b_{ij}, C_{ij}), \\ H_{ij}(x) &:= \mathscr{N}_I(x; \eta_{ij}, J_{ij}), \end{aligned} \tag{5.17}$$

where

$$\begin{aligned} A_{ij} &= A_j (I + C_i J_j)^{-1} A_i, \\ b_{ij} &= b_j + A_j (I + C_i J_j)^{-1} (b_i + C_i \eta_j), \\ C_{ij} &= C_j + A_j (I + C_i J_j)^{-1} C_i A_j^\top \\ \eta_{ij} &= \eta_i + A_i^\top (I + J_j C_i)^{-1} (\eta_j - J_j b_i), \\ J_{ij} &= J_i + A_i^\top (I + J_j C_i)^{-1} J_j A_i. \end{aligned} \tag{5.18}$$

For more details on the derivation of these equations, we refer the reader to Särkkä and García-Fernández (2021).

Smoothing. We now turn to the parallelisation of the smoothing step, which is a simpler problem than the parallelisation of the filtering step as it does not involve the computation of marginalisation constants. In fact, the smoothing step essentially follows directly from the associativity of Chapman–Kolmogorov equation without any re-normalisation. Contrary to the filtering case, we therefore only write the general associative formulation of the smoothing step, and do not consider any specific instance of the family $\mathscr{F}$. Note however that the elements F_t appearing in Proposition 5.12 below can be derived from Proposition 3.18.

Definition 5.9 (Smoothing operator)**.** *Let $\mathscr{M}$ be the family of Markov kernels $M : \mathrm{X} \to \mathscr{P}(\mathrm{X})$. Then the smoothing operator $\oplus$ is defined on $\mathscr{M}$ by $F_i \oplus F_j := F_{ij}$ with*

$$F_{ij}(\mathrm{d}z \mid x) := \int F_j(\mathrm{d}y \mid x) F_i(\mathrm{d}z \mid y). \tag{5.19a}$$

Remark 5.10. *Note how the Chapman–Kolmogorov equation is taken backwards in the definition of the smoothing operator $\oplus$, where we would have expected the following definition:*

$$F_{ij}(\mathrm{d}z \mid x) := \int F_i(\mathrm{d}y \mid x) F_j(\mathrm{d}z \mid y). \tag{5.20}$$

Proposition 5.11. *The operator $\oplus$ is associative.*

We can now define the smoothing operands as a specific instance of the family $\mathscr{M}$.

Proposition 5.12 (Smoothing as an associative operator). *Consider the family*

$$F_t(\mathrm{d}x_t \mid x_{t+1}) := \pi_T(\mathrm{d}x_t \mid x_{t+1}), \tag{5.21}$$

with, by convention, $F_T(\mathrm{d}x_T \mid x_{T+1}) := \pi_T(\mathrm{d}x_T)$, then

$$\pi_T(\mathrm{d}x_t) = \bigoplus_{s=T}^{t} F_s. \tag{5.22}$$

Proof. Given that the operator $\oplus$ is associative, it suffices to prove that the sequential application of the operator $\oplus$ to the family $(F_s)_{s=T}^{t}$ is equal to the smoothing operation. For $t = T$, the result is trivially true. If this is true up to time $t + 1 \leq T$, then write

$$\begin{aligned} \bigoplus_{s=T}^{t} F_s &= \int_{x_{t+1}} \pi_T(\mathrm{d}x_t \mid x_{t+1}) \pi_T(\mathrm{d}x_{t+1}), \\ &= \pi_T(\mathrm{d}x_t), \end{aligned}$$

where the equality follows from the fact that $\mathrm{d}\pi_T(x_{t:t+1}) = \pi_T(\mathrm{d}x_t \mid x_{t+1}) \pi_T(\mathrm{d}x_{t+1})$. And the result follows by induction. □

Remark 5.13. *We make use of these parallel operators liberally in Publications II, III, and V, the latter two introducing some modifications to the smoothing step.*

5.2.4 Extensions

The parallelisation of the filtering and smoothing steps in state-space models has been extended in several directions.

Linear quadratic control: the problem of computing the optimal control of a linear quadratic system has been shown to be equivalent to the problem of inference in a linear Gaussian state-space model (Kalman, 1960). As a result, the parallelisation of the smoothing step in linear Gaussian state-space models can be used to compute the optimal control of a linear quadratic system in parallel too (Särkkä and García-Fernández, 2023).

Viterbi algorithm: the Viterbi algorithm is a dynamic programming algorithm that computes the MAP of the distribution π_T when the model is a hidden Markov model (Forney, 1973). The recursion of the Viterbi algorithm can also be rewritten as an associative operator, and therefore parallelised using prefix-sums (Hassan et al., 2021; Särkkä and García-Fernández, 2023).

Tree prefix-sums: recent work has shown that the prefix-sums can be parallelised on a tree architecture, where the tree is defined by the dependencies between the operands of the operator (Gupta et al., 2023). This opens the door to the parallelisation of inference on directed acyclic graphs, such as the ones that arise in the context of probabilistic programming (see, e.g., Koller and Friedman, 2009).

5.3 Divide-and-conquer SMC

Another line of work that has been developed in the context of parallel inference is the *divide-and-conquer* approach, which consists in splitting the inference problem into smaller sub-problems, solving them in parallel, and then combining the results to produce the final result. This approach has been applied to SMC in Lindsten et al. (2017), and we present it in this section.

5.3.1 A recursive formulation of SIR

In Section 3.4.1, we have presented the SIR algorithm as a sequential algorithm, where the representing π_t are propagated sequentially from the representation of π_{t-1}. In this section, we take a step back, and rewrite the SIR algorithm in a recursive fashion, which will be useful to present the divide-and-conquer approach of Lindsten et al. (2017). To do so, it is useful to think of the "resampling-propagate-reweight" procedure in terms of being a single function, applied to the representation of π_{t-1}. The algorithm then proceeds as follows: the case $t = 0$ corresponds to the leaf of the recursion, and is handled by the base case of the recursion, which samples the particles from the prior distribution M_0 and computes the normalised weights. Then, for $t \geq 1$, the algorithm proceeds by first calling itself recursively on the representation of π_{t-1}, and then applying the resampling-propagate-reweight procedure to the result of the recursive call. This perspective of SIR is given in Algorithm 13.

Markovian models as given in (3.8), to which Algorithm 13 applies to, correspond to the simplest possible case of directed acyclic graphs, where the graph is a chain $\ldots \to t \to t+1 \to \ldots \to T$. Other models can however be of interest, such as the ones that arise in probabilistic programming, where the graph is a tree, or more generally a directed acyclic graph. An example is given in Figure 5.2, where the graph corresponds to the hierarchical Bayesian model given in (4.8).

Algorithm 13: Particle filtering as a recursive algorithm

```
1  Function SIR($t$):
2      if $t = 0$ then
3          Sample $X_0^1, \dots, X_0^N \sim M_0(\mathrm{d}x_0)$
4          Compute the normalised weights $W_0^1, \dots, W_0^N$
5          return $\{X_0^n, W_0^n\}_{n=1}^N$
6      Set $X_{0:t-1}^{(1)}, \dots, X_{0:t-1}^{(N)}, W_{t-1}^1, \dots, W_{t-1}^N = \mathtt{SIR}(t-1)$
7      Resample $A_t^{1:N} \sim \mathscr{R}(\cdot \mid (W_{t-1}^n)_{n=1}^N)$
8      Sample $X_t^n \sim M_t\left(\mathrm{d}x_t \mid X_{0:t-1}^{(A_t^n)}\right)$, $n = 1, \dots, N$
9      Compute the normalised weights $W_t^1, \dots, W_t^N$
10     return $\{X_{0:t}^{(n)}, W_t^n\}_{n=1}^N$
```

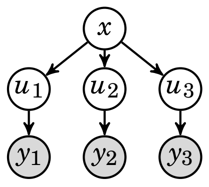


Figure 5.2. Dependency graph of hierarchical Bayesian models.

In this case, one can still apply Algorithm 13 to the graph, by considering the child-parent relationships in the graph, and applying the recursive algorithm to the parents before applying it to the children. For instance, in Figure 5.2, the leaf nodes are the nodes u_1, u_2, u_3 taken conditionally on the observations y_1, y_2, y_3. Then, the three leaves are combined at the root x of the tree.

This procedure relies on the following approach to forming an approximation over multivariate distributions $\pi(\mathrm{d}x, \mathrm{d}y)$ via marginal importance sampling:

$$\begin{aligned} \int f(x,y)\pi(\mathrm{d}x,\mathrm{d}y) &= \int f(x,y)\frac{\mathrm{d}\pi}{\mathrm{d}q_x \otimes \mathrm{d}q_y}(x,y)q_x(\mathrm{d}x)q_y(\mathrm{d}y), \\ &\approx \frac{1}{N^2}\sum_{m,n=1}^N f(X^m, Y^n)\frac{\mathrm{d}\pi}{\mathrm{d}q_x \otimes \mathrm{d}q_y}(X^m, Y^n), \end{aligned} \tag{5.23}$$

where $\frac{\mathrm{d}\pi}{\mathrm{d}q_x \otimes \mathrm{d}q_y}$ denotes the Radon–Nykodim derivative of π with respect to the product of distributions $\mathrm{d}q_x \otimes \mathrm{d}q_y$, and X^m, Y^n are i.i.d. sampled from q_x and q_y, respectively. This type of estimator, although quadratically more expensive (if one assumes that sampling from q_x and q_y is cheap) than its naïve counterpart

$$\frac{1}{N}\sum_{n=1}^N f(X^n, Y^n)\frac{\mathrm{d}\pi}{\mathrm{d}q_x \otimes \mathrm{d}q_y}(X^n, Y^n), \tag{5.24}$$

can be shown to have lower variance overall (Kuntz et al., 2024), justifying its use to combine the results of independent Monte Carlo approximations into a joint one.

This procedure is generalised in Algorithm 14, where the denomination π_c corresponds to the partial representation of the distribution π_t at the node c, and π_n to the partial representation of the distribution π_t at the node n.

Algorithm 14: Divide-and-conquer sequential Monte Carlo

```
1  Function DC-SMC($n$):
       /* $n$ now denotes a node in the tree and $k$ a particle.          */
2      if $c(n) = \emptyset$ then                    // $c(n)$ denotes the children of $n$
3          Sample $X_n^1, \dots, X_n^N \sim M_n(\mathrm{d}x_n)$
4          Compute the normalised weights $W_n^k \propto \pi_n(X_n^k) / M_n(X_k^n)$
5          return $\left\{X_n^k, W_n^k\right\}_{k=1}^N$
6      for $c \in c(n)$ do
7          Set $X_c^1, \dots, X_c^N, W_c^1, \dots, W_c^N =$ DC-SMC($c$)
8      Compute the weight tensor $W^{i_1, i_2, \cdots} \propto \prod_{c \in c(n)} W_c^{i_c}$
9      Sample $A_{c_1}^{1:N}, A_{c_2}^{1:N} \dots \sim \mathrm{Cat}(\cdot \mid W^{i_1, i_2, \cdots})$ independently, and set
        $X_c^k = X^{A_c^k}$ for $k = 1, \dots, N$
10     Sample $X_n^k \sim M_n\left(\mathrm{d}x_n \mid X_c^k, c \in c(n)\right)$, $k = 1, \dots, N$
11     Compute the normalised weights
        $W_n^k \propto \frac{\pi_n(X_k^n, \{X_c^k, c \in c(n)\})}{M_n(X_t^k \mid X_c^k, c \in c(n)) \prod_{c \in c(n)} \pi_c(X_c^k, \{X_{c'}^k, c' \in c(c)\})}$
12     return $\left\{\{X_n^k, X_c^k, c \in c(n)\}, W_n^k\right\}_{k=1}^N$
```

Recent theoretical results have been developed to analyse the performance of Algorithm 14, showing that it can be used to obtain a consistent estimator of the posterior distribution of interest (Lindsten et al., 2017; Kuntz et al., 2024), in some cases, and when parallelisation is available, at an optimal variance/cost trade-off (Kuntz et al., 2022).

Remark 5.14. *This structure is fundamental in Publication IV, where we define a tree structure for the smoothing distribution, allowing us to parallelise the smoothing step in a divide-and-conquer fashion on a GPU.*

6. Summary and Discussion

We conclude this thesis by summarising the main contributions of Publications I–VII and discussing how they relate to recent developments in literature, as well as future research directions. The organisation of this chapter is as follows: the contributions of each publication included in this thesis are summarised in separate sections, followed by a discussion of the main findings. Briefly, in Publication I a differentiable resampling mechanism for particle filters is proposed, with guarantees on the bias incurred, and is applied to the problem of learning proposals and parameters of state-space models. In Publication II, an extension of the parallel-in-time framework of Section 5.2.3 to the case of non-linear state-space models with additive noise is proposed, and its computational behaviour is discussed. In Publication III, we propose a parallel-in-time Gaussian process regression method, leveraging Sections 3.2.4 and 5.2.3, and we discuss its properties. In Publication IV, a parallel-in-time version of the particle smoother (see Section 3.4) and of the conditional SMC algorithm (see Section 4.3.2) are proposed; the convergence properties of the former are analysed in detail, and experiments are performed to illustrate the computational-statistical trade-offs. In Publication VI, a Wasserstein gradient-flow (see Section 2.3.5) approach to Gaussian-assumed filtering is proposed and compared to alternative methods; some care is taken to discuss differentiability of the procedure, and the resulting algorithm is applied to the problems of filtering and of parameter-learning in state-space models. In Publication V, we discuss an auxiliary MCMC sampler (see Section 4.2.3) for parallelisable (in time) inference in non-linear state-space models; the method builds on Publication IV, and Publication II, as well as on a re-interpretation of Section 4.3.4 and is illustrated on several high-dimensional examples. In Publication VII, we propose a comprehensive treatment of the conditional SMC instance of the auxiliary sampler of Publication V, as well as marginalisation thereof, and discuss its properties in details; in particular, we illustrate its scalability to high-dimensional problems compared to alternative methods. Finally, the chapter concludes with a short discussion of other works from the author that relate to the topics of this thesis, but are not included in it.

6.1 Publication I: Differentiable Particle Filtering via Entropy-Regularized Optimal Transport

This work inscribes itself in the trend of differentiable probabilistic programming for probabilistic inference, and is motivated by the fact that previous approaches to differentiable particle filtering (see Section 3.4.6) provided inconsistent gradients, which made the approach theoretically ungrounded. The aim of this publication is to propose a differentiable resampling mechanism for particle filters (see Section 3.4), and to illustrate its use in the context of variational learning (see Section 2.3.3) of proposal and model parameters. Formally, a way to understand resampling methods (see Section 3.4.2) is as a multiplication of the empirical distribution of the particles by a random matrix $R \in \{0,1\}^{N\times N}$, i.e., a matrix with exactly one non-zero entry per row: if $(x_t^i)_{i=1}^N$ are the particles at time t, the resampled ones are given by

$$\tilde{x}_t^i = \sum_{j=1}^{N} R_{ij} x_t^j, \tag{6.1}$$

and where $\mathbb{E}[\mathbb{1}^\top R]^\top = N W_t$, where $(W_t^i)_{i=1}^N$ are the normalised weights of the particles at time t. In other words, one can see the resampling operation as a stochastic map between the empirical distribution of the particles and the empirical distribution of the resampled particles. This observation is the starting point of the work presented in this publication (and of Reich, 2013, which we build on), which proposes to construct a differentiable deterministic approximation of this stochastic map, and to use it to compute consistent gradients of the particle filter.

The method. In practice, such a map can be constructed by considering a regularised version of the optimal transport problem used in (Reich, 2013): if $(x_t^i)_{i=1}^N$ are the particles at time t and are associated with weights $(W_t^i)_{i=1}^N$, then we can consider the following regularised optimal transport problem

$$P_\epsilon^N = \arg\min_{\gamma \in \Gamma(\mathbb{1}/N, W_t)} \sum_{i=1}^{N} \sum_{j=1}^{N} \gamma_{ij} \left\| x_t^i - x_t^j \right\|_2^2 + \epsilon \mathcal{D}_{\mathrm{KL}}(\gamma \parallel \mathbb{1}/N \otimes W_t), \tag{6.2}$$

which is the now classical regularised optimal transport problem (Cuturi, 2013). In which case, and following (6.1) and Reich (2013), the resampled particles are given by

$$\tilde{x}_t^i = N \sum_{j=1}^{N} P_{\epsilon,ij}^N x_t^j. \tag{6.3}$$

The procedure is then applied recursively to the particles at time $t-1, t, \dots, T$ and so on, until the final step is reached.

Main findings. This paper makes the following contributions.

1. A negative result: we show that previously proposed methods (Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018) for differentiable particle filtering provide inconsistent gradient estimators, only valid when the model is fully separable, i.e., when no dynamic is considered and the model is of the form
$$p(x_{0:T}, y_{1:T}) = \prod_{t=1}^{T} p(x_t)p(y_t \mid x_t). \tag{6.4}$$
This is important to understand the limitations of previous approaches to differentiable particle filtering, and to understand why the proposed method is different.
2. Because the solution of (6.2) P_ϵ^N is a deterministic function of the particles and the weights, and its gradient can be computed easily, the resulting resampling mechanism is differentiable, and therefore the resulting particle filter gradient can be computed using the reparameterisation trick (see Section 2.3) and the chain rule.
3. We show that the resulting particle filter is consistent on compact state-spaces, when the regularisation parameter ϵ and the number of particles N are chosen as $\epsilon = o(\log(N)^{-1})$ and N grows to infinity. Formally, the entropy-regularised particle filter filtering distribution converges to the true filtering distribution in the Wasserstein distance (see Section 2.3.5) sense (and in probability), and the resulting marginal likelihood estimate converges to the true one too.
4. The performance of the proposed method is shown to dominate the biased (but still useful) approaches of (Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018).

Reflections and outlook. The proposed method is a step forward in the direction of differentiable probabilistic programming, and provides a consistent gradient for particle filters. However, it is important to note that the proposed method is not without limitations. The first one is its computational cost: the proposed method requires solving a regularised optimal transport problem at each time step, which has a computational cost of $\mathcal{O}(N^2)$ in the number of particles N, albeit parallelisable on a GPU (Cuturi, 2013). This is in contrast with classical resampling methods, used in the non-consistent version of Naesseth et al. (2018); Maddison et al. (2017); Le et al. (2018), which have a computational cost of $\mathcal{O}(N)$.

The research on differentiable particle filtering (and differentiable probabilistic programming in general) has continued since the publication of this article, and has spawned several interesting developments.

1. Perhaps the most interesting development is given by Arya et al. (2023), who use smart couplings of Metropolis–Hastings (see Sections 4.1 and 5.1.2) kernels to compute low-variance gradients of MCMC algorithms. The resulting algorithm is computationally efficient, and provides excellent

preliminary results. While it is not directly clear how this can be extended to particle filters, we are hopeful that this will be possible using ideas from Jacob et al. (2020a).

2. Recently, Rosato et al. (2022) proposed to use differentiable particle filters within a gradient-informed (MALA, HMC, see Section 4.1) MCMC routine to learn parameters of state-space models. This is a different route to the one proposed in the rest of the literature, which was mostly concerned with solving variational inference problems. While it is clear (and illustrated in their article) that using a gradient-friendly approximation of the model provides increased efficiency, it is perhaps a bit disappointing that the gradients and likelihoods used do not correspond to the same quantity.

Consequently, the main gap to fill is still the obvious one: can we compute gradients of the unmodified particle filter algorithm in a way that is both computationally friendly and theoretically sound? A possible path to this could be to use control variates ideas appearing in the literature on leave-one-out gradient estimators for static latent discrete models (Dong et al., 2021; Titsias and Shi, 2022), and extend them to the case of particle filters. Dong et al. (2021) is particularly interesting as the coupling of resampling operations is now well studied within the literature on particle filters (see, e.g., Sen et al., 2018; Jacob et al., 2020a; Lee et al., 2020; Heng et al., 2021), and it is likely that the same ideas can be extended to computing gradients of the particle filter algorithm.

Finally, we mention that a review of differentiable particle filtering methods (Chen and Li, 2023) has recently been written, which provides a comprehensive overview of the topic, including of Publication I.

6.2 Publication II: Parallel Iterated Extended and Sigma-Point Kalman Smoothers

This work is concerned with the problem of state estimation in non-linear state-space models with additive noise, and proposes a parallel-in-time algorithm to solve it. The motivation for this work is that, while the iterated extended Kalman smoother (see Section 3.3.2) and the iterated sigma-point Kalman smoother (García-Fernández et al., 2016, see also Section 2.3.4) are well-studied algorithms, it is not clear how to parallelise with respect to the number of observations T, which is a problem when T is large. This article proposes a method to fully parallelise the iterated extended Kalman smoother and the iterated sigma-point Kalman smoother, and discusses its computational properties.

The method. The proposed method is based on the iterated extended Kalman smoother (see Section 3.3.2) and its extension to sigma-point linearisation (see Section 2.3.4), and the parallel-in-time framework of Section 5.2.3. The main idea is to consider the linearisation as an embarrassingly parallel problem, whereby the non-linear state-space model

$$\begin{aligned} x_{t+1} &= f(x_t) + \eta_t, \\ y_t &= h(x_t) + \varepsilon_t, \end{aligned} \tag{6.5}$$

is approximated by a linear state-space model

$$\begin{aligned} x_{t+1} &= A_t x_t + b_t + \eta_t, \\ y_t &= C_t x_t + d_t + \varepsilon_t, \end{aligned} \tag{6.6}$$

where A_t, b_t, C_t, d_t are the linearisation of f, h at time t. The choice of the linearisation is done using the previously computed marginal smoothing distribution m_t^k, P_t^k (see Section 3.3.2), whereby, using extended linearisation, the parameters of the linear state-space model are given as

$$\begin{aligned} A_t^k &= \nabla f(m_t^k), \quad & b_t^k &= f(m_t^k) - A_t m_t^k, \\ C_t^k &= \nabla h(m_t^k), \quad & d_t^k &= h(m_t^k) - C_t m_t^k. \end{aligned} \tag{6.7}$$

The subsequent smoothing approximation m_t^{k+1}, P_t^{k+1} is then computed using the parallel-in-time framework of Section 5.2.3.

Main findings. As outlined in the publication, the proposed method offers substantial computational gains compared to the sequential iterated extended and sigma-point Kalman smoother, validating the approach. In particular, we show that the parallel-in-time implementation fully dominates the sequential one in realistic scenarios, and that the computational gains are substantial. As expected, however, when the total number of observations T is very large, an empirical linear scaling of the computational cost is observed: the parallelisation capabilities of the GPU saturate.

Reflections and outlook. The proposed method is a step forward in the direction of parallelisable state estimation in non-linear state-space models, and improves on sequential methods, even when these are run on a high-performance CPU. Publication II was recently extended to the case of non-linear state-space models with non-additive noise in Yaghoobi et al. (2022), with a further extension to square-root Kalman filtering (whereby the square-root of the covariance is computed rather than the covariance itself), as well as parameter estimation. The method has since been adopted in the context of probabilistic numerics to compute the solution of ordinary differential equations (Bosch et al., 2023), and has percolated in the open-source community, for example in the DYNAMAX library[1]. Finally, the method is used to design proposal distributions in Publication IV, which is discussed in Section 6.4. In some sense, the parallel-in-time framework of Särkkä and García-Fernández (2021) is particularly suited to iterated procedures, be it linearisation or parameter learning, perhaps even more so than to the simple filtering-smoothing in LGSSMs that it was originally designed for. We expect it to thrive in this context.

[1]With some help from the author of this thesis.

6.3 Publication III: Temporal Gaussian Process Regression in Logarithmic Time

This work is concerned with the problem of Gaussian process regression (see Section 3.2.4) in 1D, and proposes a parallel-in-time algorithm to solve it. The motivation for this work is that, while Gaussian process regression is a well-studied problem, it is not clear how to parallelise it with respect to the number of observations T, which is a problem when T is large. This article proposes an end-to-end recipe to parallelise temporal Gaussian process regression, and discusses its properties.

The method. The proposed method is based on the state-space representation of Gaussian processes (see Section 3.2.4), and the parallel-in-time framework of Section 5.2.3. The main idea is to consider the state-space representation of the Gaussian process, and to use the parallel-in-time framework to compute the filtering distribution of the state-space model. Two difficulties arise in this context: the first one is that prediction consists in smoothing for missing observations, which is not considered in the original parallel-in-time framework; the second one is the state-space model representation of Gaussian processes is not unique, and some of them may be numerically unstable, obtaining a stable representation, compatible with the parallel-in-time formulation is therefore important.

The first difficulty is solved by generalising the work of Särkkä and García-Fernández (2021) described in Section 5.2.3. The main idea is to consider a formulation of the parallel-in-time operators and operands (see Section 5.2.3) such that the resulting filtering distribution is the same as the one obtained by the sequential filtering distribution. Heuristically, this is solved by remarking that, for a missing observation $y_t = \emptyset$, $p(x_t \mid x_{t-1}, y_t)$ is simply $p(x_t \mid x_{t-1})$, and that the two resulting filtering algorithms match. The smoothing pass then being a function of the filtering solution only, it can be kept unchanged. The second is solved by an operation known as balancing (Osborne, 1960) which consists in transforming the state-space model into a balanced representation, which is a representation where the state-space model is numerically stable.

Main findings. The main finding of this work is that the proposed method offers substantial computational gains compared to the sequential and batch Gaussian process regression, in particular when repeated evaluations of the procedure are required, such as when it is used within an MCMC routing, validating the approach. Nevertheless, because the procedure relies on Kalman-like operations, its computational cost scales poorly with the dimension of the state-space model (which, e.g., if a Matérn kernel was employed would be related to its order, see Section 3.2.4) so that for large-dimensional state-space models, the computational gains are less interesting.

Furthermore, care was given in this work to make sure that all operations were handled in a way that automatic differentiation tools could handle them, and that the resulting algorithm was efficient in practice. In view of this, an interesting finding of this work is the fact that the balancing operation can be fully ignored when computing the gradient of the likelihood function (for example in the context of maximum likelihood estimation), even though the resulting state-space model parametrisation depends on its original formulation. This means that we can discard the backpropagation of the balancing operation, which would take us from the stable representation to the original unstable one, and only backpropagate operations relating to the stable representation.

Reflections and outlook. The proposed method is a step forward in the direction of parallelisable Gaussian process regression, and provides substantial computational gains. This comes at the usual trade-offs of state-space representation (see, e.g., Solin, 2016), namely that they are only exact for a small class of kernels, and that they are more costly the higher the dimension of the state-space model representation. Nonetheless, extensions to this framework have been proposed in the literature, first and foremost in the context of spatio-temporal Gaussian processes (Särkkä et al., 2013), and we expect the method to extend to this context. An important caveat is that the method of Särkkä et al. (2013) requires to project the infinite dimensional state-space model onto a finite-dimensional one, which is often of relatively high dimension. As a consequence, methodological improvements would be required to make the method efficient in practice.

6.4 Publication IV: De-Sequentialized Monte Carlo

This work is concerned with the problem of particle smoothing (see Section 3.4), and proposes a parallel-in-time algorithm to solve it. The motivation for this work is that, while Gaussian approximations to the smoothing distribution are sufficient in some cases, they are biased and can never recover the true smoothing distribution, unless the model is linear and Gaussian. This article proposes an end-to-end recipe to parallelise the particle smoother, and discusses its properties.

The method. The proposed method is based on the divide-and-conquer SMC algorithm of Section 5.3 and, to a lesser extent, on the parallel-in-time framework of Section 5.2.3. The main ingredient to the proposed method is the following identity, which is a direct consequence of the definition of the smoothing distribution:

$$\begin{aligned}\pi^{\nu}_{a:b}(\mathrm{d}x_{a:b}) &:= \frac{1}{L^{\nu}_{a:b}} \nu_a(x_a) \prod_{t=a+1}^{b} M_t(x_t \mid x_{t-1}) G_t(x_t, x_{t-1}), \\ &= \frac{L^{\nu}_{a:c-1} L^{\nu}_{c:b}}{L^{\nu}_{a:b}} \omega^{\nu}_c(x_{c-1}, x_c) \pi^{\nu}_{a:c-1}(\mathrm{d}x_{a:c-1}) \pi^{\nu}_{c:b}(\mathrm{d}x_{c:b}),\end{aligned}$$

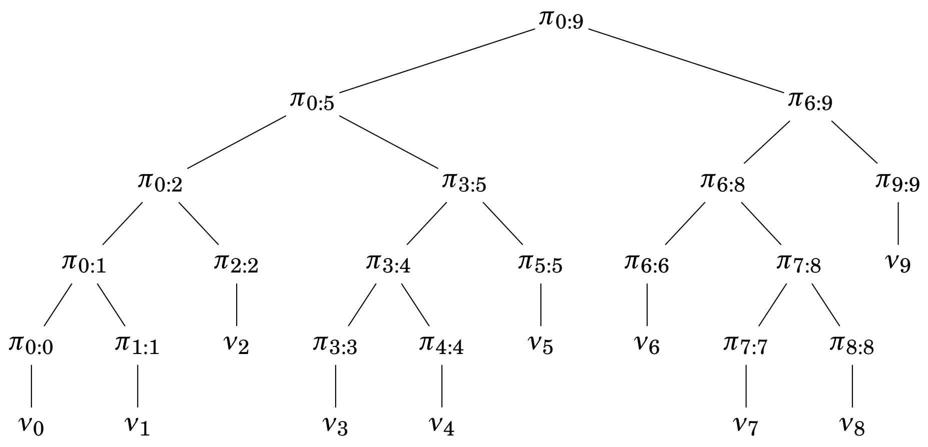


Figure 6.1. Smoothing tree structure for $T = 9$.

with

$$\omega_c^{\nu}(x_{c-1}, x_c) := \frac{M_c(x_c \mid x_{c-1}) G_c(x_c, x_{c-1})}{\nu_c(x_c)}.$$

This identity essentially states that the smoothing distribution can be achieved by stitching together the smoothing distributions of two sub-problems, for a weight $\omega_c^{\nu}(x_{c-1}, x_c)$. Clearly, provided that $\nu_0(x_0) \propto M_0(x_0)G_0(x_0)$, we have $\pi_{0:b}^{\nu} = \pi_{0:b}$, and we can recover the true smoothing distribution. This corresponds to the divide-and-conquer structure of Figure 6.1, and is the starting point of the proposed method. The partial smoothing distributions can then be replaced by Monte Carlo approximations, and the resulting algorithm consists in a recursive weighting and resampling of the particles, via stitching weights $\omega_c^{\nu}(x_{c-1}, x_c)$.

Main findings. As outlined in the publication, the proposed method offers computational gains compared to the sequential particle smoother, validating the approach. The method is particularly useful in the context when the dynamics are almost separable, so that little statistical information is shared between the different sub-problems, and the degeneracy problem is less severe. Several improvements to the method are proposed, including a lazy resampling scheme, leveraging similar ideas as in (Murray et al., 2016, see also Section 5.2.2), not requiring to form the weights $\omega_c^{\nu}(x_{c-1}, x_c)$, "smart" proposals, based on Publication II, and a parallel-in-time version of the conditional SMC algorithm of Section 4.3.2 that extends to the DnC-SMC algorithm in general. Theoretical guarantees on the approximation error (complementing the contemporary ones of Kuntz et al., 2024) are provided, and the method is illustrated on several examples.

Reflections and outlook. The proposed method is a step forward in the direction of parallelisable state estimation in non-linear state-space models, and provides substantial computational gains. However, and contrary to the Gaussian approximations of Section 5.2.3 and Publication II, they suffer from needing independent proposal distributions in time, in this case the ν_c. This was one of the reasons for introducing the auxiliary MCMC sampler of Publication V, which is discussed in Section 6.5. Nevertheless, we believe that it may be possible, by careful handling of the importance weights involved, to consider proposals $\nu_{0:T}(x_{0:T})$ over the entire trajectory, and to use the divide-and-conquer structure to compute the smoothing distribution in a parallel-in-time manner (provided that the proposals can also be sampled from in parallel). Furthermore, it is likely that part of the work of Publication V can be applied there too in a 'resample-move' fashion (Gilks and Berzuini, 2001), allowing the trajectories to be rejuvenated at each step, and the resulting algorithm to be more efficient.

6.5 Publication V: Auxiliary MCMC samplers for parallelisable inference in high-dimensional latent dynamical systems

This work is concerned with the problem of state and parameter inference in non-linear state-space models, and proposes two auxiliary MCMC samplers to solve it, both based on the same principle of augmenting the state-space model with an auxiliary variable:

$$\pi_T(\mathrm{d}x_{0:T}, \mathrm{d}u_{0:T}) = \pi_T(\mathrm{d}x_{0:T}) \prod_{t=0}^{T} \mathcal{N}\left(\mathrm{d}u_t; x_t, \frac{\delta_t}{2} I\right). \tag{6.8}$$

The motivation for this work is to leverage Gaussian approximations (see Section 3.3) to design efficient MCMC proposals, and, if possible, ones that can be parallelised in time, resulting in a computationally and statistically efficient algorithm.

The method. The proposed method is based on realising that the augmented state-space model (6.8) calls for a Hastings-within-Gibbs algorithm (see Section 4.1): given the state $x^k_{0:T}$,

1. sample $u^k_{0:T}$ from $\pi_T(\mathrm{d}u_{1:T} \mid x^k_{0:T})$ which can be done in closed form, and
2. sample $x^{k+1}_{0:T}$ from a $\pi_T(\mathrm{d}x_{0:T} \mid u^k_{0:T})$-invariant Markov kernel.

Step 2 can be done in many different ways, and we propose two different methods: (i) recognise that for Gaussian dynamics, the method is an instance of Titsias and Papaspiliopoulos (2018a, see also Section 4.2.3), and that the proposal can be reformulated as sampling from an auxiliary LGSSM (see Section 3.2.1), and (ii) use a conditional SMC algorithm (see Section 4.3.2) to sample from the conditional distribution of the state given the auxiliary variable, using approximations centered around the auxiliary variable $u^k_{0:T}$ to design the proposal.

MH extension*:* The first method essentially can be seen, in its simplest form as a direct application of the auxiliary MCMC principles of Titsias and Papaspiliopoulos (2018a, see also Section 4.2.3) to the state-space model, but also allows, when the dynamics are not Gaussian, to use linearisation methods such as these of Section 2.3.4 to design better proposals. Furthermore, the resulting algorithm can be parallelised in time using Publication II and its extension Yaghoobi et al. (2022).

CSMC extension*:* The second method essentially corresponds to swapping the roles of the dynamics and potential in the Feynman–Kac representation (see Section 3.1.2):

$$
\begin{aligned}
\pi(x_{0:T} \mid u_{0:T}) \propto\ & M_0(x_0)\left\{\prod_{t=1}^{T} M_t(x_t \mid x_{t-1})\right\} \\
& \times G_0(x_0)\mathcal{N}\left(u_0; x_0, \frac{\delta_0}{2}I\right)\left\{\prod_{t=1}^{T} G_t(x_t, x_{t-1})\mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}I\right)\right\}, \\
\propto\ & \mathcal{N}\left(x_0; u_0, \frac{\delta_0}{2}I\right)\left\{\prod_{t=1}^{T} \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}I\right)\right\} \\
& \times M_0(x_0)G_0(x_0)\left\{\prod_{t=1}^{T} G_t(x_t, x_{t-1})M_t(x_t \mid x_{t-1})\right\},
\end{aligned}
$$

where orange corresponds to the dynamics (or proposal), and blue to the potential (or likelihood) terms, respectively. This also allows one to use additional information, such as the gradient of the potential, or the prior dynamics in combination with the auxiliary variable, to design better proposals: for instance, if $M_t(x_t \mid x_{t-1})\mathcal{N}(u_t; x_t, \delta_t/2I)$ is tractable (conditionally on x_{t-1}), then we can use this as a proposal for x_t rather than simply $\mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}I\right)$. When the proposals are separable in time, i.e., when $\widetilde{M}_t(\mathrm{d}x_t \mid x_{t-1}) \equiv \widetilde{M}_t(\mathrm{d}x_t)$, the resulting algorithm can be parallelised in time using the conditional parallel-in-time SMC of Publication IV.

Main findings. We find that the proposed methods provide substantial statistical and computational gains compared to the traditional CSMC and MCMC algorithms, including the state-of-the-art one of Finke and Thiery (2023), on problems exhibiting high state-dimension (Crucinio and Johansen, 2022), or long time series with parameter learning (Mider et al., 2021).

In particular, whereas the parallel-in-time conditional SMC of Publication IV is more degenerate than its sequential counterpart, combining it with the auxiliary variable allows to design efficient proposals, and to solve high-dimensional (in this case the dimension was 30) problems while allowing for parallelisation in time. Indeed, when the proposals use a step-size δ_t that is small enough, the degeneracy problem disappears (comparatively to the standard CSMC algorithm using the same proposals), and the resulting algorithm is efficient in practice.

Reflections and outlook. It is of the author's opinion that the proposed methods are a step forward in the direction of parallelisable state estimation in non-linear state-space models, in particular that of Section 6.5, as they allow to use dynamics and likelihood information at a logarithmic computational cost while not incurring any approximation (asymptotically). This is a good example of the power of local approximations, and of the fact that they can be used to design efficient algorithms both statistically and computationally. Nonetheless, as for any local MCMC algorithm, the proposed methods are not without limitations, and it is likely that they will fail for some problems, in particular when the posterior is multi-modal, or when the dynamics are not well approximated by a Gaussian distribution, even after localisation. A very interesting piece of future work (not necessarily limited to state-space models) would be to embed the method within a tempering procedure (see Section 5.1.2), as there is a natural connection between the tempering parameter and the localisation parameter δ_t.

Importantly within the context of this thesis, the approach draws a bridge between two *a priori* completely different methods: that of Titsias and Papaspiliopoulos (2018a, see Section 4.2.3), corresponding to the choice in Section 6.5 and that of Finke and Thiery (2023), which the approach in Publication Publication V generalises. We fully leverage this connection in the follow-up work of Publication VII, which is discussed in Section 6.7.

Finally, we have recently been made aware of Dellaportas et al. (2023), which also considers a structural improvement to the auxiliary MCMC algorithm of Titsias and Papaspiliopoulos (2018a) within the context of state-space models. However, their method presents a number of key differences with our work: (i) they do not consider parallelisation, (ii) they do not extend the model to non-Gaussian priors, and, in fact, their method is not defined in this case as they explicitly require the precision matrix of the full prior dynamics. This is in contrast with our method, which we use in the context of non-linear dynamics too, and (iii) finally, they do not consider the CSMC extension, which is a key contribution of our work.

6.6 Publication VI: Variational Gaussian filtering via Wasserstein gradient flows

This work is concerned with the problem of Gaussian- and Gaussian-mixture-assumed filtering (see Section 3.3), and proposes a Wasserstein gradient-flow (Section 2.3.5) algorithm to solve it. It stems from the realisations that traditional Gaussian approximations, such as those presented in Section 2.3.4, are unable to treat fully multiplicative noise, that is, models of the form

$$\begin{aligned} x_{t+1} &= f(x_t) + \eta_t, \\ y_t &= h(x_t)\varepsilon_t, \end{aligned} \tag{6.9}$$

such as stochastic volatility models (Heston, 1993), or multi-modal models, such as those arising in the context of multi-target tracking (see, e.g., the original particle filter example of Gordon et al., 1993). This article proposes an end-to-end recipe to approximate the filtering distribution using Wasserstein gradient flows, and discusses its properties.

The method. The bulk of the method is based on Lambert et al. (2022), who proposed to restrict gradient flows to the subspace of Gaussian (and mixture thereof) distributions, recovering a method first proposed by Särkkä (2007) to perform Gaussian-assumed filtering in stochastic differential equations. For simplicity, we considered the case where the dynamics are Gaussian, so that Gaussian distributions (and mixture thereof) are preserved by the dynamics, and only the update of the distribution, given the observation, needs to be approximated. The main idea is to consider the Wasserstein gradient flow of the Kullback–Leibler divergence between the true filtering distribution and the Gaussian (or Gaussian-mixture) distribution, and to approximate it using a gradient flow of the Kullback–Leibler divergence between the Gaussian (or Gaussian-mixture) distribution and the true filtering distribution: At time t, the filtering distribution is given by $\pi_t(\mathrm{d}x_t) \approx \mathcal{N}(m_t, P_t)$, and the update of the filtering distribution is given by minimising the Kullback–Leibler divergence between $\pi_t(\mathrm{d}x_{t+1})$ and the true filtering distribution $\pi_{t+1}(\mathrm{d}x_{t+1})$, given the observation y_t. While this can be done within the parameter space (as is traditionally done in, e.g., García-Fernández et al., 2016), we elect to do it within the space of probability distributions, and to use the Gaussian-constrained Wasserstein gradient flow of the Kullback–Leibler divergence between $\pi_t(\mathrm{d}x_{t+1})$ and $\pi_{t+1}(\mathrm{d}x_{t+1})$, given the observation y_t to find the best Gaussian (or Gaussian-mixture) approximation to the true filtering distribution. The resulting algorithm is then applied recursively to the particles at time $t-1, t, \dots, T$ and so on, until the final step is reached. We also leverage the fixed-point structure of the Wasserstein gradient flow to allow for gradient backpropagation, and to compute the gradients of the (approximate) marginal likelihood with respect to parameters.

Main findings. We find that the proposed method recovers reasonable posterior approximations for the filtering distribution, and that the resulting likelihood estimates are consistent to the point that they can be used for parameter learning. In fact, there is hardly any difference in the solution recovered using the proposed method and state-of-the-art (almost everywhere) differentiable particle filters such as Malik and Pitt (2011a, see also Section 3.4.2). This is in contrast with the Gaussian approximations of Section 2.3.4, which are doomed to fail in this case.

Reflections and outlook. Originally, we were trying to solve the problem of Gaussian-approximated smoothing, solving the full smoothing problem at once using a variational approach such as Campbell et al. (2021), where a reverse-time Markov kernel is used to approximate the smoothing distribution. However, this quickly resulted in a large number of intractable integrals that were not easily approximated, and we elected to consider the filtering problem instead. We still believe that the full smoothing problem can (and should) be solved at once, at least in convex cases such as additive noise models, and that it would likely result in better solutions than the current procedure, but also than iterative solutions such as Tronarp et al. (2018, see also Section 3.3.2).

6.7 Publication VII: Particle-MALA and Particle-mGRAD

This work is concerned with the problem of state inference in non-linear Markovian models. The motivation for this work is to take advantage of the representation of Finke and Thiery (2023) as given in Publication V to form a full generalisation of gradient-based samplers (see Section 4.1) to CSMC, thereby solving the problem of the curse of dimensionality in high-dimensional state-space models.

The method. The proposed method starts from a different auxiliary variable augmentation than that of Publication V:

$$\pi_T(\mathrm{d}x_{0:T}, \mathrm{d}u_{0:T}) = \pi_T(\mathrm{d}x_{0:T}) \prod_{t=0}^{T} \mathcal{N}\left(\mathrm{d}u_t; x_t + \nabla_{x_t}\varphi(x_{0:T}), \frac{\delta_t}{2} I\right), \tag{6.10}$$

where $\varphi(x_{0:T})$ is a placeholder that can take several forms: for instance, $\varphi(x_{0:T}) = \log \pi_t(x_{0:t})$ corresponds to using "filtering" gradient information, and $\varphi(x_{0:T}) = \log \pi_T(x_{0:T})$ corresponds to using "smoothing" or "look-ahead" gradient information.

Once this representation is obtained, as for Publication V, we can sample from $\pi_T(\mathrm{d}u_{0:T} \mid x_{0:T})$ in closed form, and sample from $\pi_T(\mathrm{d}x_{0:T} \mid u_{0:T})$ using a conditional SMC algorithm (see Section 4.3.2). The choice of proposal distributions for the conditional SMC algorithm is crucial, and we propose three different methods.

1. Sample from $\mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2} I\right)$, which generalises the auxiliary MALA algorithm (see Section 4.2.3). It does not require any knowledge of the dynamics, and can be used as a default proposal. This comprises the Particle-MALA algorithm and its variants.
2. When dynamics are conditionally Gaussian:
$$M_t(x_t \mid x_{t-1}) = \mathcal{N}(x_t; f_t(x_{t-1}), Q_t(x_{t-1})),$$
sample from the distribution proportional to $M_t(x_t \mid x_{t-1}) \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2} I\right)$. This comprises the Particle-aGRAD algorithm and some of its variants.
3. When the dynamics are fully Gaussian: $M_t(x_t \mid x_{t-1}) = \mathcal{N}(x_t; F_t x_t + b_t, Q_t)$, sample from the twisted distribution (see Section 3.4.5) proportional to
$$M_t(x_t \mid x_{t-1}) \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2} I\right) \times \int \prod_{s=t+1}^{T} p(\mathrm{d}x_s \mid x_{s-1}) \mathcal{N}\left(u_s; x_s, \frac{\delta_s}{2} I\right), \tag{6.11}$$
which can be done using Kalman filtering and smoothing (see Section 3.2.2) while not incurring additional computational cost. It is the least general of the three methods, but it is also the most efficient when applicable. This is the twisted Particle-aGrad algorithm.

In the above, the modified CSMC algorithm potential function will have the auxiliary variable $u_{0:T}$ appear in the calculation. We show that, in some cases,

Table 6.2. The methods mentioned introduced in this work.

Method	**Special case if $N = T = 1$**
Particle-aMALA	aMALA (see Section 4.2.3)
Particle-MALA	MALA (see Section 4.2.3)
Particle-aGrad	aGrad (see Section 4.2.3)
Particle-mGrad	mGrad (Titsias and Papaspiliopoulos, 2018b)
Twisted Particle-aGrad	aGrad
Particle-PCNL	PCNL (Cotter et al., 2013)

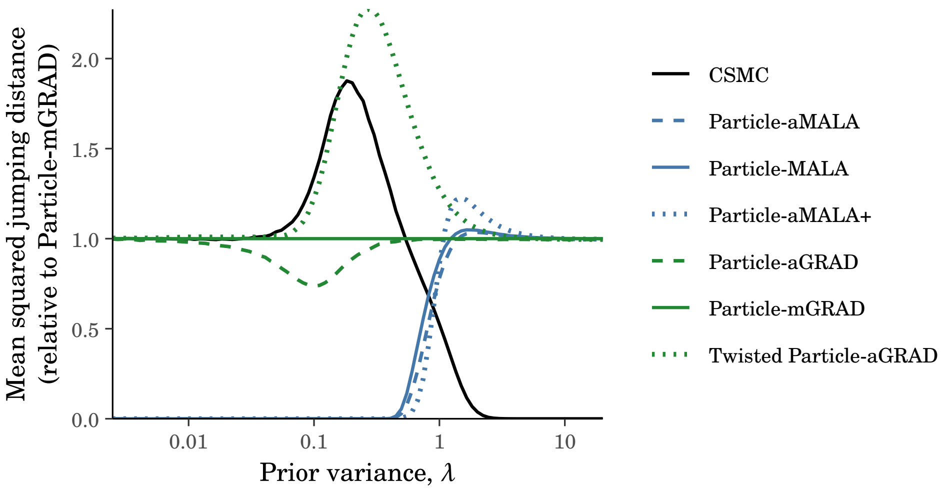


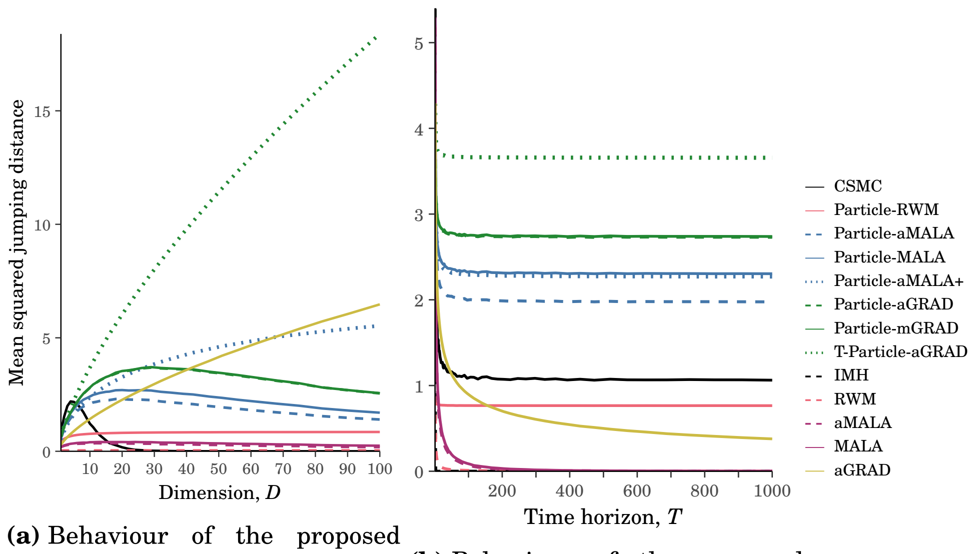
Figure 6.3. Behaviour of the proposed methods in the dimension of the state-space model D as a function of the informativeness (variance) of the dynamics, this is reported as relative expected squared jump distance (Pasarica and Gelman, 2010) of the Markov chain with Particle-mGrad as a reference.

we can marginalise it out while still keeping a linear cost in both the time and particles dimension, and that the resulting algorithm is a valid MCMC algorithm, always more efficient in the Peskun ordering sense (Peskun, 1973; Andrieu and Vihola, 2016) than its auxiliary variable counterpart.

Main findings. The algorithms proposed and their static special cases are given in Table 6.2.

We find that the aGrad and mGrad variants of our methods interpolate between the CSMC and the Particle-(a)MALA algorithms, respectively when dynamics of the model are highly informative, and when they are not (see Figure 6.3). This, in a sense, solves the calibration problem of having to choose between a fully prior-driven and a fully likelihood-driven algorithm, and allows to use the best of both worlds.

Additionally, we find that the proposed methods provide substantial computational gains compared to the traditional MCMC algorithms and non-gradient based CSMC algorithms, and that the resulting algorithms are statistically



(a) Behaviour of the proposed methods in the dimension of the state-space model D.

(b) Behaviour of the proposed methods in the number of time steps T.

Figure 6.4. Behaviour of the proposed methods in the dimension of the state-space model D and the number of time steps T. The metric reported is the expected squared jump distance (Pasarica and Gelman, 2010) of the Markov chain, which is a measure of the mixing of the Markov chain. See Publication VII for details on the scaling. In the legend, 'T-' stands for twisted, and '+' for the use of "smoothing" gradients.

efficient. More precisely, we observed that all proposed method have stable behaviour in the number of time steps T (see Figure 6.4b), and that their behaviour improves upon previous methods in the literature when the dimension of the state-space model D increases (see Figure 6.4a). In particular, Particle-aMALA+ (i.e., with smoothing gradients) and Particle-aGrad with twisted proposals do not see their efficiency degrade when the dimension of the state-space model increases, which is a substantial improvement over previous methods in the literature. On the other hand, aGrad, using the implementation of Publication V, while providing a good scaling in the dimension, sees its efficiency degrade when the number of time steps increases, which is a substantial limitation of the method. This justifies the interpretation of the proposed methods as unifying both CSMC and MCMC methods, providing good scaling in both the dimension and the number of time steps.

Reflections and outlook. The proposed methods are a step forward in the direction of high-dimensional state estimation in non-linear state-space models, and provide substantial statistical gains. Some of them are directly compatible with the parallel-in-time framework of Publication IV, and we expect that they will be used in this context in the future. Furthermore, while we considered Markovian models only, tree-based methods such as those of Kuntz et al. (2024), presented in Section 5.3 can also be treated using the proposed methods, by considering the DnC extension of CSMC given in Publication IV. Finally, while we have considered continuous-state models only, a natural extension of the proposed methods would be to consider discrete-state models which are naturally high-dimensional (if the number of states is high). Auxiliary MCMC methods have been used in this context before (Märtens et al., 2019; Rhodes and Gutmann, 2022), and we expect that the proposed methods can be used in this context too, and that they will provide substantial gains.

A number of questions remain open: while we have shown that the proposed methods are empirically more efficient than their non-gradient based counterparts, we do not have (in contrast to Finke and Thiery, 2023) a formal proof of this. We believe that a dimensional scaling of the style given in Roberts and Rosenthal (1998) would likely hold for at least Particle-aMALA with smoothing gradients, but the technical details of the proof are still to be worked out and are likely to be challenging due to the non-Markovianity of the auxiliary model. This is nonetheless a very promising direction, and we expect that the proposed methods will be used in the future to solve high-dimensional state-space models, where traditional methods have so far failed.

Finally, the marginalisation of the auxiliary variable in the CSMC weights presents some kinship with Wigren et al. (2019), which is too interested in eliminating a parameter (akin to our auxiliary variable) from the weights of the CSMC algorithm to improve its efficiency. This relationship is not yet fully understood in particular because they use marginalised *proposals*, whereas we take full advantage of the auxiliary variable to design better proposals, but we believe that the two methods can be combined to extend the method to a more general class of prior-likelihood models than the one considered in this work.

6.8 Open source contributions

The author has also been working on some open-source library, which are directly related to the content of this thesis. For example, the author has contributed to Flamary et al. (2021), a Python package for optimal transport, and is an active contributor of Cabezas et al. (2024), a Python package for MCMC and SMC algorithms, which includes some of the methods presented in this thesis. A particularity of BlackJAX is its composability, where the emphasis is to allow the user to combine MCMC and SMC primitives in a very flexible manner so that they can design and implement their own algorithms if needed. The success of this approach is highlighted by the library's popularity and its use in several research projects, some of them listed in Cabezas et al. (2024). The library also offers high-level access to standard MCMC and SMC algorithms for users who are not interested in designing their own algorithms, and therefore provides a good balance between flexibility and ease of use.

6.9 Relevant ongoing works

In this thesis, we have presented a number of methods to solve the problem of state and parameter inference in non-linear state-space models, some exhibiting statistical parallelisation, others computational parallelisation. While not formally included as part of this thesis, we would have felt remiss not to mention the following works, which are directly related to the content of this thesis.

Square-root and fully non-linear parallel-in-time Kalman smoothing. This work is a follow-up on the parallel-in-time smoother of Publication II for non-linear state-space models with additive noise. A number of improvements are made to Publication II: (i) the original method of Särkkä and García-Fernández (2021) is adapted to square-root filters and smoothers (Thornton, 1976; Bierman, 1977), allowing for better numerical stability at lower precision, (ii) the method is extended to the case of fully non-linear state-space models via generalised statistical linear regression (Tronarp et al., 2018, see also Section 2.3.4), and (iii) the fixed-point structure of the resulting smoother is used to compute the gradients of the marginal likelihood with respect to the parameters in an efficient manner (Christianson, 1994), allowing for parameter learning.

Coupled piecewise-deterministic Markov processes. This ongoing work, available as a preprint (Corenflos et al., 2023), is an exciting piece of future work, which is concerned with debiasing a class of continuous-time non-reversible Markov chain Monte Carlo algorithms (see Section 4.1) using the framework of coupled MCMC (Jacob et al., 2020b, see also Section 5.1.2). The problem of debiasing continuous-time algorithms is largely more challenging than its discrete-time counterpart, due to the stochasticity of the time components, and two coupled PDMPs will not necessarily be synchronous, making the framework of Jacob et al. (2020b) not directly applicable. We provide a general framework to debias continuous-time algorithms, via successive time-synchronisation and state-matching, which is applicable to a wide class of continuous-time algorithms, and we illustrate it on some standard PDMPs (e.g., Bouchard-Côté et al., 2018). However, in its current state, the article lacks empirical evidence, and we therefore cannot report on its findings as part of this thesis. This work is partially based on another work of the author, Corenflos and Särkkä (2022), which is concerned with forming coupled distributions using rejection sampling, and which is also not included in this thesis.

References

Acevedo, W., de Wiljes, J., and Reich, S. (2017). Second-order accurate ensemble transform particle filters. *SIAM Journal on Scientific Computing*, 39(5):A1834–A1850. Cited on page 72.

Agapiou, S., Papaspiliopoulos, O., Sanz-Alonso, D., and Stuart, A. M. (2017). Importance Sampling: Intrinsic Dimension and Computational Cost. *Statistical Science*, 32(3):405 – 431. Cited on page 35.

Alenlöv, J., Doucet, A., and Lindsten, F. (2021). Pseudo-marginal Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 22(141):1–45. Cited on page 92.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342. With discussion. Cited on pages 81, 98, 99, 100, and 101.

Andrieu, C., Lee, A., and Livingstone, S. (2020). A general perspective on the Metropolis–Hastings kernel. *arXiv e-prints*, page arXiv:2012.14881. Cited on page 82.

Andrieu, C., Lee, A., and Vihola, M. (2018). Uniform ergodicity of the iterated conditional SMC and geometric ergodicity of particle Gibbs samplers. *Bernoulli*, 24(2):842–872. Cited on pages 99, 101, and 102.

Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697 – 725. Cited on pages 29, 89, 91, and 99.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and computing*, 18(4):343–373. Cited on pages 82 and 86.

Andrieu, C. and Vihola, M. (2016). Establishing some order amongst exact approximations of MCMCs. *Annals of Applied Probability*, 26(5):2661–2696. Cited on page 136.

Arya, G., Seyer, R., Schäfer, F., Lew, A., Huot, M., Mansinghka, V. K., Rackauckas, C., Chandra, K., and Schauer, M. (2023). Differentiating Metropolis-Hastings to optimize intractable densities. *arXiv preprint arXiv:2306.07961*. Cited on page 125.

Banerjee, A. (2006). On Bayesian bounds. In *Proceedings of the 23rd international conference on Machine learning*, pages 81–88. Cited on page 38.

Bardenet, R. and Hardy, A. (2020). Monte Carlo with determinantal point processes. *The Annals of Applied Probability*, 30(1):368 – 417. Cited on page 36.

Barker, A. A. (1965). Monte Carlo calculations of the radial distribution functions for a proton–electron plasma. *Australian Journal of Physics*, 18(2):119–134. Cited on page 85.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43. Cited on page 66.

Bell, B. and Cathey, F. (1993). The iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297. Cited on page 63.

Bell, B. M. (1994). The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization*, 4(3):626–636. Cited on page 64.

Berger, J. (2013). *Statistical decision theory: foundations, concepts, and methods*. Springer Science & Business Media. Cited on page 27.

Besag, J. E. (1994). Contribution to the discussion on 'Representations of knowledge in complex systems' by Grenander, U and Miller, M. I.. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581. Cited on page 87.

Betancourt, M. (2018). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*. Cited on page 88.

Bierkens, J., Fearnhead, P., and Roberts, G. (2019). The Zig-Zag process and super-efficient sampling for Bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320. Cited on page 82.

Bierman, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation*. Academic Press. Cited on page 139.

Biswas, N., Jacob, P. E., and Vanetti, P. (2019). Estimating convergence of Markov chains with L-lag couplings. *Advances in Neural Information Processing Systems*, 32. Cited on page 110.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877. Cited on page 36.

Blelloch, G. E. (1989). Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11):1526–1538. Cited on page 111.

Blelloch, G. E. (1990). Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University. Cited on page 110.

Bolić, M., Djurić, P. M., and Hong, S. (2004). Resampling algorithms for particle filters: A computational complexity perspective. *EURASIP Journal on Advances in Signal Processing*, 2004:1–11. Cited on page 69.

Bosch, N., Corenflos, A., Yaghoobi, F., Tronarp, F., Hennig, P., and Särkkä, S. (2023). Parallel-in-time probabilistic numerical ODE solvers. *arXiv preprint arXiv:2310.01145*. Cited on page 127.

Bouchard-Côté, A., Vollmer, S. J., and Doucet, A. (2018). The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method. *Journal of the American Statistical Association*, 113(522):855–867. Cited on pages 82 and 139.

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. Cited on page 64.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov chain Monte Carlo*. CRC press. Cited on pages 36, 81, and 88.

Bugallo, M. F., Elvira, V., Martino, L., Luengo, D., Miguez, J., and Djuric, P. M. (2017). Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79. Cited on page 36.

Cabezas, A., Corenflos, A., Lao, J., Louf, R., et al. (2024). BlackJAX: Composable Bayesian inference in JAX. Cited on pages 25 and 138.

Calderhead, B. (2014). A general construction for parallelizing Metropolis–Hastings algorithms. *Proceedings of the National Academy of Sciences*, 111(49):17408–17413. Cited on pages 89, 92, and 93.

Campbell, A., Shi, Y., Rainforth, T., and Doucet, A. (2021). Online variational filtering and parameter learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*. Cited on page 134.

Cappé, O. and Moulines, E. (2009). On-Line Expectation–Maximization Algorithm for latent Data Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(3):593–613. Cited on page 76.

Cardoso, G., Janati El Idrissi, Y., Le Corff, S., Moulines, E., and Olsson, J. (2023). State and parameter learning with PARIS particle Gibbs. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3625–3675. PMLR. Cited on page 102.

Casella, G. and George, E. I. (1992). Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174. Cited on page 88.

Chada, N. K., Franks, J., Jasra, A., Law, K. J., and Vihola, M. (2021). Unbiased inference for discretely observed hidden Markov model diffusions. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2):763–787. Cited on page 109.

Chandra, R., Dagum, L., Kohr, D., Menon, R., Maydan, D., and McDonald, J. (2001). *Parallel programming in OpenMP*. Morgan kaufmann. Cited on page 108.

Chen, F., Lovász, L., and Pak, I. (1999). Lifting Markov chains to speed up mixing. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 275–281. Cited on page 82.

Chen, X. and Li, Y. (2023). An overview of differentiable particle filters for data-adaptive sequential Bayesian inference. Cited on pages 80 and 126.

Chen, Y., Chewi, S., Salim, A., and Wibisono, A. (2022). Improved analysis for a proximal algorithm for sampling. In Loh, P.-L. and Raginsky, M., editors, *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 2984–3014. PMLR. Cited on page 97.

Chopin, N. and Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo*. Springer. Cited on pages 34, 35, 36, 49, 53, 54, 67, 69, 71, 99, and 102.

Chopin, N. and Singh, S. S. (2015a). On particle Gibbs sampling. *arXiv preprint arXiv:1304.1887 version 1*. Cited on pages 93 and 104.

Chopin, N. and Singh, S. S. (2015b). On particle Gibbs sampling. *Bernoulli*, 21(3):1855–1883. Cited on pages 100, 101, and 102.

Chopin, N., Singh, S. S., Soto, T., and Vihola, M. (2022). On resampling schemes for particle filters with weakly informative observations. *The Annals of Statistics*, 50(6):3197–3222. Cited on page 69.

Christianson, B. (1994). Reverse accumulation and attractive fixed points. *Optimization Methods & Software - OPTIM METHOD SOFTW*, 3:311–326. Cited on pages 66 and 139.

Corcoran, J. and Tweedie, R. (2002). Perfect sampling from independent Metropolis–Hastings chains. *Journal of Statistical Planning and Inference*, 104(2):297–314. Cited on page 109.

Corenflos, A. and Särkkä, S. (2022). The coupled rejection sampler. *arXiv preprint arXiv:2201.09585*. Cited on page 139.

Corenflos, A., Sutton, M., and Chopin, N. (2023). Debiasing piecewise deterministic Markov process samplers using couplings. *arXiv preprint arXiv:2306.15422*. Cited on page 139.

Cotter, S. L., Roberts, G. O., Stuart, A. M., and White, D. (2013). MCMC methods for functions: Modifying old algorithms to make them faster. *Statistical Science*, 28(3):424–446. Cited on pages 96 and 136.

Cox, H. (1964). On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Transactions on Automatic Control*, 9(1):5–12. Cited on page 63.

Crisan, D. and Doucet, A. (2002). A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746. Cited on page 68.

Crucinio, F. R. and Johansen, A. M. (2022). A divide and conquer sequential Monte Carlo approach to high dimensional filtering. *arXiv preprint arXiv:2211.14201*. Cited on page 132.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc. Cited on pages 76, 124, and 125.

Dahlin, J., Lindsten, F., Kronander, J., and Schön, T. B. (2015). Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. Cited on page 92.

Dau, H.-D. and Chopin, N. (2023). On backward smoothing algorithms. *The Annals of Statistics*, 51(5):2145 – 2169. Cited on pages 74 and 102.

Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer. Cited on pages 35, 53, and 68.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436. Cited on pages 36 and 67.

Del Moral, P., Doucet, A., and Jasra, A. (2012). On adaptive resampling strategies for sequential Monte Carlo methods. *Bernoulli*, 18(1):252–278. Cited on page 69.

Deligiannidis, G., Doucet, A., and Pitt, M. K. (2018). The Correlated Pseudomarginal Method. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 80(5):839–870. Cited on page 92.

Deligiannidis, G., Doucet, A., and Rubenthaler, S. (2020). Ensemble rejection sampling. *arXiv preprint arXiv:2001.09188*. Cited on page 104.

Dellaportas, P., Titsias, M. K., Petrova, K., and Plataniotis, A. (2023). Scalable inference for a full multivariate stochastic volatility model. *Journal of Econometrics*, 232(2):501–520. Cited on page 133.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22. Cited on page 66.

Diaconis, P. and Freedman, D. (1980). Finite Exchangeable Sequences. *The Annals of Probability*, 8(4):745 – 764. Cited on page 94.

Diaconis, P., Holmes, S., and Neal, R. M. (2000). Analysis of a nonreversible Markov chain sampler. *The Annals of Applied Probability*, 10(3):726 – 752. Cited on page 82.

Diggle, P. J., Tawn, J. A., and Moyeed, R. A. (1998). Model-based geostatistics. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 47(3):299–350. Cited on page 88.

Dong, Z., Mnih, A., and Tucker, G. (2021). Coupled gradient estimators for discrete latent variables. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*. Cited on pages 42 and 126.

Donsker, M. D. and Varadhan, S. R. S. (1976). Asymptotic evaluation of certain Markov process expectations for large time—iii. *Communications on Pure and Applied Mathematics*, 29(4):389–461. Cited on page 38.

Douc, R., Moulines, E., Priouret, P., and Soulier, P. (2018). *Markov Chains*. Springer International Publishing. Cited on pages 38, 82, and 115.

Doucet, A. and Johansen, A. M. (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In Crisan, D. and Rozovskii, B., editors, *The Oxford Handbook of Nonlinear Filtering*, Oxford Handbooks, chapter 24, pages 656–704. Oxford University Press. Cited on pages 49, 54, 67, 68, and 77.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222. Cited on page 88.

Durmus, A. and Moulines, É. (2019). High-dimensional Bayesian inference via the unadjusted Langevin algorithm. *Bernoulli*, 25(4A):2854 – 2882. Cited on page 87.

Durmus, A., Moulines, E., and Pereyra, M. (2018). Efficient Bayesian computation by proximal Markov chain Monte Carlo: when Langevin meets Moreau. *SIAM Journal on Imaging Sciences*, 11(1):473–506. Cited on page 97.

Efron, B. (2022). *Exponential families in theory and practice*. Cambridge University Press. Cited on page 30.

Elvira, V. and Martino, L. (2021). Advances in importance sampling. *arXiv preprint arXiv:2102.05407*. Cited on page 33.

Elvira, V., Martino, L., and Robert, C. P. (2022). Rethinking the effective sample size. *International Statistical Review*, 90(3):525–550. Cited on page 69.

Fearnhead, P. and Meligkotsidou, L. (2016). Augmentation schemes for particle MCMC. *Statistics and Computing*, 26:1293–1306. Cited on page 106.

Finke, A. and Thiery, A. H. (2023). Conditional sequential Monte Carlo in high dimensions. *Annals of Statistics*. Cited on pages 98, 102, 104, 105, 132, 133, 135, and 138.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. (2021). POT: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8. Cited on page 138.

Forney, G. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278. Cited on page 120.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. Cited on page 59.

Garcìa-Fernàndez, A. F., Svensson, L., Morelande, M. R., and Särkkä, S. (2015). Posterior linearization filter: Principles and implementation using sigma points. *IEEE Transactions on Signal Processing*. Cited on pages 43 and 44.

García-Fernández, Á. F., Svensson, L., and Särkkä, S. (2016). Iterated posterior linearization smoother. *IEEE Transactions on Automatic Control*, 62(4):2056–2063. Cited on pages 44, 63, 126, and 134.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-6(6):721–741. Cited on page 88.

Gentle, J. E. (2003). *Random number generation and Monte Carlo methods*, volume 381. Springer. Cited on page 32.

Gerber, M. and Chopin, N. (2015). Sequential quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(3):509–579. Cited on page 36.

Gerber, M., Chopin, N., and Whiteley, N. (2019). Negative association, ordering and convergence of resampling methods. *The Annals of Statistics*, 47(4):2236–2260. Cited on page 71.

Gerber, M. and Douc, R. (2021). A global stochastic optimization particle filter algorithm. *Biometrika*, 109(4):937–955. Cited on page 67.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *Interface Proceedings*. Cited on page 109.

Geyer, C. J. and Thompson, E. A. (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90(431):909–920. Cited on page 109.

Gilks, W. R. and Berzuini, C. (2001). Following a moving target – Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146. Cited on page 131.

Glynn, P. W. and Rhee, C.-h. (2014). Exact estimation for Markov chain equilibrium expectations. *Journal of Applied Probability*, 51(A):377–389. Cited on page 109.

Glynn, P. W. and Szechtman, R. (2002). Some new perspectives on the method of control variates. In *Monte Carlo and Quasi-Monte Carlo Methods 2000: Proceedings of a Conference held at Hong Kong Baptist University, Hong Kong SAR, China, November 27–December 1, 2000*, pages 27–49. Springer. Cited on page 42.

Godsill, S. J., Doucet, A., and West, M. (2004). Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168. Cited on page 73.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. Cited on page 108.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F, Radar and Signal Processing*, 140(2):107–113. Cited on pages 24, 67, 68, 69, and 133.

Guarniero, P., Johansen, A. M., and Lee, A. (2017). The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647. Cited on page 78.

Gupta, C., Latypov, R., Maus, Y., Pai, S., Särkkä, S., Studený, J., Suomela, J., Uitto, J., and Vahidi, H. (2023). Fast dynamic programming in trees in the MPC model. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '23, page 443–453, New York, NY, USA. Association for Computing Machinery. Cited on page 120.

Haario, H., Laine, M., Mira, A., and Saksman, E. (2006). DRAM: efficient adaptive MCMC. *Statistics and computing*, 16(4):339–354. Cited on page 86.

Hartikainen, J. and Särkkä, S. (2010). Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *Proceedings of the 2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 379–384. Cited on page 61.

Hassan, S., Särkkä, S., and García-Fernández, Á. F. (2021). Temporal parallelization of inference in hidden Markov models. *IEEE Transactions on Signal Processing*, 69:4875–4887. Cited on pages 114, 117, and 120.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109. Cited on pages 84 and 85.

Heng, J., Bishop, A. N., Deligiannidis, G., and Doucet, A. (2020). Controlled sequential Monte Carlo. *The Annals of Statistics*, 48(5):2904 – 2929. Cited on page 78.

Heng, J., Houssineau, J., and Jasra, A. (2021). On unbiased score estimation for partially observed diffusions. *arXiv preprint arXiv:2105.04912*. Cited on page 126.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*. Cited on page 133.

Higdon, D. M. (1998). Auxiliary variable methods for Markov chain Monte Carlo with applications. *Journal of the American Statistical Association*, 93(442):585–595. Cited on pages 81 and 89.

Hillis, W. D. and Steele Jr, G. L. (1986). Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183. Cited on pages 110 and 111.

Hoffman, A. J. et al. (1963). On simple linear programming problems. In *Proceedings of Symposia in Pure Mathematics*, volume 7, pages 317–327. Cited on page 72.

Hoffman, M. D., Gelman, A., et al. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623. Cited on page 110.

Huber, M. L. (2016). *Perfect simulation*, volume 148. CRC Press. Cited on page 109.

Jacob, P. E., Lindsten, F., and Schön, T. B. (2020a). Smoothing with couplings of conditional particle filters. *Journal of the American Statistical Association*, 115(530):721–729. Cited on pages 109 and 126.

Jacob, P. E., O'Leary, J., and Atchadé, Y. F. (2020b). Unbiased Markov chain Monte Carlo methods with couplings. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(3):543–600. Cited on pages 109 and 139.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations*. Cited on page 42.

Julier, S., Uhlmann, J., and Durrant-Whyte, H. F. (2000). A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482. Cited on page 62.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422. Cited on pages 44 and 62.

Kallenberg, O. (1997). *Foundations of modern probability*, volume 2. Springer. Cited on pages 31 and 32.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82(1):35–45. Cited on pages 24, 56, and 119.

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., and Chopin, N. (2015). On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351. Cited on page 76.

Karimi, H., Nutini, J., and Schmidt, M. (2016). Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In Frasconi, P., Landwehr, N., Manco, G., and Vreeken, J., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 795–811, Cham. Springer International Publishing. Cited on page 46.

Karjalainen, J., Lee, A., Singh, S. S., and Vihola, M. (2023). Mixing time of the conditional backward sampling particle filter. *arXiv preprint arXiv:2312.17572*. Cited on page 102.

Karppinen, S., Singh, S. S., and Vihola, M. (2023). Conditional particle filters with bridge backward sampling. *Journal of Computational and Graphical Statistics*, 0(0):1–15. Cited on pages 100 and 102.

Karppinen, S. and Vihola, M. (2021). Conditional particle filters with diffuse initial distributions. *Statistics and Computing*, 31:1–14. Cited on page 106.

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Cited on page 40.

Kingman, J. F. C. (1978). Uses of Exchangeability. *The Annals of Probability*, 6(2):183 – 197. Cited on page 31.

Kitagawa, G. (1994). The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623. Cited on pages 73 and 101.

Klass, M. and Teicher, H. (1987). The central limit theorem for exchangeable random variables without moments. *The Annals of Probability*, 15(1):138–153. Cited on page 32.

Kleijnen, J. P. and Rubinstein, R. Y. (1996). Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3):413–427. Cited on page 40.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press. Cited on pages 29 and 120.

Kool, W., van Hoof, H., and Welling, M. (2019). Buy 4 REINFORCE samples, get a baseline for free! Cited on page 42.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86. Cited on page 37.

Kuntz, J., Crucinio, F. R., and Johansen, A. M. (2022). Product-form estimators: exploiting independence to scale up Monte Carlo. *Statistics and Computing*, 32(1):12. Cited on page 122.

Kuntz, J., Crucinio, F. R., and Johansen, A. M. (2024). The divide-and-conquer sequential Monte Carlo algorithm: theoretical properties and limit theorems. *The Annals of Applied Probability*. Cited on pages 121, 122, 130, and 137.

Kviman, O., Branchini, N., Elvira, V., and Lagergren, J. (2024). Variational resampling. In *to appear in Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR. Cited on page 72.

Lambert, M., Chewi, S., Bach, F., Bonnabel, S., and Rigollet, P. (2022). Variational inference via Wasserstein gradient flows. Cited on pages 47 and 134.

Laplace, P. S. (1774). Mémoire sur la probabilité des causes par les évènements. *Mémoires de l'Académie Royale des Sciences de Paris*. Cited on page 36.

Łatuszyński, K. and Roberts, G. O. (2013). CLTs and asymptotic variance of time-sampled Markov chains. *Methodology and Computing in Applied Probability*, 15:237–247. Cited on page 85.

Le, T., Igl, M., Rainforth, T., Jin, T., and Wood, F. (2018). Auto-encoding sequential Monte Carlo. In *International Conference on Learning Representations (ICLR)*. OpenReview. Cited on pages 79, 80, and 125.

Lee, A., Singh, S. S., and Vihola, M. (2020). Coupled conditional backward sampling particle filter. *Annals of Statistics*, 48(5):3066–3089. Cited on pages 101, 102, and 126.

Lee, A. and Whiteley, N. (2016). Forest resampling for distributed sequential Monte Carlo. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 9(4):230–248. Cited on page 114.

Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2010). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789. Cited on page 112.

Li, Y., Wang, W., Deng, K. E., and Liu, J. S. (2021). Stratification and optimal resampling for sequential Monte Carlo. *Biometrika*, 109(1):181–194. Cited on page 71.

Lindgren, F., Bolin, D., and Rue, H. (2022). The SPDE approach for Gaussian and non-Gaussian fields: 10 years and still running. *Spatial Statistics*, 50:100599. Special Issue: The Impact of Spatial Statistics. Cited on page 61.

Lindsten, F., Douc, R., and Moulines, E. (2015). Uniform ergodicity of the particle Gibbs sampler. *Scandinavian Journal of Statistics*, 42(3):775–797. Cited on page 101.

Lindsten, F., Johansen, A. M., Naesseth, C. A., Kirkpatrick, B., Schön, T. B., Aston, J. A., and Bouchard-Côté, A. (2017). Divide-and-conquer with sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458. Cited on pages 120 and 122.

Lindsten, F., Jordan, M. I., and Schön, T. B. (2014). Particle Gibbs with ancestor sampling. *The Journal of Machine Learning Research*, 15(1):2145–2184. Cited on page 102.

Lindsten, F. and Schön, T. B. (2013). Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends in Machine Learning*, 6(1):1–143. Cited on page 102.

Liu, J. S. (1996). Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and computing*, 6:113–119. Cited on pages 69 and 86.

Liu, J. S. and Chen, R. (1995). Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576. Cited on page 69.

Livingstone, S. and Zanella, G. (2022). The Barker proposal: Combining robustness and efficiency in gradient-based MCMC. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(2):496–523. Cited on page 85.

Lotz, M. (2018). *Mathematics of machine learning*. Cited on page 41.

Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. W. (2017). Filtering variational objectives. In *Advances in Neural Information Processing Systems*. Cited on pages 79, 80, and 125.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*. Cited on page 42.

Malik, S. and Pitt, M. K. (2011a). Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*. Cited on pages 72, 75, and 134.

Malik, S. and Pitt, M. K. (2011b). Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, 165(2):190–209. Cited on pages 72, 75, and 76.

Malory, S. (2021). *Bayesian inference for stochastic processes*. PhD thesis, Lancaster University. Cited on page 104.

Manavski, S. A. and Valle, G. (2008). CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. *BMC bioinformatics*, 9:1–9. Cited on page 111.

Marie d'Avigneau, A., Singh, S. S., and Murray, L. M. (2021). Anytime parallel tempering. *Statistics and Computing*, 31:1–23. Cited on page 110.

Marinari, E. and Parisi, G. (1992). Simulated tempering: a new Monte Carlo scheme. *Europhysics letters*, 19(6):451. Cited on page 109.

Märtens, K., Titsias, M., and Yau, C. (2019). Augmented ensemble MCMC sampling in factorial Hidden Markov models. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2359–2367. PMLR. Cited on page 138.

Martin, G. M., Frazier, D. T., and Robert, C. P. (2023a). Approximating Bayes in the 21st century. *Statistical Science*, pages 1 – 26. Cited on pages 23 and 27.

Martin, G. M., Frazier, D. T., and Robert, C. P. (2023b). Computing Bayes: from then 'til now. *Statistical Science*, pages 1 – 17. Cited on pages 23 and 27.

Martino, L. (2018). A review of multiple try MCMC algorithms for signal processing. *Digital Signal Processing*, 75:134–152. Cited on page 92.

Maskell, S., Alun-Jones, B., and Macleod, M. (2006). A single instruction multiple data particle filter. In *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pages 51–54. Cited on page 111.

Meng, X.-L. and Van Dyk, D. (2002). The EM Algorithm—an Old Folk-song Sung to a Fast New Tune. *Journal of the Royal Statistical Society: Series B (Methodological)*, 59(3):511–567. Cited on page 66.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092. Cited on pages 36, 84, and 86.

Meyn, S. P. and Tweedie, R. L. (2009). *Markov chains and stochastic stability*. Cambridge University Press. Cited on pages 82 and 83.

Middleton, L., Deligiannidis, G., Doucet, A., and Jacob, P. E. (2019). Unbiased smoothing using particle independent Metropolis-Hastings. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2378–2387. PMLR. Cited on page 109.

Mider, M., Schauer, M., and Van der Meulen, F. (2021). Continuous-discrete smoothing of diffusions. *Electronic Journal of Statistics*, 15(2):4295–4342. Cited on page 132.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press. Cited on pages 30 and 40.

Murray, I., Ghahramani, Z., and MacKay, D. J. C. (2006). MCMC for doubly-intractable distributions. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 359–366. AUAI Press. Cited on page 29.

Murray, I. and Graham, M. (2016). Pseudo-marginal slice sampling. In *Artificial Intelligence and Statistics*, pages 911–919. PMLR. Cited on page 92.

Murray, L. (2012). Gpu acceleration of the particle filter: the metropolis resampler. Cited on pages 72, 112, and 114.

Murray, L. M., Lee, A., and Jacob, P. E. (2016). Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3):789–805. Cited on pages 114 and 130.

Møller, J., Pettitt, A. N., Reeves, R., and Berthelsen, K. K. (2006). An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, 93(2):451–458. Cited on page 29.

Naesseth, C., Linderman, S., Ranganath, R., and Blei, D. (2018). Variational sequential Monte Carlo. In *International conference on artificial intelligence and statistics*, pages 968–977. PMLR. Cited on pages 79, 80, and 125.

Naesseth, C. A., Lindsten, F., and Schön, T. B. (2019). Elements of sequential Monte Carlo. *arXiv preprint arXiv:1903.04797*. Cited on page 99.

Neal, R. M. (1996). Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6:353–366. Cited on page 110.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and computing*, 11:125–139. Cited on page 36.

Nemeth, C., Lindsten, F., Filippone, M., and Hensman, J. (2019). Pseudo-extended Markov chain Monte Carlo. *Advances in Neural Information Processing Systems*, 32. Cited on page 92.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer. Cited on page 65.

Osborne, E. E. (1960). On pre-conditioning of matrices. *Journal of the ACM*, 7(4):338–345. Cited on page 128.

Pachpatte, B. G. (1998). *Inequalities for differential and integral equations*, volume 197. Academic Press Limited. Cited on page 46.

Pagès, G. (2018). *Numerical Probability: An Introduction with Applications to Finance*. Springer International Publishing, Cham. Cited on page 36.

Papaspiliopoulos, O., Roberts, G. O., and Sköld, M. (2007). A General Framework for the Parametrization of Hierarchical Models. *Statistical Science*, 22(1):59 – 73. Cited on page 91.

Pasarica, C. and Gelman, A. (2010). Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, pages 343–364. Cited on pages 15, 136, and 137.

Pele, O. and Werman, M. (2009). Fast and robust Earth Mover's Distances. In *2009 IEEE 12th International Conference on Computer Vision*, pages 460–467. Cited on page 72.

Peskun, P. H. (1973). Optimum Monte-Carlo sampling using Markov chains. *Biometrika*, 60(3):607–612. Cited on page 136.

Pibiri, G. E. and Venturini, R. (2021). Practical trade-offs for the prefix-sum problem. *Software: Practice and Experience*, 51(5):921–949. Cited on page 111.

Poyiadjis, G., Doucet, A., and Singh, S. S. (2011). Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80. Cited on page 76.

Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252. Cited on page 109.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. MIT Press. Cited on pages 59, 60, and 108.

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450. Cited on page 58.

Reich, S. (2013). A nonparametric ensemble transform method for Bayesian inference. *SIAM Journal on Scientific Computing*, 35(4):A2013–A2024. Cited on pages 71, 72, 73, and 124.

Rényi, A. (1961). On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pages 547–562. University of California Press. Cited on page 38.

Rhee, C.-H. and Glynn, P. W. (2015). Unbiased estimation with square root convergence for SDE models. *Operations Research*, 63(5):1026–1043. Cited on page 109.

Rhodes, B. and Gutmann, M. (2022). Enhanced gradient-based MCMC in discrete spaces. *arXiv preprint arXiv:2208.00040*. Cited on page 138.

Robert, C. P. (1995). Simulation of truncated normal variables. *Statistics and computing*, 5(2):121–125. Cited on page 34.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120. Cited on page 86.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268. Cited on page 138.

Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363. Cited on page 87.

Roeder, G., Wu, Y., and Duvenaud, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, pages 6925–6934. Cited on page 41.

Roger, E. (1987). Stan Ulam, John Von Neumann, and the Monte Carlo Method. *Los Alamos Science*, (15):131–137. Cited on page 31.

Rosato, C., Devlin, L., Beraud, V., Horridge, P., Schön, T. B., and Maskell, S. (2022). Efficient learning of the parameters of non-linear models using differentiable resampling in particle filters. *IEEE Transactions on Signal Processing*, 70:3676–3692. Cited on page 126.

Rosenbluth, M. N. (2003). Genesis of the Monte Carlo Algorithm for Statistical Mechanics. *AIP Conference Proceedings*, 690(1):22–30. Cited on page 84.

Ruiz, F. R., Titsias, M., and Blei, D. (2016). The generalized reparameterization gradient. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc. Cited on page 41.

Santambrogio, F. (2017). Euclidean, metric, and Wasserstein gradient flows: An overview. *Bulletin of Mathematical Sciences*. Cited on page 47.

Särkkä, S. (2007). On unscented Kalman filtering for state estimation of continuous-time nonlinear systems. *IEEE Transactions on Automatic Control*. Cited on pages 48 and 134.

Särkkä, S. and García-Fernández, Á. F. (2021). Temporal parallelization of Bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306. Cited on pages 114, 116, 117, 118, 127, 128, and 139.

Särkkä, S. and García-Fernández, A. F. (2023). Temporal parallelization of dynamic programming and linear quadratic control. *IEEE Transactions on Automatic Control*, 68(2):851–866. Cited on pages 117, 119, and 120.

Särkkä, S. and Solin, A. (2019). *Applied Stochastic Differential Equations*. Cambridge University Press. Cited on pages 60 and 87.

Särkkä, S., Solin, A., and Hartikainen, J. (2013). Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing. *IEEE Signal Processing Magazine*, 30(4):51–61. Cited on pages 61 and 129.

Särkkä, S. and Svensson, L. (2020). Levenberg-Marquardt and line-search extended Kalman smoothers. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5875–5879. IEEE. Cited on page 65.

Särkkä, S. and Svensson, L. (2023). *Bayesian filtering and smoothing*, volume 17. Cambridge university press. Cited on pages 34, 43, 49, 56, 57, 63, and 67.

Sen, D., Thiery, A. H., and Jasra, A. (2018). On coupling particle filter trajectories. *Statistics and Computing*, 28:461–475. Cited on page 126.

Sherlock, C. and Thiery, A. H. (2022). A discrete bouncy particle sampler. *Biometrika*, 109(2):335–349. Cited on page 82.

Shi, J., Zhou, Y., Hwang, J., Titsias, M., and Mackey, L. (2022). Gradient estimation with discrete Stein operators. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25829–25841. Curran Associates, Inc. Cited on page 42.

Sisson, S. A., Fan, Y., and Beaumont, M. (2018). *Handbook of approximate Bayesian computation*. CRC Press. Cited on page 29.

Solin, A. (2016). *Stochastic Differential Equation Methods for Spatio-Temporal Gaussian Process Regression*. Doctoral dissertation, Aalto University, Helsinki, Finland. Cited on page 129.

Stein, M. L. (2012). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media. Cited on page 60.

Suchard, M. A. and Rambaut, A. (2009). Many-core algorithms for statistical phylogenetics. *Bioinformatics*, 25(11):1370–1376. Cited on page 111.

Swendsen, R. H. and Wang, J.-S. (1986). Replica Monte Carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607. Cited on page 110.

Syed, S., Bouchard-Côté, A., Deligiannidis, G., and Doucet, A. (2022). Non-reversible parallel tempering: A scalable highly parallel MCMC scheme. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(2):321–350. Cited on page 110.

Taroni, A. (2015). 90 years of the Ising model. *Nature Physics*, 11(12):997–997. Cited on page 29.

The Event Horizon Telescope Collaboration (2019). First M87 event horizon telescope results. i. the shadow of the supermassive black hole. *The Astrophysical Journal Letters*, 875(1):L1. Cited on page 110.

Thornton, C. L. (1976). *Triangular covariance factorizations for Kalman Filtering*. PhD thesis, California University. Cited on page 139.

Titsias, M. and Shi, J. (2022). Double control variates for gradient estimation in discrete latent variable models. In *International Conference on Artificial Intelligence and Statistics*, pages 6134–6151. PMLR. Cited on pages 42 and 126.

Titsias, M. K. and Papaspiliopoulos, O. (2018a). Auxiliary gradient-based sampling algorithms. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 80(4):749–767. Cited on pages 45, 89, 95, 96, 97, 131, and 133.

Titsias, M. K. and Papaspiliopoulos, O. (2018b). Auxiliary gradient-based sampling algorithms. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(4):749–767. Cited on page 136.

Tjelmeland, H. (2004). Using all Metropolis–Hastings proposals to estimate mean values. preprint 4/2004, Norwegian University of Science and Technology, Trondheim, Norway. Cited on pages 89, 92, and 95.

Tomasz Cakala, B. M. and Niemiro, W. (2021). Particle MCMC With Poisson resampling: Parallelization and continuous time models. *Journal of Computational and Graphical Statistics*, 30(3):671–684. Cited on page 114.

Tran, M. N., Kohn, R., Quiroz, M., and Villani, M. (2017). The block pseudo-marginal sampler. Cited on page 92.

Tronarp, F., García-Fernández, Á. F., and Särkkä, S. (2018). Iterative filtering and smoothing in nonlinear and non-Gaussian systems using conditional moments. *IEEE Signal Processing Letters*, 25(3):408–412. Cited on pages 43, 44, 64, 134, and 139.

Tsybakov, A. B. (2009). *Introduction to Nonparametric Estimation*. Springer New York, New York, NY. Cited on page 39.

Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the theory of the brownian motion. *Phys. Rev.*, 36:823–841. Cited on page 61.

Van Der Merwe, R., Doucet, A., De Freitas, N., and Wan, E. (2000). The unscented particle filter. *Advances in neural information processing systems*, 13. Cited on page 77.

Varsi, A., Maskell, S., and Spirakis, P. G. (2021). An o(log2 n) fully-balanced resampling algorithm for particle filters on distributed memory architectures. *Algorithms*, 14(12):342. Cited on page 114.

Vats, D., Gonçalves, F. B., Łatuszyński, K., and Roberts, G. O. (2021). Efficient Bernoulli factory Markov chain Monte Carlo for intractable posteriors. *Biometrika*, 109(2):369–385. Cited on page 85.

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., and Bürkner, P.-C. (2021). Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*, 16(2):667 – 718. Cited on page 84.

Vihola, M. (2018). Unbiased estimators and multilevel Monte Carlo. *Operations Research*, 66(2):448–462. Cited on page 109.

Villani, C. (2009). *Optimal transport: old and new*, volume 338. Springer. Cited on page 47.

Walker, A. M. (1969). On the asymptotic behaviour of posterior distributions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 31(1):80–88. Cited on page 37.

Whiteley, N. (2010). Discussion on particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):306–307. Cited on page 102.

Whiteley, N. and Lee, A. (2014). Twisted particle filters. *The Annals of Statistics*, 42(1):115 – 141. Cited on page 78.

Wigren, A., Risuleo, R. S., Murray, L., and Lindsten, F. (2019). Parameter elimination in particle Gibbs sampling. *Advances in Neural Information Processing Systems*, 32. Cited on page 138.

Wilkinson, W. J., Särkkä, S., and Solin, A. (2023). Bayes–Newton methods for approximate Bayesian inference with PSD guarantees. *Journal of Machine Learning Research*, 24(83):1–50. Cited on page 61.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256. Cited on page 40.

Yaghoobi, F., Corenflos, A., Hassan, S., and Särkkä, S. (2022). Parallel square-root statistical linear regression for inference in nonlinear state-space models. *arXiv preprint arXiv:2207.00426*. Cited on pages 65, 66, 127, and 132.

Yang, J., Roberts, G. O., and Rosenthal, J. S. (2020). Optimal scaling of random-walk Metropolis algorithms on general target distributions. *Stochastic Processes and their Applications*, 130(10):6094–6132. Cited on page 86.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., et al. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65. Cited on page 108.

Zanella, G. (2020). Informed proposals for local MCMC in discrete spaces. *Journal of the American Statistical Association*, 115(530):852–865. Cited on page 85.

Publication I

Adrien Corenflos, James Thornton, George Deligiannidis, Arnaud Doucet. Differentiable Particle Filtering via Entropy-Regularized Optimal Transport. In *Proceedings of the 38th International Conference on Machine Learning*, Volume 139, Pages 2100–2111, July 2021.

Differentiable Particle Filtering via Entropy-Regularized Optimal Transport

Adrien Corenflos [*1] James Thornton [*2] George Deligiannidis [2] Arnaud Doucet [2]

Abstract

Particle Filtering (PF) methods are an established class of procedures for performing inference in non-linear state-space models. Resampling is a key ingredient of PF, necessary to obtain low variance likelihood and states estimates. However, traditional resampling methods result in PF-based loss functions being non-differentiable with respect to model and PF parameters. In a variational inference context, resampling also yields high variance gradient estimates of the PF-based evidence lower bound. By leveraging optimal transport ideas, we introduce a principled differentiable particle filter and provide convergence results. We demonstrate this novel method on a variety of applications.

1. Introduction

In this section we provide a brief introduction to state-space models (SSMs) and PF methods. We then illustrate one of the well-known limitations of PF (Kantas et al., 2015): resampling steps are required in order to compute low-variance estimates, but these estimates are not differentiable w.r.t. to model and PF parameters. This hinders end-to-end training. We discuss recent approaches to address this problem in econometrics, statistics and machine learning (ML), outline their limitations and our contributions.

1.1. State-Space Models

SSMs are an expressive class of sequential models, used in numerous scientific domains including econometrics, ecology, ML and robotics; see e.g. (Chopin & Papaspiliopoulos, 2020; Douc et al., 2014; Doucet & Lee, 2018; Kitagawa & Gersch, 1996; Lindsten & Schön, 2013; Thrun et al., 2005). SSM may be characterized by a latent $\mathcal{X}$-valued Markov process $(X_t)_{t\geq 1}$ and $\mathcal{Y}$-valued observations $(Y_t)_{t\geq 1}$ satisfying $X_1 \sim \mu_\theta(\cdot)$ and for $t \geq 1$

$$X_{t+1}|\{X_t = x\} \sim f_\theta(\cdot|x),\ \ Y_t|\{X_t = x\} \sim g_\theta(\cdot|x), \quad (1)$$

where $\theta \in \Theta$ is a parameter of interest. Given observations $(y_t)_{t\geq 1}$ and parameter values θ, one may perform state inference at time t by computing the posterior of X_t given $y_{1:t} := (y_1, ..., y_t)$ where

$$p_\theta(x_t|y_{1:t-1}) = \int f_\theta(x_t|x_{t-1}) p_\theta(x_{t-1}|y_{1:t-1}) \mathrm{d}x_{t-1},$$

$$p_\theta(x_t|y_{1:t}) = \frac{g_\theta(y_t|x_t) p_\theta(x_t|y_{1:t-1})}{\int g_\theta(y_t|x_t) p_\theta(x_t|y_{1:t-1}) \mathrm{d}x_t},$$

with $p_\theta(x_1|y_0) := \mu_\theta(x_1)$.

The log-likelihood $\ell(\theta) = \log p_\theta(y_{1:T})$ is then given by

$$\ell(\theta) = \sum_{t=1}^{T} \log p_\theta(y_t|y_{1:t-1}),$$

with $p_\theta(y_1|y_0) := \int g_\theta(y_1|x_1)\mu_\theta(x_1)\mathrm{d}x_1$ and for $t \geq 2$

$$p_\theta(y_t|y_{1:t-1}) = \int g_\theta(y_t|x_t) p_\theta(x_t|y_{1:t-1}) \mathrm{d}x_t.$$

The posteriors $p_\theta(x_t|y_{1:t})$ and log-likelihood $p_\theta(y_{1:T})$ are available analytically for only a very restricted class of SSM such as linear Gaussian models. For non-linear SSM, PF provides approximations of such quantities.

1.2. Particle Filtering

PF are Monte Carlo methods entailing the propagation of N weighted particles $(w_t^i, X_t^i)_{i\in[N]}$, here $[N] := \{1, ..., N\}$, over time to approximate the filtering distributions $p_\theta(x_t|y_{1:t})$ and log-likelihood $\ell(\theta)$. Here $X_t^i \in \mathcal{X}$ denotes the value of the ith particle at time t and $\mathbf{w}_t := (w_t^1, ..., w_t^N)$ are weights satisfying $w_t^i \geq 0,\ \sum_{i=1}^N w_t^i = 1$. Unlike variational methods, PF methods provide consistent approximations under weak assumptions as $N \to \infty$ (Del Moral, 2004). In the general setting, particles are sampled according to proposal distributions $q_\phi(x_1|y_1)$ at time $t = 1$ and $q_\phi(x_t|x_{t-1}, y_t)$ at time $t \geq 2$ prior to weighting and resampling. One often chooses $\theta = \phi$ but this is not necessarily the case (Le et al., 2018; Maddison et al., 2017; Naesseth et al., 2018).

[*]Equal contribution [1]Department of Electrical Engineering and Automation, Aalto University [2]Department of Statistics, University of Oxford. Correspondence to: Adrien Corenflos <adrien.corenflos@aalto.fi>, James Thornton <james.thornton@spc.ox.ac.uk>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

Algorithm 1 Standard Particle Filter

```
1:  Sample $X_1^i \overset{\text{i.i.d.}}{\sim} q_\phi(\cdot|y_1)$  for $i \in [N]$
2:  Compute $\omega_1^i = \frac{p_\theta(X_1^i, y_1)}{q_\phi(X_1^i|y_1)}$ for $i \in [N]$
3:  $\hat{\ell}(\theta) \leftarrow \frac{1}{N}\sum_{i=1}^N \omega_1^i$
4:  **for** $t = 2, ..., T$ **do**
5:      Normalize weights $w_{t-1}^i \propto \omega_{t-1}^i$, $\sum_{i=1}^N w_{t-1}^i = 1$
6:      Resample $\tilde{X}_{t-1}^i \sim \sum_{i=1}^N w_{t-1}^i \delta_{X_{t-1}^i}$  for $i \in [N]$
7:      Sample $X_t^i \sim q_\phi(\cdot|\tilde{X}_{t-1}^i, y_t)$  for $i \in [N]$
8:      Compute $\omega_t^i = \frac{p_\theta(X_t^i, y_t|\tilde{X}_{t-1}^i)}{q_\phi(X_t^i|\tilde{X}_{t-1}^i, y_t)}$ for $i \in [N]$
9:      Compute $\hat{p}_\theta(y_t|y_{1:t-1}) = \frac{1}{N}\sum_{i=1}^N \omega_t^i$
10:     $\hat{\ell}(\theta) \leftarrow \hat{\ell}(\theta) + \log \hat{p}_\theta(y_t|y_{1:t-1})$
11: **end for**
12: **Return:** log-likelihood estimate $\hat{\ell}(\theta) = \log \hat{p}_\theta(y_{1:T})$
```

A generic PF is described in Algorithm 1 where $p_\theta(x_1, y_1) := \mu_\theta(x_1)g_\theta(y_1|x_1)$ and $p_\theta(x_t, y_t|x_{t-1}) := f_\theta(x_t|x_{t-1})g_\theta(y_t|x_t)$. Resampling is performed in step 6 of Algorithm 1; it ensures particles with high weights are replicated and those with low weights are discarded, allowing one to focus computational efforts on 'promising' regions. The scheme used in Algorithm 1 is known as multinomial resampling and is unbiased (as are other traditional schemes such as stratified and systematic (Chopin & Papaspiliopoulos, 2020)), i.e.

$$\mathbb{E}\left[\tfrac{1}{N}\textstyle\sum_{i=1}^N \psi(\tilde{X}_t^i)\right] = \mathbb{E}\left[\textstyle\sum_{i=1}^N w_t^i \psi(X_t^i)\right], \qquad (2)$$

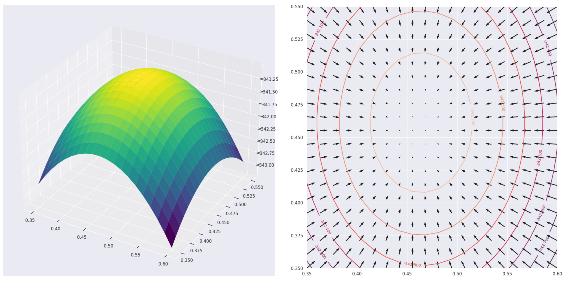
for any $\psi : \mathcal{X} \to \mathbb{R}$. This property guarantees $\exp(\hat{\ell}(\theta))$ is an unbiased estimate of the likelihood $\exp(\ell(\theta))$ for any N.

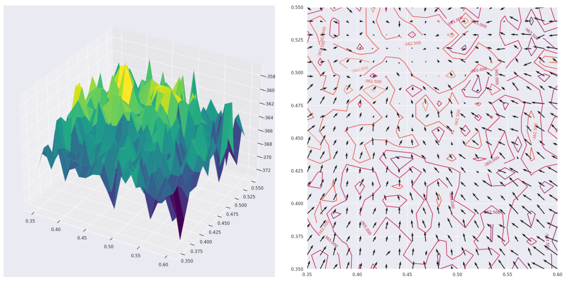
Henceforth, let $\mathcal{X} = \mathbb{R}^{d_x}$, $\theta \in \Theta = \mathbb{R}^{d_\theta}$ and $\phi \in \Phi = \mathbb{R}^{d_\phi}$. We assume here that $\theta \mapsto \mu_\theta(x)$, $\theta \mapsto f_\theta(x'|x)$ and $\theta \mapsto g_\theta(y_t|x)$ are differentiable for all x, x' and $t \in [T]$ and $\theta \mapsto \ell(\theta)$ is differentiable. These assumptions are satisfied by a large class of SSMs. We also assume that we can use the reparameterization trick (Kingma & Welling, 2014) to sample the particles; i.e. we have $\Gamma_\phi(y_1, U) \sim q_\phi(x_1|y_1)$, $\Psi_\phi(y_t, x_{t-1}, U) \sim q_\phi(x_t|x_{t-1}, y_t)$ for some mappings Γ_ϕ, Ψ_ϕ differentiable w.r.t. ϕ and $U \sim \lambda$, λ being independent of ϕ.

1.3. Related Work and Contributions

Let $\mathbf{U}$ be the set of all random variables used to sample and resample the particles. The distribution of $\mathbf{U}$ is (θ, ϕ)-independent as we use the reparameterization trick[1]. However, even if we sample and fix $\mathbf{U} = \mathbf{u}$, resampling involves sampling from an atomic distribution and introduces discontinuities in the particles selected when θ, ϕ vary.

[1]For example, multinomial resampling relies on N uniform random variables.


(a) Kalman Filter


(b) Standard PF

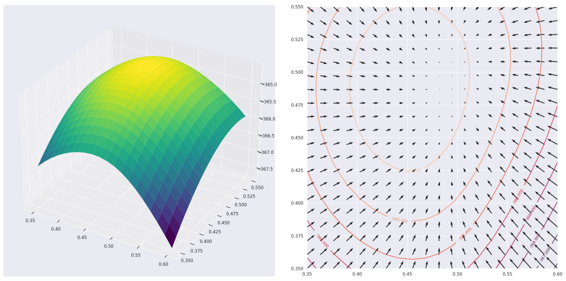

(c) Differentiable PF

Figure 1. Left: Log-likelihood $\ell(\theta)$ and PF estimates $\hat{\ell}(\theta; \phi, \mathbf{u})$ for linear Gaussian SSM, given in Section 5.1, with $d_\theta = 2\, d_x = 2$, and $T = 150, N = 50$. Right: $\nabla_\theta \ell(\theta)$ and $\nabla_\theta \hat{\ell}(\theta; \phi, \mathbf{u})$.

For $d_x = 1$, Malik & Pitt (2011) make $\theta \mapsto \hat{\ell}(\theta; \phi, \mathbf{u})$ continuous w.r.t. θ by sorting the particles and then sampling from a smooth approximation of their cumulative distribution function. For $d_x > 1$, Lee (2008) proposes a smoother but only piecewise continuous estimate. DeJong et al. (2013) returns a differentiable log-likelihood estimate $\hat{\ell}(\theta; \phi, \mathbf{u})$ by using a marginal PF (Klaas et al., 2005), where importance sampling is performed on a collapsed state-space. However, the proposal distribution typically used in the marginal PF is the mixture distribution $q_\phi(x_t) := \frac{1}{N}\sum_{i=1}^N q_\phi(x_t|\tilde{X}_{t-1}^i, y_t)$ from which one cannot sample smoothly in general. As a consequence they instead suggest using a simple Gaussian distribution for $q_\phi(x_t)$, which can lead to poor estimates for multimodal posteriors. Moreover, in contrast to standard PF, this marginal PF cannot be applied in scenarios where the transition density can be sampled from (e.g. using the reparameterization trick) but not evaluated pointwise (Murray et al., 2013), as the importance weight would be intractable.

In the context of robot localization, a modified resampling scheme has been proposed in (Karkus et al., 2018; Ma et al., 2020a;b) referred to as 'soft-resampling' (SPF). SPF has parameter $\alpha \in [0, 1]$ where $\alpha = 1$ corresponds to regular PF resampling and $\alpha = 0$ is essentially sampling particles uniformly at random. The resulting PF-net is said to be differentiable but computes gradients that ignore the non-differentiable component of the resampling step. Jonschkowski et al. (2018) proposed another PF scheme which is said to be differentiable but simply ignores the non-differentiable resampling terms and proposes new states based on the observation and some neural network. This approach however does not propagate gradients through time. Finally, Zhu et al. (2020) propose a differentiable resampling scheme based on transformers but they report that the best results are achieved when not backpropagating through it, due to exploding gradients. Hence no fully differentiable PF is currently available in the literature (Kloss et al., 2020).

PF methods have also been fruitfully exploited in Variational Inference (VI) to estimate θ, ϕ (Le et al., 2018; Maddison et al., 2017; Naesseth et al., 2018). As $\mathbb{E}_{\mathbf{U}}[\exp(\hat{\ell}(\theta; \phi, \mathbf{U}))] = \exp(\ell(\theta))$ is an unbiased estimate of $\exp(\ell(\theta))$ for any N, ϕ for standard PF, then one has indeed by Jensen's inequality

$$\ell^{\mathrm{ELBO}}(\theta, \phi) := \mathbb{E}_{\mathbf{U}}[\hat{\ell}(\theta; \phi, \mathbf{U})] \leq \ell(\theta). \tag{3}$$

The standard ELBO corresponds to $N = 1$ and many variational families for approximating $p_\theta(x_{1:T}|y_{1:T})$ have been proposed in this context (Archer et al., 2015; Krishnan et al., 2017; Rangapuram et al., 2018). The variational family induced by a PF differs significantly as $\ell^{\mathrm{ELBO}}(\theta, \phi) \to \ell(\theta)$ as $N \to \infty$ and thus yields a variational approximation converging to $p_\theta(x_{1:T}|y_{1:T})$. This attractive property comes at a computational cost; i.e. the PF approach trades off fidelity to the posterior with computational complexity. While unbiased gradient estimates of the PF-ELBO (3) can be computed, they suffer from high variance as the resampling steps require having to use REINFORCE gradient estimates (Williams, 1992). Consequently, Hirt & Dellaportas (2019); Le et al. (2018); Maddison et al. (2017); Naesseth et al. (2018) use biased gradient estimates which ignore these terms, yet report improvements as N increases over standard VI approaches and Importance Weighted Auto-Encoders (IWAE) (Burda et al., 2016).

The contributions of this paper are four-fold.

- We propose the first fully Differentiable Particle Filter (DPF), which unlike (DeJong et al., 2013), can use general proposal distributions. DPF provides a differentiable estimate of $\ell(\theta)$, see Figure 1-c, and more generally differentiable estimates of PF-based losses. Empirically, in a VI context, DPF-ELBO gradient estimates also exhibit much smaller variance than those of PF-ELBO.
- We provide quantitative convergence results on the differentiable resampling scheme and establish consistency results for DPF.
- We show that existing techniques provide inconsistent gradient estimates and that the non-vanishing bias can be very significant, leading practically to unreliable parameter estimates.
- We demonstrate that DPF empirically outperforms recent alternatives for end-to-end parameter estimation on a variety of applications.

Proofs of results are given in the Supplementary Material.

2. Resampling via Optimal Transport

2.1. Optimal Transport and the Wasserstein Metric

Since Optimal Transport (OT) (Peyré & Cuturi, 2019; Villani, 2008) is a core component of our scheme, the basics are presented here. Given two probability measures α, β on $\mathcal{X} = \mathbb{R}^{d_x}$ the squared 2-Wasserstein metric between these measures is given by

$$\mathcal{W}_2^2(\alpha, \beta) = \min_{\mathcal{P} \in \mathcal{U}(\alpha, \beta)} \mathbb{E}_{(U,V)\sim\mathcal{P}}\left[||U - V||^2\right], \tag{4}$$

where $\mathcal{U}(\alpha, \beta)$ the set of distributions on $\mathcal{X} \times \mathcal{X}$ with marginals α and β, and the minimizing argument of (4) is the OT plan denoted $\mathcal{P}^{\mathrm{OT}}$. Any element $\mathcal{P} \in \mathcal{U}(\alpha, \beta)$ allows one to "transport" α to β (and vice-versa) i.e.

$$\beta(\mathrm{d}v) = \int \mathcal{P}(\mathrm{d}u, \mathrm{d}v) = \int \mathcal{P}(\mathrm{d}v|u)\alpha(\mathrm{d}u).$$

For atomic probability measures $\alpha_N = \sum_{i=1}^N a_i \delta_{u_i}$ and $\beta_N = \sum_{j=1}^N b_j \delta_{v_j}$ with weights $\mathbf{a} = (a_i)_{i\in[N]}$, $\mathbf{b} = (b_j)_{j\in[N]}$, and atoms $\mathbf{u} = (u_i)_{i\in[N]}$, $\mathbf{v} = (v_j)_{j\in[N]}$, one can show that

$$\mathcal{W}_2^2(\alpha_N, \beta_N) = \min_{\mathbf{P}\in\mathcal{S}(\mathbf{a},\mathbf{b})} \textstyle\sum_{i=1}^N \sum_{j=1}^N c_{i,j} p_{i,j}, \tag{5}$$

where any $\mathcal{P} \in \mathcal{U}(\alpha_N, \beta_N)$ is of the form

$$\mathcal{P}(\mathrm{d}u, \mathrm{d}v) = \textstyle\sum_{i,j} p_{i,j} \delta_{u_i}(\mathrm{d}u)\delta_{v_j}(\mathrm{d}v),$$

$c_{i,j} = ||u_i - v_j||^2$, $\mathbf{P} = (p_{i,j})_{i,j\in[N]}$ and $\mathcal{S}(\mathbf{a}, \mathbf{b})$ is the following set of matrices

$$\mathcal{S}(\mathbf{a}, \mathbf{b}) = \left\{\mathbf{P} \in [0, 1]^{N\times N} : \sum_{j=1}^N p_{i,j} = a_i, \; \sum_{i=1}^N p_{i,j} = b_j\right\}.$$

In such cases, one has

$$\mathcal{P}(\mathrm{d}v|u = u_i) = \textstyle\sum_j a_i^{-1} p_{i,j} \delta_{v_j}(\mathrm{d}v). \tag{6}$$

The optimization problem (5) may be solved through linear programming. It is also possible to exploit the dual formulation

$$\mathcal{W}_2^2(\alpha_N, \beta_N) = \max_{\mathbf{f},\mathbf{g}\in\mathcal{R}(C)} \mathbf{a}^t\mathbf{f} + \mathbf{b}^t\mathbf{g}, \tag{7}$$

where $\mathbf{f} = (f_i)$, $\mathbf{g} = (g_i)$, $\mathbf{C} = (c_{i,j})$ and $\mathcal{R}(\mathbf{C}) = \{\mathbf{f}, \mathbf{g} \in \mathbb{R}^N | f_i + g_j \leq c_{i,j}, i, j \in [N]\}$.

2.2. Ensemble Transform Resampling

The use of OT for resampling in PF has been pioneered by Reich (2013). Unlike standard resampling schemes (Chopin & Papaspiliopoulos, 2020; Doucet & Lee, 2018), it relies not only on the particle weights but also on their locations.

At time t, after the sampling step (Step 7 in Algorithm 1), $\alpha_N^{(t)} = \frac{1}{N}\sum_{i=1}^N \delta_{X_t^i}$ is a particle approximation of $\alpha^{(t)} := \int q_\phi(x_t|x_{t-1}, y_t) p_\theta(x_{t-1}|y_{1:t-1}) \mathrm{d}x_{t-1}$ and $\beta_N^{(t)} = \sum w_t^i \delta_{X_t^i}$ is an approximation of $\beta^{(t)} := p_\theta(x_t|y_{1:t})$. Under mild regularity conditions, the OT plan minimizing $\mathcal{W}_2(\alpha^{(t)}, \beta^{(t)})$ is of the form $\mathcal{P}^{\mathrm{OT}}(\mathrm{d}x, \mathrm{d}x') = \alpha^{(t)}(\mathrm{d}x)\delta_{\mathbf{T}^{(t)}(x)}(\mathrm{d}x')$ where $\mathbf{T}^{(t)} : \mathcal{X} \to \mathcal{X}$ is a deterministic map; i.e if $X \sim \alpha^{(t)}$ then $\mathbf{T}^{(t)}(X) \sim \beta^{(t)}$. It is shown in (Reich, 2013) that one can one approximate this transport map with the 'Ensemble Transform' (ET) denoted $\mathbf{T}_N^{(t)}$. This is found by solving the OT problem (5) between $\alpha_N^{(t)}$ and $\beta_N^{(t)}$ and taking an expectation w.r.t. (6), that is

$$\tilde{X}_t^i = N\textstyle\sum_{k=1}^N p_{i,k}^{\mathrm{OT}} X_t^k := \mathbf{T}_N^{(t)}(X_t^i), \tag{8}$$

where we slightly abuse notation as $\mathbf{T}_N^{(t)}$ is a function of $X_t^{1:N}$. Reich (2013) uses this update instead of using $\tilde{X}_t^i \sim \sum_{i=1}^N w_t^i \delta_{X_t^i}$. This is justified by the fact that, as $N \to \infty$, $\mathbf{T}_N^{(t)}(X_t^i) \to \mathbf{T}^{(t)}(X_t^i)$ in some weak sense (Reich, 2013; Myers et al., 2021). Compared to standard resampling schemes, the ET only satisfies (2) for affine functions ψ.

This OT approach to resampling involves solving the linear program (4) at cost $O(N^3 \log N)$ (Bertsimas & Tsitsiklis, 1997). This is not only prohibitively expensive but moreover the resulting ET is not differentiable. To address these problems, one may instead rely on entropy-regularized OT (Cuturi, 2013).

3. Differentiable Resampling via Entropy-Regularized Optimal Transport

3.1. Entropy-Regularized Optimal Transport

Entropy-regularized OT may be used to compute a transport matrix that is differentiable with respect to inputs and computationally cheaper than the non-regularized version, i.e. we consider the following regularized version of (5) for some $\epsilon > 0$ (Cuturi, 2013; Peyré & Cuturi, 2019)

$$\mathcal{W}_{2,\epsilon}^2(\alpha_N, \beta_N) = \min_{\mathbf{P} \in \mathcal{S}(\mathbf{a},\mathbf{b})} \sum_{i,j=1}^N p_{i,j}\left(c_{i,j} + \epsilon \log \frac{p_{i,j}}{a_i b_j}\right). \tag{9}$$

The function minimized in (9) is strictly convex and hence admits a unique minimizing argument $\mathbf{P}_\epsilon^{\mathrm{OT}} = (p_{\epsilon,i,j}^{\mathrm{OT}})$. $\mathcal{W}_{2,\epsilon}^2(\alpha_N, \beta_N)$ can also be computed using the regularized dual; i.e. $\mathcal{W}_{2,\epsilon}^2(\alpha_N, \beta_N) = \max_{\mathbf{f},\mathbf{g}} \mathrm{DOT}_\epsilon(\mathbf{f}, \mathbf{g})$ with

$$\mathrm{DOT}_\epsilon(\mathbf{f}, \mathbf{g}) := \mathbf{a}^t\mathbf{f} + \mathbf{b}^t\mathbf{g} - \epsilon\mathbf{a}^t\mathbf{M}\mathbf{b} \tag{10}$$

where $(\mathbf{M})_{i,j} := \exp\big(\epsilon^{-1}(f_i + g_j - c_{i,j})\big) - 1$ and $\mathbf{f}, \mathbf{g}$ are now unconstrained. For the dual pair $(\mathbf{f}^*, \mathbf{g}^*)$ maximizing (10), we have $\nabla_{\mathbf{f},\mathbf{g}} \mathrm{DOT}_\epsilon(\mathbf{f}, \mathbf{g})|_{(\mathbf{f}^*,\mathbf{g}^*)} = \mathbf{0}$. This first-order condition leads to

$$f_i^* = \mathcal{T}_\epsilon(\mathbf{b}, \mathbf{g}^*, \mathbf{C}_{i:}), \qquad g_i^* = \mathcal{T}_\epsilon(\mathbf{a}, \mathbf{f}^*, \mathbf{C}_{:i}), \tag{11}$$

where $\mathbf{C}_{i:}$ (resp. $\mathbf{C}_{:i}$) is the i^{th} row (resp. column) of $\mathbf{C}$. Here $\mathcal{T}_\epsilon : \mathbb{R}^N \times \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N$ denotes the mapping

$$\mathcal{T}_\epsilon(\mathbf{a}, \mathbf{f}, \mathbf{C}_{:,i}) = -\epsilon \log \sum_k \exp\left\{ \log a_k + \epsilon^{-1}(f_k - c_{k,i}) \right\}.$$

One may then recover the regularized transport matrix as

$$p_{\epsilon,i,j}^{\mathrm{OT}} = a_i b_j \exp\left(\epsilon^{-1}(f_i^* + g_j^* - c_{i,j})\right). \tag{12}$$

The dual can be maximized using the Sinkhorn algorithm introduced for OT in the seminal paper of Cuturi (2013). Algorithm 2 presents the implementation of Feydy et al. (2019) where the fixed point updates based on Equation (11) have been stabilized.

Algorithm 2 Sinkhorn Algorithm

```
1:  Function Potentials($\mathbf{a}, \mathbf{b}, \mathbf{u}, \mathbf{v}$)
2:  Local variables: $\mathbf{f}, \mathbf{g} \in \mathbb{R}^N$
3:  Initialize: $\mathbf{f} = \mathbf{0}, \mathbf{g} = \mathbf{0}$
4:  Set $\mathbf{C} \leftarrow \mathbf{u}\mathbf{u}^t + \mathbf{v}\mathbf{v}^t - 2\mathbf{u}\mathbf{v}^t$
5:  while stopping criterion not met do
6:    for $i \in [N]$ do
7:      $f_i \leftarrow \frac{1}{2}(f_i + \mathcal{T}_\epsilon(\mathbf{b}, \mathbf{g}, \mathbf{C}_{i:}))$
8:      $g_i \leftarrow \frac{1}{2}(g_i + \mathcal{T}_\epsilon(\mathbf{a}, \mathbf{f}, \mathbf{C}_{:i}))$
9:    end for
10: end while
11: Return $\mathbf{f}, \mathbf{g}$
```

The resulting dual vectors $(\mathbf{f}^*, \mathbf{g}^*)$ can then be differentiated for example using automatic differentiation through the Sinkhorn algorithm loop (Flamary et al., 2018), or more efficiently using "gradient stitching" on the dual vectors at convergence, which we do here (see Feydy et al. (2019) for details). The derivatives of $\mathbf{P}_\epsilon^{\mathrm{OT}}$ are readily accessible by combining the derivatives of (11) with the derivatives of (12), using automatic differentiation at no additional cost.

3.2. Differentiable Ensemble Transform Resampling

We obtain a differentiable ET (DET), denoted $\mathbf{T}_{N,\epsilon}^{(t)}$, by computing the entropy-regularized OT using Algorithm 3 for the weighted particles $(\mathbf{X}_t, \mathbf{w}_t, N)$ at time t

$$\tilde{X}_t^i = N\textstyle\sum_{k=1}^N p_{\epsilon,i,k}^{\mathrm{OT}} X_t^k := \mathbf{T}_{N,\epsilon}^{(t)}(X_t^i). \tag{13}$$

Algorithm 3 DET Resampling

```
1: Function EnsembleTransform($\mathbf{X}, \mathbf{w}, N$)
2: $\mathbf{f}, \mathbf{g} \leftarrow$ Potentials($\mathbf{w}, \frac{1}{N}\mathbf{1}, \mathbf{X}, \mathbf{X}$)
3: for $i \in [N]$ do
4:     for $j \in [N]$ do
5:         $p^{\mathrm{OT}}_{\epsilon,i,j} = \frac{w_i}{N} \exp\left(\frac{f_i + g_j - c_{i,j}}{\epsilon}\right)$
6:     end for
7: end for
8: Return $\tilde{\mathbf{X}} = N\mathbf{P}^{\mathrm{OT}}_{\epsilon}\mathbf{X}$
```

Compared to the ET, the DET is differentiable and can be computed at cost $O(N^2)$ as it relies on the Sinkhorn algorithm. This algorithm converges quickly (Altschuler et al., 2017) and is particularly amenable to GPU implementation.

The DPF proposed in this paper is similar to Algorithm 1 except that we sample from the proposal q_ϕ using the reparameterization trick and Step 6 is replaced by the DET. While such a differentiable approximation of the ET has previously been suggested in ML (Cuturi & Doucet, 2014; Seguy et al., 2018), it has never been realized before that this could be exploited to obtain a DPF. In particular, we obtain differentiable estimates of expectations w.r.t. the filtering distributions with respect to θ and ϕ and, for a fixed "seed" $\mathbf{U} = \mathbf{u}$[2], we obtain a differentiable estimate of the log-likelihood function $\theta \mapsto \hat{\ell}_\epsilon(\theta; \phi, \mathbf{u})$.

Like ET, DET only satisfies (2) for affine functions ψ. Unlike $\mathbf{P}^{\mathrm{OT}}$, $\mathbf{P}^{\mathrm{OT}}_{\epsilon}$ is sensitive to the scale of $\mathbf{X}_t$. To mitigate this sensitivity, one may compute $\delta(\mathbf{X}_t) = \sqrt{d_x}\max_{k\in[d_x]} \mathrm{std}_i(X^i_{t,k})$ for $\mathbf{X}_t \in \mathbb{R}^{N\times d_x}$ and rescale $\mathbf{C}$ accordingly to ensure that ϵ is approximately independent of the scale and dimension of the problem.

4. Theoretical Analysis

We show here that the gradient estimates of PF-based losses ignoring gradients terms due to resampling are not consistent and can suffer from a large non-vanishing bias. On the contrary, we establish that DPF provides consistent and differentiable estimates of the filtering distributions and log-likelihood function. This is achieved by obtaining novel quantitative convergence results for the DET.

4.1. Gradient Bias from Ignoring Resampling Terms

We first provide theoretical results on the asymptotic bias of the gradient estimates computed from PF-losses, by dropping the gradient terms from resampling, as adopted in (Hirt & Dellaportas, 2019; Jonschkowski et al., 2018; Karkus et al., 2018; Le et al., 2018; Ma et al., 2020b; Maddison et al., 2017; Naesseth et al., 2018). We limit ourselves here to the ELBO loss. Similar analysis can be carried out for the non-differentiable resampling schemes and losses considered in robotics.

[2]Here $\mathbf{U}$ denotes only the set of θ, ϕ-independent random variables used to generate particles as, contrary to standard PF, DET resampling does not rely on any additional random variable.

Proposition 4.1. *Consider the PF in Algorithm 1 where ϕ is distinct from θ then, under regularity conditions, the expectation of the ELBO gradient estimate $\hat{\nabla}_\theta \ell^{\mathrm{ELBO}}(\theta, \phi)$ ignoring resampling terms considered in (Le et al., 2018; Maddison et al., 2017; Naesseth et al., 2018) converges as $N \to \infty$ to*

$$\mathbb{E}[\hat{\nabla}_\theta \ell^{\mathrm{ELBO}}(\theta,\phi)] \to \int \nabla_\theta \log p_\theta(x_1, y_1)\, p_\theta(x_1|y_1)\mathrm{d}x_1 + \sum_{t=2}^{T} \int \nabla_\theta \log p_\theta(x_t, y_t|x_{t-1})\, p_\theta(x_{t-1:t}|y_{1:t})\mathrm{d}x_{t-1:t}$$

whereas Fisher's identity yields

$$\nabla_\theta \ell(\theta) = \int \nabla_\theta \log p_\theta(x_1, y_1)\, p_\theta(x_1|y_{1:T})\mathrm{d}x_1 + \sum_{t=2}^{T} \int \nabla_\theta \log p_\theta(x_t, y_t|x_{t-1})\, p_\theta(x_{t-1:t}|y_{1:T})\mathrm{d}x_{t-1:t}.$$

Hence, whereas we have $\nabla_\theta \ell^{\mathrm{ELBO}}(\theta,\phi) \to \nabla_\theta \ell(\theta)$ as $N \to \infty$ under regularity assumptions, the asymptotic bias of $\hat{\nabla}_\theta \ell^{\mathrm{ELBO}}(\theta,\phi)$ only vanishes if $p_\theta(x_{t-1:t}|y_{1:t}) = p_\theta(x_{t-1:t}|y_{1:T})$; i.e. for models where the X_t are independent. When $y_{t+1:T}$ do not bring significant information about X_t given $y_{t:T}$, as for the models considered in (Le et al., 2018; Maddison et al., 2017; Naesseth et al., 2018), this is a reasonable approximation which explains the good performance reported therein. However, we show in Section 5 that this bias can also lead practically to inaccurate parameter estimation.

4.2. Quantitative Bounds on the DET

Weak convergence results for the ET have been established in (Reich, 2013; Myers et al., 2021) and the DET in (Seguy et al., 2018). We provide here the first quantitative bound for the ET ($\epsilon = 0$) and DET ($\epsilon > 0$) which holds for any $N \geq 1$ by building upon results of (Li & Nochetto, 2021) and (Weed, 2018). We use the notation $\nu(\psi) := \int \psi(x)\nu(\mathrm{d}x)$ for any measure ν and function ψ.

Proposition 4.2. *Consider atomic probability measures $\alpha_N = \sum_{i=1}^N a_i \delta_{Y^i}$ with $a_i > 0$ and $\beta_N = \sum_{i=1}^N b_i \delta_{X^i}$, with support $\mathcal{X} \subset \mathbb{R}^d$. Let $\tilde{\beta}_N = \sum_{i=1}^N a_i \delta_{\tilde{X}^i_{N,\epsilon}}$ where $\tilde{\mathbf{X}}_{N,\epsilon} = \Delta^{-1}\mathbf{P}^{\mathrm{OT}}_{\epsilon}\mathbf{X}$ for $\Delta = \mathrm{diag}(a_1, ..., a_N)$ and $\mathbf{P}^{\mathrm{OT}}_{\epsilon}$ is the transport matrix corresponding to the ϵ-regularized OT coupling, $\mathcal{P}^{\mathrm{OT},N}_{\epsilon}$, between α_N and β_N. Let α, β be two other probability measures, also supported on $\mathcal{X}$, such that there exists a unique λ-Lipschitz optimal transport map $\mathbf{T}$*

between them. Then for any bounded 1*-Lipschitz function* ψ*, we have*

$$\left|\beta_N(\psi) - \tilde{\beta}_N(\psi)\right| \leq 2\lambda^{1/2}\mathcal{E}^{1/2}\left[\mathfrak{d}^{1/2} + \mathcal{E}\right]^{1/2} + \max\{\lambda, 1\}\left[\mathcal{W}_2(\alpha_N, \alpha) + \mathcal{W}_2(\beta_N, \beta)\right], \quad (14)$$

where $\mathfrak{d} := \sup_{x,y\in\mathcal{X}} |x - y|$ *and* $\mathcal{E} = \mathcal{W}_2(\alpha_N, \alpha) + \mathcal{W}_2(\beta_N, \beta) + \sqrt{2\epsilon \log N}$.

If $\mathcal{W}_2(\alpha_N, \alpha), \mathcal{W}_2(\beta_N, \beta) \to 0$ and we choose $\epsilon_N = o(1/\log N)$ the bound given in (14) vanishes with $N \to \infty$. This suggested dependence of ϵ on N comes from the entropic radius, see Lemma C.1 in the Supplementary and (Weed, 2018), and is closely related to the fact that entropy-regularized OT is sensitive to the scale of $\mathbf{X}$. Equivalently one may rescale $\mathbf{X}$ by a factor $\log N$ when computing the cost matrix. In particular when α_N and β_N are Monte Carlo approximations of α and β, we expect $\mathcal{W}_2(\alpha_N, \alpha), \mathcal{W}_2(\beta_N, \beta) = O(N^{-1/d})$ with high probability (Fournier & Guillin, 2015).

4.3. Consistency of DPF

The parameters θ, ϕ are here fixed and omitted from notation. We now establish consistency results for DPF, showing that both the resulting particle approximations $\tilde{\beta}_N^{(t)} = \frac{1}{N}\sum_{i=1}^N \delta_{\tilde{X}_t^i}$ of $\beta^{(t)} = p(x_t|y_{1:t})$ and the corresponding log-likelihood approximation $\log \hat{p}_N(y_{1:T})$ of $\log p(y_{1:T})$ are consistent. In the interest of simplicity, we limit ourselves to the scenario where the proposal is the transition, $q = f$, so $\omega(x_{t-1}, x_t, y_t) = g(y_t|x_t)$, known as the bootstrap PF and study a slightly non-standard version of it proposed in (Del Moral & Guionnet, 2001); see Appendix D for details. Consistency is established under regularity assumptions detailed in the Supplementary. Assumption B.1 is that the space $\mathcal{X} \subset \mathbb{R}^d$ has a finite diameter $\mathfrak{d}$. Assumption B.2 implies that the proposal mixes exponentially fast in the Wasserstein sense at a rate κ, which is reasonable given compactness, and essential for the error to not accumulate. Assumption B.3 assumes a bounded importance weight function i.e. $g(y_t|x_t) \in [\Delta, \Delta^{-1}]$, again not unreasonable given compactness. Assumption B.4 states that at each time step, the optimal transport problem between $\alpha^{(t)}$ and $\beta^{(t)}$ is solved uniquely by a deterministic, globally Lipschitz map. Uniqueness is crucial for the quantitative stability results provided in the following proposition.

Proposition 4.3. *Under Assumptions B.1, B.2, B.3 and B.4, for any* $\delta > 0$*, with probability at least* $1 - 2\delta$ *over the sampling steps, for any bounded 1-Lipschitz* ψ*, for any* $t \in [1 : T]$*, the approximations of the filtering distributions and log-likelihood computed by the bootstrap DPF satisfy*

$$|\tilde{\beta}_N^{(t)}(\psi) - \beta^{(t)}(\psi)| \leq \mathfrak{G}_{\epsilon,\delta/T,N,d}^{(t)}\left(\lambda(c, C, d, T, N, \delta)\right),$$

$$\left|\log \frac{\hat{p}_N(y_{1:T})}{p(y_{1:T})}\right| \leq \frac{\kappa}{\Delta} \max_{t\in[1:T]} \operatorname{Lip}\left[g(y_t \mid \cdot)\right] \times \sum_{t=1}^T \mathfrak{G}_{\epsilon,\delta/T,N,d}^{(t)}\left(\lambda(c, C, d, T, N, \delta)\right),$$

for $\lambda(c, C, d, T, N, \delta) = \sqrt{f_d^{-1}\left(\frac{\log(CT/\delta)}{cN}\right)}$ *where* c, C *are finite constants independent of* T*, and* $\operatorname{Lip}[f]$ *is the Lipschitz constant of the function* f*, and* $\mathfrak{G}_{N,\epsilon}^{(t)}$*,* f_d *defined in Appendix D are two functions such that if we set* $\epsilon_N = o(1/\log N)$ *then we have in probability*

$$|\tilde{\beta}_N^{(t)}(\psi) - \beta^{(t)}(\psi)| \to 0, \qquad \left|\log \frac{\hat{p}_N(y_{1:T})}{p(y_{1:T})}\right| \to 0.$$

The above bounds are certainly not sharp. A glimpse into the behavior of the above bounds in terms of T can be obtained through careful consideration of the quantities appearing in Proposition D.1 in the supplement. In particular, for κ small enough, it suggests that the bound on the error of the log-likelihood estimator grows linearly with T as for standard PF under mixing assumptions. Sharper bounds are certainly possible, e.g. using a L_1 version of Theorem 3.5 in (Li & Nochetto, 2021). It would also be of interest to weaken the assumptions, in particular, to remove the bounded space assumption although it is very commonly made in the PF literature to obtain quantitative bounds; see e.g. (Del Moral, 2004; Douc et al., 2014). Although this is not made explicit in the expressions above, there is an exponential dependence of the bounds on the state dimension d_x. This is unavoidable however and a well-known limitation of PF methods.

Finally note that DPF provides a biased estimate of the likelihood contrary to standard PF, so we cannot guarantee that the expectation of its logarithm, $\ell_\epsilon^{\text{ELBO}}(\theta, \phi) := \mathbb{E}_{\mathbf{U}}[\hat{\ell}_\epsilon(\theta; \phi, \mathbf{U})]$. is actually a valid ELBO. However in all our experiments, see e.g. Section 5.1, $|\ell_\epsilon^{\text{ELBO}}(\theta, \phi) - \ell^{\text{ELBO}}(\theta, \phi)|$ is significantly smaller than $\ell(\theta) - \ell^{\text{ELBO}}(\theta, \phi)$ so $\ell_\epsilon^{\text{ELBO}}(\theta, \phi) < \ell(\theta)$. Hence we keep the ELBO terminology.

5. Experiments

In Section 5.1, we assess the sensitivity of the DPF to the regularization parameter ϵ. All other DPF experiments presented here use the DET Resampling detailed in Algorithm 3 with $\epsilon = 0.5$, which ensures stability of the gradient calculations while adding little bias to the calculation of the ELBO compared to standard PF. Our method is implemented in both PyTorch and TensorFlow, the code to replicate the experiments as well as further experiments may be found at `https://github.com/JTT94/filterflow`.

5.1. Linear Gaussian State-Space Model

We consider here a simple two-dimensional linear Gaussian SSM for which the exact likelihood can be computed exactly using the Kalman filter

$$X_{t+1}|\{X_t = x\} \sim \mathcal{N}\left(\operatorname{diag}(\theta_1\ \theta_2)x, 0.5\mathbf{I}_2\right), \quad Y_t|\{X_t = x\} \sim \mathcal{N}(x, 0.1\mathbf{I}_2).$$

We simulate $T = 150$ observations using $\theta = (\theta_1, \theta_2) = (0.5, 0.5)$, for which we evaluate the ELBO at $\theta = (0.25, 0.25)$, $\theta = (0.5, 0.5)$, and $\theta = (0.75, 0.75)$. More precisely, using a standard PF with $N = 25$ particles, we compute the mean and standard deviation of $\frac{1}{T}(\hat{\ell}(\theta; \mathbf{U}) - \ell(\theta))$ over 100 realizations of $\mathbf{U}$. The mean is an estimate of the ELBO minus the true log-likelihood (rescaled by $1/T$). We then perform the same calculations for the DPF using the same number of particles and $\epsilon = 0.25, 0.5, 0.75$. As mentioned in Section 3.2 and Section 4.3, the DET resampling scheme is only satisfying Equation (2) for affine functions ψ so the DPF provides a biased estimate of the likelihood. Hence we cannot guarantee that the expectation of the corresponding log-likelihood estimate is a true ELBO. However, from Table 1, we observe that the difference between the ELBO estimates computed using PF and DPF is negligible for the three values of ϵ. The standard deviation of the log-likelihood estimates is also similar.

Table 1. Mean & std of $\frac{1}{T}(\hat{\ell}(\theta; \mathbf{U}) - \ell(\theta))$

θ_1, θ_2		0.25	0.5	0.75
PF	mean	-1.13	-0.93	-1.05
	std	0.20	0.18	0.17
DPF ($\epsilon = 0.25$)	mean	-1.14	-0.94	-1.07
	std	0.20	0.18	0.19
DPF ($\epsilon = 0.5$)	mean	-1.14	-0.94	-1.08
	std	0.20	0.18	0.18
DPF ($\epsilon = 0.75$)	mean	-1.14	-0.94	-1.08
	std	0.20	0.18	0.18

5.2. Learning the Proposal Distribution

We consider a similar example as in (Naesseth et al., 2018) where one learns the parameters ϕ of the proposal using the ELBO for the following linear Gaussian SSM:

$$X_{t+1}|\{X_t = x\} \sim \mathcal{N}\left(\mathbf{A}x, \mathbf{I}_{d_x}\right), \tag{15}$$

$$Y_t|\{X_t = x\} \sim \mathcal{N}(\mathbf{I}_{d_y,d_x}x, \mathbf{I}_{d_y}), \tag{16}$$

with $\mathbf{A} = (0.42^{|i-j|+1})_{1\leq i,j\leq d_x}$, $\mathbf{I}_{d_y,d_x}$ is a $d_y \times d_x$ matrix with 1 on the diagonal for the d_y first rows and zeros elsewhere. For $\phi \in \mathbb{R}^{d_x+d_y}$, we consider

$$q_\phi(x_t|x_{t-1}, y_t) = \mathcal{N}(x_t|\Delta_\phi^{-1}\left(\mathbf{A}x_{t-1} + \Gamma_\phi y_t\right), \Delta_\phi),$$

with $\Delta_\phi = \operatorname{diag}(\phi_1, \dots, \phi_{d_x})$ and a $d_x \times d_y$ matrix $\Gamma_\phi = \operatorname{diag}_{d_x,d_y}(\phi_1, \dots, \phi_{d_x})$ with ϕ_i on the diagonal for d_x first rows and zeros elsewhere. The locally optimal proposal $p(x_t|x_{t-1}, y_t) \propto g(y_t|x_t)f(x_t|x_{t-1})$ in (Doucet & Johansen, 2009) corresponds to $\phi = \mathbf{1}$, the vector with unit entries of dimension $d_\phi = d_x + d_y$.

For $d_x = 25, d_y = 1$, $M = 100$ realizations of $T = 100$ observations using (15)-(16), we learn ϕ on each realization using 100 steps of stochastic gradient ascent with learning rate 0.1 on the $\ell^{\text{ELBO}}(\phi)$ using regular PF with biased gradients as in (Maddison et al., 2017; Le et al., 2018; Naesseth et al., 2018) and $\ell^{\text{ELBO}}(\phi)$ with four independent filters using DPF. We use $N = 500$ for regular PF and $N = 25$ for DPF so as to match the computational complexity. While $p(x_t|x_{t-1}, y_t)$ is not guaranteed to maximize the ELBO, our experiments showed that it outperforms optimized proposals. We therefore report the RMSE of $\phi - \mathbf{1}$ and the average Effective Sample Size (ESS) (Doucet & Johansen, 2009) as proxy performance metrics. On both metrics, DPF outperforms regular PF. The RMSE over 100 experiments is 0.11 for DPF vs 0.22 for regular PF while the average ESS after convergence is around 60% for DPF vs 25% for regular PF. The average time per iteration was around 15 seconds for both DPF and PF.

5.3. Variational Recurrent Neural Network (VRNN)

A VRNN is an SSM introduced by (Chung et al., 2015) to improve upon LSTMs (Long Short Term Memory networks) with the addition of a stochastic component to the hidden state, this extends variational auto-encoders to a sequential setting. Indeed let latent state be $X_t = (R_t, Z_t)$ where R_t is an RNN state and Z_t a latent Gaussian variable, here Y_t is a vector of binary observations. The VRNN is detailed as follows. RNN_θ denotes the forward call of an LSTM cell which at time t emits the next RNN state R_{t+1} and output O_{t+1}. E_θ, h_θ, μ_θ, σ_θ are fully connected neural networks; detailed fully in the Supplementary Material. This model is trained on the polyphonic music benchmark datasets (Boulanger-Lewandowski et al., 2012), whereby Y_t represents which notes are active. The observation sequences are capped to length 150 for each dataset, with each observation of dimension 88. We chose latent states Z_t and R_t to be of dimension $d_z = 8$ and $d_r = 16$ respectively so $d_x = 24$. We use $q_\phi(x_t|x_{t-1}, y_t) = f_\theta(x_t|x_{t-1})$.

$$\begin{aligned}(R_{t+1}, O_{t+1}) &= \text{RNN}_\theta(R_t, Y_{1:t}, E_\theta(Z_t)),\\ Z_{t+1} &\sim \mathcal{N}(\mu_\theta(O_{t+1}), \sigma_\theta(O_{t+1})),\\ \hat{p}_{t+1} &= h_\theta(E_\theta(Z_{t+1}), O_{t+1}),\\ Y_t|X_t &\sim \text{Ber}(\hat{p}_t).\end{aligned}$$

The VRNN model is trained by maximizing $\ell_\epsilon^{\text{ELBO}}(\theta)$ using DPF. We compare this to the same model trained by

Table 2. ELBO $\pm$ Standard Deviation evaluated using Test Data.

	MuseData	JSB	Nottingham
DPF	$\mathbf{-7.59}_{\pm 0.01}$	$\mathbf{-7.67}_{\pm 0.08}$	$\mathbf{-3.79}_{\pm 0.02}$
PF	$-7.60_{\pm 0.06}$	$-7.92_{\pm 0.13}$	$-3.81_{\pm 0.02}$
SPF	$-7.73_{\pm 0.14}$	$-8.17_{\pm 0.07}$	$-3.91_{\pm 0.05}$

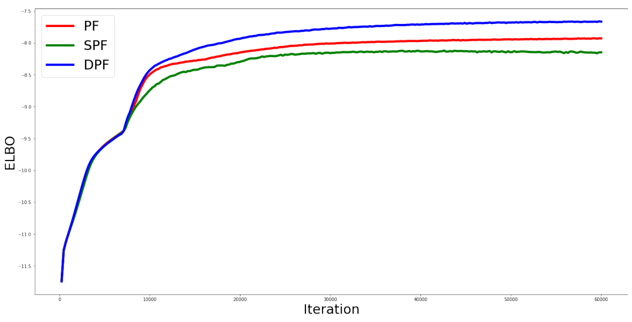


Figure 2. ELBO during training, evaluated on Test Data for JSB.

maximizing $\ell^{\text{ELBO}}(\theta)$ computed with regular PF (Maddison et al., 2017) and also trained with 'soft-resampling' (SPF) introduced by (Karkus et al., 2018) and described in Section 1.3, SPF is used here with parameter $\alpha = 0.1$. Unlike regular resampling, SPF partially preserves a gradient through the resampling step, however SPF still involves a non-differentiable operation, again resulting in a biased gradient. SPF also produces higher variance estimates as the resampled approximation is not uniformly weighted, essentially interpolating between PF and IWAE. Each of the methods are performed with $N = 32$ particles. Although DET is computationally more expensive than the other resampling schemes, the computational times of DPF, PF, and SPF are very similar due to most of the complexity coming from neural network operations. The learned models are then evaluated on test data using multinomial resampling for comparable ELBO results. Due to the fact that our observation model is $\text{Ber}(\hat{p}_t)$, this recovers the negative log-predictive cross-entropy.

Figure 2 and Table 2 illustrate the benefit of using DPF over regular PF and SPF for the JSB dataset. Although DPF remains competitive compared to other heuristic approaches, the difference is relatively minor for the other datasets. We speculate that the performance of the heuristic methods is likely due to low predictive uncertainty for the next observation given the previous one.

5.4. Robot Localization

Consider the setting of a robot/agent in a maze (Jonschkowski et al., 2018; Karkus et al., 2018). Given the agent's initial state, S_1, and inputs a_t, one would like to infer the location of the agent at any specific time given observations O_t. Let the latent state be denoted $S_t = (X_t^{(1)}, X_t^{(2)}, \gamma_t)$ where $(X_t^{(1)}, X_t^{(2)})$ are location coordinates and γ_t the robot's orientation. In our setting observations O_t are images, which are encoded to extract useful features using a neural network E_θ, where $Y_t = E_\theta(O_t)$. This problem requires learning the relationship between the robot's location, orientation and the observations. Given actions $a_t = (v_t^{(1)}, v_t^{(2)}, \omega_t)$, we have

$$S_{t+1} = F_\theta(S_t, a_t) + \nu_t, \quad \nu_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \Sigma_F),$$
$$Y_t = G_\theta(S_t) + \epsilon_t, \quad \epsilon_t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \sigma_G^2 \mathbb{I}_{e_d}),$$

where $\Sigma_F = \text{diag}(\sigma_x^2, \sigma_x^2, \sigma_\theta^2)$ and the relationship between state S_t and image encoding Y_t may be parameterized by another neural network G_θ. We consider here a simple linear model of the dynamics

$$F(S_t, a_t) = \begin{bmatrix} X_t^{(1)} + v_t^{(1)} \cos(\gamma_t) + v_t^{(2)} \sin(\gamma_t) \\ X_t^{(2)} + v_t^{(1)} \sin(\gamma_t) - v_t^{(2)} \cos(\gamma_t) \\ \gamma_t + \omega_t \end{bmatrix}.$$

D_θ denotes a decoder neural network, mapping the encoding back to the original image. E_θ, G_θ and D_θ are trained using a loss function consisting of the PF-estimated log-likelihood $\hat{\mathcal{L}}_{\text{PF}}$; PF-based mean squared error (MSE), $\hat{\mathcal{L}}_{\text{MSE}}$; and auto-encoder loss, $\hat{\mathcal{L}}_{\text{AE}}$, given per-batch as in (Wen et al., 2020):

$$\hat{\mathcal{L}}_{\text{MSE}} := \frac{1}{T}\sum_{t=1}^{T} ||X_t^* - \sum_{i=1}^{N} w_t^i X_t^i||^2, \quad \hat{\mathcal{L}}_{\text{PF}} := -\frac{1}{T}\hat{\ell}(\theta),$$
$$\hat{\mathcal{L}}_{\text{AE}} := \sum_{t=1}^{T} ||D_\theta(E_\theta(O_t)) - O_t||^2,$$

where $X_t^\star$ are the true states available from training data and $\sum_{i=1}^{N} w_t^i X_t^i$ are the PF estimates of $\mathbb{E}[X_t | y_{1:t}]$. The auto-encoder / reconstruction loss $\hat{\mathcal{L}}_{\text{AE}}$ ensures the encoder is informative and prevents the case whereby networks G_θ, E_θ map to a constant. The PF-based loss terms $\hat{\mathcal{L}}_{\text{MSE}}$ and $\hat{\mathcal{L}}_{\text{PF}}$ are not differentiable w.r.t. θ under traditional resampling schemes.

We use the setup from (Jonschkowski et al., 2018) with data from DeepMind Lab (Beattie et al., 2016). This consists of 3 maze layouts of varying sizes. We have access to 'true' trajectories of length $1,000$ steps for each maze. Each step has an associated state, action and observation image, as described above. The visual observation O_t consists of 32×32 RGB pixel images, compressed to 24×24, as shown in Figure 3. Random, noisy subsets of fixed length are sampled at each training iteration. To illustrate the benefits of our proposed method, we select the random subsets to be of length 50 as opposed to length 20 as chosen in (Jonschkowski et al., 2018). Training details in terms of learning rates, number of training steps and neural network architectures for E_θ, G_θ and D_θ are given in the Appendices.

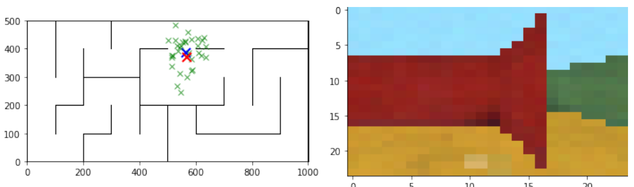
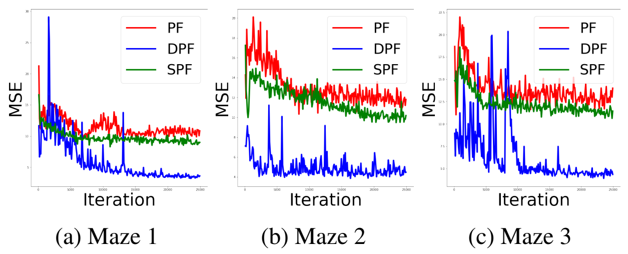


Figure 3. Left: Particles $(X_t^{(1),i}, X_t^{(2),i})$ (green), PF estimate of $\mathbb{E}[X_t|y_{1:t}]$ (blue), true state X_t^* (red). Right: Observation, O_t.

We compare our method, DPF, to regular PF used in (Maddison et al., 2017) and Soft PF (SPF) used in (Karkus et al., 2018; Ma et al., 2020a;b), whereby the soft resampling is used with $\alpha = 0.1$. As most of the computational complexity arises from neural network operations, DPF is of similar overall computational cost to SPF and PF. As shown in Table 3 and Figure 4, DPF significantly outperforms previously considered PF methods in this experiment. The observation model becomes increasingly important for longer sequences due to resampling and weighting operations. Indeed, as shown in Figure 5, the error is small for both models at the start of the sequence, however the error at later stages in the sequence is visibly smaller for the model trained using DPF.

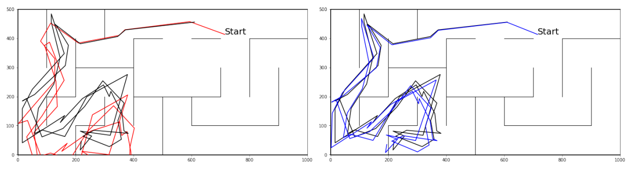
Table 3. MSE and $\pm$ Standard Deviation evaluated on Test Data: Lower is better

	MAZE 1	MAZE 2	MAZE 3
DPF	$\mathbf{3.55}_{\pm\mathbf{0.20}}$	$\mathbf{4.65}_{\pm 0.50}$	$\mathbf{4.44}_{\pm\mathbf{0.26}}$
PF	$10.71_{\pm 0.45}$	$11.86_{\pm 0.57}$	$12.88_{\pm 0.65}$
SPF	$9.14_{\pm 0.39}$	$10.12_{\pm\mathbf{0.40}}$	$11.42_{\pm 0.37}$



(a) Maze 1 (b) Maze 2 (c) Maze 3

Figure 4. MSE of PF (red), SPF (green) and DPF (blue) estimates, evaluated on test data during training.



(a) Standard PF (b) Differentiable PF

Figure 5. Illustrative Example: PF estimate of path compared to true path (black) on a single 50-step trajectory from test data.

6. Discussion

This paper introduces the first principled, fully differentiable PF (DPF) which permits parameter inference in state-space models using end-to-end gradient based optimization. This property allows the use of PF routines in general differentiable programming pipelines, in particular as a differentiable sampling method for inference in probabilistic programming languages (Dillon et al., 2017; Ge et al., 2018; van de Meent et al., 2018).

For a given number of particles N, existing PF methods ignoring resampling gradient terms have computational complexity $O(N)$. Training with these resampling schemes however is unreliable and performance cannot be improved by increasing N as gradient estimates are inconsistent and the limiting bias can be significant. DPF has complexity $O(N^2)$ during training. However, this cost is dwarfed when training large neural networks. Additionally, once the model is trained, standard PF may be ran at complexity $O(N)$. The benefits of DPF are confirmed by our experimental results where it was shown to outperform existing techniques, even when an equivalent computational budget was used. Moreover, recent techniques have been proposed to speed up the Sinkhorn algorithm (Altschuler et al., 2019; Scetbon & Cuturi, 2020) at the core of DPF and could potentially be used here to reduce its complexity.

Regularization parameter ϵ was not fine-tuned in our experiments. In future work, it would be interesting to obtain sharper quantitative bounds on DPF to propose principled guidelines on choosing ϵ, further improving its performance. Finally, we have focused on the use of the differentiable ensemble transform to obtain a differentiable resampling scheme. However, alternative OT approaches could also be proposed such as a differentiable version of the second order ET presented in (Acevedo et al., 2017), or techniques based on point cloud optimization (Cuturi & Doucet, 2014; Peyré & Cuturi, 2019) relying on the Sinkhorn divergence (Genevay et al., 2018) or the sliced-Wasserstein metric. Alternative non-entropic regularizations, such as the recently proposed Gaussian smoothed OT (Goldfeld & Greenewald, 2020), could also lead to DPFs of interest.

Acknowledgments

The work of Adrien Corenflos was supported by the Academy of Finland (projects 321900 and 321891). Arnaud Doucet is supported by the EPSRC CoSInES (COmputational Statistical INference for Engineering and Security) grant EP/R034710/1, James Thornton by the OxWaSP CDT through grant EP/L016710/1. Computing resources provided through the Google Cloud Platform Research Credits Programme.

References

Acevedo, W., de Wiljes, J., and Reich, S. Second-order accurate ensemble transform particle filters. *SIAM Journal on Scientific Computing*, 39(5):A1834–A1850, 2017.

Altschuler, J., Niles-Weed, J., and Rigollet, P. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in Neural Information Processing Systems*, pp. 1964–1974, 2017.

Altschuler, J., Bach, F., Rudi, A., and Niles-Weed, J. Massively scalable Sinkhorn distances via the Nyström method. In *Advances in Neural Information Processing Systems*, pp. 4429–4439, 2019.

Archer, E., Park, I. M., Buesing, L., Cunningham, J., and Paninski, L. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. DeepMind Lab, 2016.

Bertsimas, D. and Tsitsiklis, J. N. *Introduction to Linear Optimization*. Athena Scientific Belmont, MA, 1997.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *International Conference on Machine Learning*, pp. 1881–1888, 2012.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

Chopin, N. and Papaspiliopoulos, O. *An Introduction to Sequential Monte Carlo*. Springer, 2020.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pp. 2980–2988, 2015.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pp. 2292–2300, 2013.

Cuturi, M. and Doucet, A. Fast computation of Wasserstein barycenters. In *International Conference on Machine Learning*, pp. 685–693, 2014.

DeJong, D. N., Liesenfeld, R., Moura, G. V., Richard, J.-F., and Dharmarajan, H. Efficient likelihood evaluation of state-space representations. *Review of Economic Studies*, 80(2):538–567, 2013.

Del Moral, P. *Feynman-Kac Formulae*. Springer, 2004.

Del Moral, P. and Guionnet, A. On the stability of interacting processes with applications to filtering and genetic algorithms. In *Annales de l'Institut Henri Poincaré (B) Probability and Statistics*, volume 37, pp. 155–194, 2001.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. A. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

Douc, R., Moulines, E., and Stoffer, D. *Nonlinear Time Series: Theory, Methods and Applications with R Examples*. CRC press, 2014.

Doucet, A. and Johansen, A. M. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

Doucet, A. and Lee, A. Sequential Monte Carlo methods. *Handbook of Graphical Models*, pp. 165–189, 2018.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-I., Trouvé, A., and Peyré, G. Interpolating between optimal transport and MMD using Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Flamary, R., Cuturi, M., Courty, N., and Rakotomamonjy, A. Wasserstein discriminant analysis. *Machine Learning*, 107(12):1923–1945, 2018.

Fournier, N. and Guillin, A. On the rate of convergence in Wasserstein distance of the empirical measure. *Probability Theory and Related Fields*, 162(3):707–738, 2015.

Ge, H., Xu, K. X., and Ghahramani, Z. Turing: A language for flexible probabilistic inference. In *International Conference on Artificial Intelligence and Statistics*, pp. 1682–1690, 2018.

Genevay, A., Peyré, G., and Cuturi, M. Learning generative models with Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pp. 1608–1617, 2018.

Goldfeld, Z. and Greenewald, K. Gaussian-smooth optimal transport: Metric structure and statistical efficiency. *arXiv preprint arXiv 2001.09206*, 2020.

Hirt, M. and Dellaportas, P. Scalable Bayesian learning for state space models using variational inference with SMC samplers. In *International Conference on Artificial Intelligence and Statistics*, pp. 76–86, 2019.

Jonschkowski, R., Rastogi, D., and Brock, O. Differentiable particle filters: End-to-end learning with algorithmic priors. In *Proceedings of Robotics: Science and Systems*, 2018.

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., and Chopin, N. On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351, 2015.

Karkus, P., Hsu, D., and Lee, W. S. Particle filter networks with application to visual localization. In *Conference on Robot Learning*, 2018.

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

Kitagawa, G. and Gersch, W. *Smoothness Priors Analysis of Time Series*, volume 116. Springer Science & Business Media, 1996.

Klaas, M., De Freitas, N., and Doucet, A. Toward practical N^2 Monte Carlo: the marginal particle filter. *Uncertainty in Artificial Intelligence*, 2005.

Kloss, A., Martius, G., and Bohg, J. How to train your differentiable filter. *arXiv preprint arXiv:2012.14313*, 2020.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. In *AAAI Conference on Artificial Intelligence*, pp. 2101–2109, 2017.

Le, T. A., Igl, M., Rainforth, T., Jin, T., and Wood, F. Auto-encoding sequential Monte Carlo. In *International Conference on Learning Representations*, 2018.

Lee, A. Towards smooth particle filters for likelihood estimation with multivariate latent variables. Master's thesis, University of British Columbia, 2008.

Li, W. and Nochetto, R. H. Quantitative stability and error estimates for optimal transport plans. *IMA Journal of Numerical Analysis*, 2021.

Lindsten, F. and Schön, T. B. Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends® in Machine Learning*, 6(1):1–143, 2013.

Ma, X., Karkus, P., Hsu, D., and Lee, W. S. Particle filter recurrent neural networks. In *AAAI Conference on Artificial Intelligence*, 2020a.

Ma, X., Karkus, P., Ye, N., Hsu, D., and Lee, W. S. Discriminative particle filter reinforcement learning for complex partial observations. In *International Conference on Learning Representations*, 2020b.

Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. W. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, 2017.

Malik, S. and Pitt, M. K. Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, 165(2):190–209, 2011.

Murray, L. M., Jones, E. M., and Parslow, J. On disturbance state-space models and the particle marginal Metropolis–Hastings sampler. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):494–521, 2013.

Myers, A., Thiery, A. H., Wang, K., and Bui-Thanh, T. Sequential ensemble transform for Bayesian inverse problems. *Journal of Computational Physics*, 427:110055, 2021.

Naesseth, C. A., Linderman, S. W., Ranganath, R., and Blei, D. M. Variational sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Peyré, G. and Cuturi, M. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6): 355–607, 2019.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pp. 7785–7794, 2018.

Reich, S. A nonparametric ensemble transform method for Bayesian inference. *SIAM Journal on Scientific Computing*, 35(4):A2013–A2024, 2013.

Scetbon, M. and Cuturi, M. Linear time Sinkhorn divergences using positive features. In *Advances in Neural Information Processing Systems*, 2020.

Seguy, V., Damodaran, B. B., Flamary, R., Courty, N., Rolet, A., and Blondel, M. Large-scale optimal transport and mapping estimation. In *International Conference on Learning Representations*, 2018.

Thrun, S., Burgard, W., and Fox, D. *Probabilistic Robotics*. MIT Press, 2005.

van de Meent, J.-W., Paige, B., Hongseok, Y., and Wood, F. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

Villani, C. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media, 2008.

Weed, J. An explicit analysis of the entropic penalty in linear programming. In *Proceedings of the 31st Conference On Learning Theory*, 2018.

Wen, H., Chen, X., Papagiannis, G., Hu, C., and Li, Y. End-to-end semi-supervised learning for differentiable particle filters. *arXiv preprint arXiv:2011.05748*, 2020.

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

Zhu, M., Murphy, K., and Jonschkowski, R. Towards differentiable resampling. *arXiv preprint arXiv:2004.11938*, 2020.

A. Proof of Proposition 4.1

A particle filter with multinomial resampling is defined by the following joint distribution

$$\overline{q}_{\theta,\phi}(x_{1:T}^{1:N}, a_{1:T-1}^{1:N}) = \prod_{i=1}^{N} q_\phi\left(x_1^i\right) \prod_{t=2}^{T}\prod_{i=1}^{N} w_{t-1}^{a_{t-1}^i} q_\phi\left(x_t^i | x_{t-1}^{a_{t-1}^i}\right)$$

where $a_{t-1}^i \in \{1, ..., N\}$ is the ancestral index of particle x_t^i and

$$\omega_{\theta,\phi}(x_1, y_1) = \frac{p_\theta(x_1, y_1)}{q_\phi(x_1)}, \quad \omega_{\theta,\phi}(x_{t-1}, x_t, y_t) = \frac{p_\theta(x_t, y_t | x_{t-1})}{q_\phi\left(x_t | x_{t-1}\right)}.$$

Finally, we have $w_t^i \propto \omega_{\theta,\phi}(x_{t-1}^{a_{t-1}^i}, x_t^i, y_t)$, $\sum_{i=1}^N w_t^i = 1$. We do not emphasize notationally that the weights $w_{t-1}^{a_{t-1}^i}$ are θ, ϕ and observations dependent.

The ELBO is given by

$$\ell^{\text{ELBO}}(\theta,\phi) = \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\log \widehat{p}_\theta(y_{1:T})\right] = \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\log\left(\frac{1}{N}\sum_{i=1}^{N}\omega_{\theta,\phi}(X_1^i, y_1)\right) + \sum_{t=2}^{T}\log\left(\frac{1}{N}\sum_{i=1}^{N}\omega_{\theta,\phi}(X_{t-1}^{A_{t-1}^i}, X_t^i, y_t)\right)\right].$$

We now compute $\nabla_\theta \ell^{\text{ELBO}}(\theta,\phi)$. We assume from now on that the regularity conditions allowing us to swap the expectation and differentiation operators are satisfied as in (Maddison et al., 2017; Le et al., 2018; Naesseth et al., 2018). We can split the gradient using the product rule and apply the log-derivative trick:

$$\begin{aligned}\nabla_\theta \ell^{\text{ELBO}}(\theta,\phi) =& \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\nabla_\theta \log \widehat{p}_\theta(y_{1:T})\right] + \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\log \widehat{p}_\theta(y_{1:T}) \nabla_\theta \log \overline{q}_{\theta,\phi}(X_{1:T}^{1:N}, A_{1:T-1}^{1:N})\right] \\ =& \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\nabla_\theta \log\left(\frac{1}{N}\sum_{i=1}^{N}\omega_{\theta,\phi}(X_1^i, y_1)\right) + \sum_{t=2}^{T}\nabla_\theta \log\left(\frac{1}{N}\sum_{i=1}^{N}\omega_{\theta,\phi}(X_{t-1}^{A_{t-1}^i}, X_t^i, y_t)\right)\right] \quad (17) \\ &+ \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\log \widehat{p}_\theta(y_{1:T})\left\{\sum_{t=2}^{T}\sum_{i=1}^{N}\nabla_\theta \log w_{t-1}^{A_{t-1}^i}\right\}\right] \quad (18)\end{aligned}$$

For the first part of the ELBO gradient (17), we have

$$\nabla_\theta \log\left(\frac{1}{N}\sum_{i=1}^{N} w_{\theta,\phi}(X_1^i), y_1\right) = \sum_{i=1}^{N} w_1^i \nabla_\theta \log w_{\theta,\phi}(X_1^i, y_1) = \sum_{i=1}^{N} \omega_1^i \nabla_\theta \log p_\theta(X_1^i, y_1)$$

and

$$\nabla_\theta \log\left(\frac{1}{N}\sum_{i=1}^{N}\omega_{\theta,\phi}(X_{t-1}^{A_{t-1}^i}, X_t^i, y_t)\right) = \sum_{i=1}^{N} w_t^i \nabla_\theta \log \omega_{\theta,\phi}(X_{t-1}^{A_{t-1}^i}, X_t^i, y_t) = \sum_{i=1}^{N} w_t^i \nabla_\theta \log p_\theta(X_t^i, y_t | X_{t-1}^{A_{t-1}^i}).$$

This gives

$$\begin{aligned}\nabla_\theta \ell^{\text{ELBO}}(\theta,\phi) =& \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\sum_{i=1}^{N} w_1^i \nabla_\theta \log p_\theta(X_1^i, y_1) + \sum_{t=2}^{T}\sum_{i=1}^{N} w_t^i \nabla_\theta \log p_\theta(X_t^i, y_t | X_{t-1}^{A_{t-1}^i})\right] \quad (19) \\ &+ \mathbb{E}_{\overline{q}_{\theta,\phi}}\left[\log \widehat{p}_\theta(y_{1:T})\left\{\sum_{t=2}^{T}\sum_{i=1}^{N}\nabla_\theta \log w_{t-1}^{A_{t-1}^i}\right\}\right]. \quad (20)\end{aligned}$$

When we ignore the gradient terms due to resampling corresponding to (20) as proposed in (Naesseth et al., 2018; Le et al., 2018; Maddison et al., 2017; Hirt & Dellaportas, 2019), we only use an unbiased estimate of the first term (19), i.e.

$$\hat{\nabla}_\theta \ell^{\text{ELBO}}(\theta,\phi) := \sum_{i=1}^{N} w_1^i \nabla_\theta \log p_\theta(X_1^i, y_1) + \sum_{t=2}^{T}\sum_{i=1}^{N} w_t^i \nabla_\theta \log p_\theta(X_t^i, y_t | X_{t-1}^{A_{t-1}^i}), \quad \text{where } (X_{1:T}^{1:N}, A_{1:T-1}^{1:N}) \sim \overline{q}_{\theta,\phi}(\cdot). \quad (21)$$

Now we assume that the mild assumptions ensuring almost sure convergence of the PF estimates are satisfied (see e.g. (Del Moral, 2004)). Under these assumptions, the estimator (21) converges almost surely as $N \to \infty$ towards

$$\int \nabla_\theta \log p_\theta(x_1, y_1) p_\theta(x_1|y_1) \mathrm{d}x_1 + \sum_{t=2}^{T} \int \nabla_\theta \log p_\theta(x_t, y_t|x_{t-1}) p_\theta(x_{t-1:t}|y_{1:t-1}) \mathrm{d}x_{t-1:t}. \tag{22}$$

Under an additional uniform integrability condition on $\hat{\nabla}_\theta \ell^{\text{ELBO}}(\theta, \phi)$, we thus have that $\mathbb{E}_{\overline{q}_{\theta,\phi}}[\hat{\nabla}_\theta \ell^{\text{ELBO}}(\theta, \phi)]$ converges towards (22). We recall that the true score is given by Fisher's identity and satisfies

$$\int \nabla_\theta \log p_\theta(x_1, y_1) p_\theta(x_1|y_{1:T}) \mathrm{d}x_1 + \sum_{t=2}^{T} \int \nabla_\theta \log p_\theta(x_t, y_t|x_{t-1}) p_\theta(x_{t-1:t}|y_{1:T}) \mathrm{d}x_{t-1:t}.$$

This concludes the proof of Proposition 4.1.

B. Notation and Assumptions

B.1. Filtering Notation

Recall $\mathcal{X} = \mathbb{R}^{d_x}$, denote the Borel sets of $\mathcal{X}$ by $\mathcal{B}(\mathcal{X})$ and $\mathcal{P}(\mathcal{X})$ the set of Borel probability measures on $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$. In an abuse of notation, we shall use the same notation for a probability measure and its density w.r.t. Lebesgue measure; i.e. $\nu(\mathrm{d}x) = \nu(x)\mathrm{d}x$. We also use the standard notation $\nu(\psi) = \int \psi(x)\nu(x)\mathrm{d}x$ for any test function ψ. In the interest of notational clarity, we will remove subscript θ, ϕ where unnecessary in further workings.

We denote $\{\alpha^{(t)}\}_{t\geq 0}$ the predictive distributions where $\alpha^{(t)}(x_t) = p(x_t|y_{1:t-1})$ for $t > 1$ and $\alpha^{(1)}(x_1) = \mu(x_1)$ while $\{\beta^{(t)}\}_{t\geq 1}$ denotes the filtering distributions; i.e. $\beta^{(t)}(x_t) = p(x_t|y_{1:t})$ for $t \geq 1$.

Using this notation, we have

$$\alpha^{(t)}(\psi) = \int \psi(x_t) f(x_t|x_{t-1}) \beta^{(t-1)}(x_{t-1}) \mathrm{d}x_{t-1} \mathrm{d}x_t := \beta^{(t-1)} f(\psi), \tag{23}$$

$$\beta^{(t)}(\psi) = \frac{\alpha^{(t)}(g(y_t|\cdot)\psi)}{\alpha^{(t)}(g(y_t|\cdot))} = \frac{\beta^{(t-1)}(f(g(y_t|\cdot)\psi))}{\beta^{(t-1)}(f(g(y_t|\cdot)))}. \tag{24}$$

More generally, for a proposal distribution $q(x_t|x_{t-1}, y_t) \neq f(x_t|x_{t-1})$ with parameter $\phi \neq \theta$, the following recursion holds

$$\beta^{(t)}(\psi) = \frac{\beta^{(t-1)}(q(\omega_t\,\psi))}{\beta^{(t-1)}(q(\omega_t))} \tag{25}$$

$$\omega_t(x_{t-1}, x_t) := \omega(x_{t-1}, x_t, y_t) = \frac{g(y_t|x_t) f(x_t|x_{t-1})}{q(x_t|x_{t-1}, y_t)}. \tag{26}$$

To simplify the presentation, we will present the analysis in the scenario where $\phi = \theta$ and $q(x_t|x_{t-1}, y_t) = f(x_t|x_{t-1})$ so we will analyze (23) for which $\omega_t(x_{t-1}, x_t) = g(y_t|x_t)$. In this case, the particle approximations of μ is denoted μ_N and for $t > 1$, $\alpha^{(t)}$ and $\beta^{(t)}$ are given by the random measures

$$\alpha_N^{(t)}(\psi) = \frac{1}{N} \sum_{i=1}^{N} \psi(X_t^i), \quad \beta_N^{(t)}(\psi) = \sum_{i=1}^{N} w_t^i \psi(X_t^i), \quad \tilde{\beta}_N^{(t)}(\psi) = \frac{1}{N} \sum_{i=1}^{N} \psi(\tilde{X}_t^i), \tag{27}$$

where $w_t^i \propto g(y_t|X_t^i)$ with $\sum_{i=1}^{N} w_t^i = 1$ and particles are drawn from $X_t^i \sim f(\cdot|\tilde{X}_{t-1}^i)$.

Here $\beta_N^{(t)}$ denotes the weighted particle approximation of $\beta^{(t)}$ while $\tilde{\beta}_N^{(t)}$ is the uniformly weighted approximation obtained after the DET transformation described in Section 3.2.

B.2. Optimal Transport Notation

Recall from Section 2.1, $\mathcal{P}_t^{\text{OT}}$ denotes a transport between $\alpha^{(t)}$ and $\beta^{(t)}$ with accompanying map $\mathbf{T}^{(t)}$. $\mathcal{P}_t^{\text{OT},N}$ denotes an optimal transport between particle approximations $\alpha_N^{(t)}$ and $\beta_N^{(t)}$ with corresponding transport matrix, $\mathbf{P}^{\text{OT}}$ with i, j entry $p_{i,j}^{\text{OT}}$. To simplify notation, we remove script t when not needed.

Similarly from Section 3.1, $\mathcal{P}_\epsilon^{\text{OT,N}}$ denotes the regularized transport between $\alpha_N^{(t)}$ and $\beta_N^{(t)}$ with accompanying matrix $\mathbf{P}_\epsilon^{\text{OT}}$ with i, j entry $p_{\epsilon,i,j}^{\text{OT}}$. Recall $\tilde{\beta}_N^{(t)} = \frac{1}{N}\sum_{i=1}^N \delta_{\tilde{X}^i}$ is the uniformly weighted particle approximation for $\beta^{(t)}$ under the DET, i.e. $\tilde{X}^i = \mathbf{T}_{N,\epsilon}^{(t)}(X^i) = \int y\mathcal{P}_\epsilon^{\text{OT},N}(\mathrm{d}y|x^i)$. Note that $\tilde{X}_{N,\epsilon}^i$ will be used where necessary to avoid ambiguity when comparing to other resampling schemes.

Recall also for $p > 0$:

$$\mathcal{W}_p^p(\alpha, \beta) = \min_{\mathcal{P}\in\mathcal{U}(\alpha,\beta)} \mathbb{E}_{(U,V)\sim\mathcal{P}}\left[||U - V||^p\right] \tag{28}$$

where $\mathcal{U}(\alpha, \beta)$ is the collection of couplings with marginals α and β.

B.3. Assumptions

Our results will rely on the following four assumptions.

Assumption B.1. $\mathcal{X} \subset \mathbb{R}^d$ is a compact subset with diameter

$$\mathfrak{d} := \sup_{x,y\in\mathcal{X}} |x - y|.$$

Assumption B.2. There exists $\kappa \in (0, 1)$ such that for any two probability measures π, ρ on $\mathcal{X}$

$$\mathcal{W}_k(\pi f, \rho f) \leq \kappa \mathcal{W}_k(\pi, \rho), \qquad k = 1, 2.$$

Assumption B.3. The weight function $\omega^{(t)} : \mathcal{X} \to [\Delta, \Delta^{-1}]$ is 1-Lipschitz for all t.

Assumption B.4. There exists a $\lambda > 0$, such that for all $t \geq 0$ the unique optimal transport plan between $\alpha^{(t)}$ and $\beta^{(t)}$ is given by a deterministic, λ-Lipschitz map $\mathbf{T}^{(t)}$.

C. Auxiliary Results and Proof of Proposition 4.2

We start by establishing a couple of key auxiliary results which will be then used subsequently to establish Proposition 4.2.

C.1. Auxiliary Results

As per section 2.1, let $\mathcal{S}(\alpha_N, \beta_N)$ denote the collection of coupling matrices between $\alpha_N = \sum_{i=1}^N a_i\delta_{Y^i}$ with $a_i > 0$ and $\beta_N = \sum_{i=1}^N b_i\delta_{X^i}$. We also denote entropy by H where $H(\mathbf{P}) = \sum_{i,j} p_{i,j}\log(1/p_{i,j})$ for $\mathbf{P} = (p_{i,j})_{i,j} \in \mathcal{S}(\alpha_N, \beta_N)$.

Lemma C.1. *The entropic radius, R_H, of simplex $\mathcal{U}(\alpha_N, \beta_N)$ may be bounded above as follows*

$$R_H := \max_{\mathbf{P}_1,\mathbf{P}_2\in\mathcal{S}(\alpha_N,\beta_N)} H(\mathbf{P}_1) - H(\mathbf{P}_2) \leq 2\log(N)$$

Proof. Notice that $-H(\mathbf{P})$ is convex, so that $H(\mathbf{P})$ is concave.

$$\begin{aligned}\sum_{i,j} p_{i,j}\log\left(\frac{1}{p_{i,j}}\right) &= N^2\sum_{i,j}\frac{1}{N^2}p_{i,j}\log\left(\frac{1}{p_{i,j}}\right)\\ &\leq N^2 H\left(\frac{1}{N^2}\sum_{i,j}p_{i,j}\right) = N^2H(1/N^2) = N^2\frac{1}{N^2}\log(N^2) = 2\log(N).\end{aligned}$$

In addition since $p_{i,j} \leq 1$ for all i, j, we have that $H(\mathbf{P}) \geq 0$ and therefore we can bound

$$R_H = \max_{P_1, P_2 \in \mathcal{P}} H(\mathbf{P}_1) - H(\mathbf{P}_2) \leq \max_{\mathbf{P}_1 \in \mathcal{S}(\alpha_N, \beta_N)} H(\mathbf{P}_1) \leq 2\log(N).$$

□

Lemma C.2. *Let $\mathcal{X} \subset \mathbb{R}^d$ be compact with diameter $\mathfrak{d} > 0$. Suppose we are given two probability measures α, β on $\mathcal{X}$ with a unique deterministic, λ-Lipschitz optimal transport map $\mathbf{T}$ while $\alpha_N = \sum_{i=1}^N a_i \delta_{Y^i}$ with $a_i > 0$ and $\beta_N = \sum_{i=1}^N b_i \delta_{X^i}$. We write $\mathcal{P}^{\mathrm{OT},N}$, resp. $\mathcal{P}_\epsilon^{\mathrm{OT},N}$, for an optimal coupling between α_N and β_N, resp. the ϵ-regularized optimal transport plan, between α_N and β_N. Then*

$$\left[\int ||y - \mathbf{T}(x)||^2 \mathcal{P}_\epsilon^{\mathrm{OT},N}(\mathrm{d}x, \mathrm{d}y)\right]^{\frac{1}{2}} \leq 2\lambda^{1/2}\mathcal{E}^{1/2}\left[\mathfrak{d}^{1/2} + \mathcal{E}\right]^{1/2} + \max\{\lambda, 1\}\left[\mathcal{W}_2(\alpha_N, \alpha) + \mathcal{W}_2(\beta_N, \beta)\right],$$

where

$$\mathcal{E} := \mathcal{E}(N, \epsilon, \alpha, \beta) := \mathcal{W}_2(\alpha_N, \alpha) + \mathcal{W}_2(\beta_N, \beta) + \sqrt{2\epsilon \log(N)}.$$

Proof. From Corollary 3.8 from (Li & Nochetto, 2021)

$$\left[\int \|\mathbf{T}(x) - y\|^2 \mathcal{P}_\epsilon^{\mathrm{OT},N}(\mathrm{d}x, \mathrm{d}y)\right]^{1/2} \leq 2\lambda^{1/2}\sqrt{\tilde{e}_{N,\epsilon}}\left[\mathcal{W}_2(\alpha, \beta) + \tilde{e}_{N,\epsilon}\right]^{1/2} + \lambda\mathcal{W}_2(\alpha_N, \alpha) + \mathcal{W}_2(\beta_N, \beta),$$

where λ is the Lipschitz constant of the optimal transport map $\mathbf{T}$ sending α to β, and

$$\tilde{e}_{N,\epsilon} := \mathcal{W}_2(\alpha_N, \alpha) + \mathcal{W}_2(\beta_N, \beta) + \left[\int \|x - y\|^2 \mathcal{P}_\epsilon^{\mathrm{OT},N}(\mathrm{d}x, \mathrm{d}y)\right]^{1/2} - \mathcal{W}_2(\alpha_N, \beta_N). \tag{29}$$

From Proposition 4 of (Weed, 2018),

$$\sum_{i,j=1,\ldots,N} p_{\epsilon,i,j}^{\mathrm{OT}}|Y_i - X_j|^2 - \mathcal{W}_2^2(\alpha_N, \beta_N) \leq \epsilon R_H,$$

where R_H is the entropic radius as defined in Lemma C.1.

By Lemma C.1 we therefore have that

$$\int \|x - y\|^2 \mathcal{P}_\epsilon^{\mathrm{OT},N}(\mathrm{d}x, \mathrm{d}y) - \mathcal{W}_2^2(\alpha_N, \beta_N) \leq 2\epsilon \log(N).$$

Since $x \mapsto \sqrt{x}$ is sub-additive, for $r, s > 0$ we have that $\sqrt{r} - \sqrt{s} \leq \sqrt{r - s}$, whence

$$\left[\int \|x - y\|^2 \mathcal{P}_\epsilon^{\mathrm{OT},N}(\mathrm{d}x, \mathrm{d}y)\right]^{1/2} - \mathcal{W}_2(\alpha_N, \beta_N) \leq \sqrt{2\epsilon \log N}.$$

We thus have

$$\tilde{e}_{N,\epsilon} \leq \mathcal{W}_2(\alpha_N, \alpha) + \mathcal{W}_2(\beta_N, \beta) + \sqrt{2\epsilon \log(N)}.$$

In addition, by Assumption B.1 we have that $\mathcal{W}_2(\alpha, \beta) \leq \mathfrak{d}^{1/2}$ and the result follows. □

C.2. Proof of Proposition 4.2

Proof of Proposition 4.2. By definition, we have $\tilde{\beta}_N(\mathrm{d}\tilde{x}) = \int \alpha_N(\mathrm{d}x)\delta_{\mathbf{T}_{N,\epsilon}(x)}(\mathrm{d}\tilde{x})$ with $\mathbf{T}_{N,\epsilon}(x) := \int \tilde{x}\mathcal{P}_\epsilon^{\mathrm{OT,N}}(\mathrm{d}\tilde{x}|x)$ while, as $\mathcal{P}_\epsilon^{\mathrm{OT},N}$ belongs to $\mathcal{U}(\alpha_N, \beta_N)$, we also have $\beta_N(\mathrm{d}\tilde{x}) = \int \alpha_N(\mathrm{d}x)\mathcal{P}_\epsilon^{\mathrm{OT,N}}(\mathrm{d}\tilde{x}|x)$. We then have for any 1-Lipschitz function

$$\begin{aligned}\left|\beta_N(\psi)-\tilde{\beta}_N(\psi)\right| &= \left|\int\left[\int(\psi(\tilde{x})-\psi(\mathbf{T}_{N,\epsilon}(x)))\mathcal{P}_\epsilon^{\mathrm{OT,N}}(\mathrm{d}\tilde{x}|x)\right]\alpha_N(\mathrm{d}x)\right|\\ &\le \iint|\psi(\tilde{x})-\psi(\mathbf{T}_{N,\epsilon}(x))|\,\alpha_N(\mathrm{d}x)\mathcal{P}_\epsilon^{\mathrm{OT,N}}(\mathrm{d}\tilde{x}|x)\\ &\le \iint||\tilde{x}-\mathbf{T}_{N,\epsilon}(x)||\mathcal{P}_\epsilon^{\mathrm{OT,N}}(\mathrm{d}x,\mathrm{d}\tilde{x})\\ &\le \left(\iint||\tilde{x}-\mathbf{T}_{N,\epsilon}(x)||^2\mathcal{P}_\epsilon^{\mathrm{OT,N}}(\mathrm{d}x,\mathrm{d}\tilde{x})\right)^{\frac{1}{2}}\\ &\le \left(\iint||\tilde{x}-\mathbf{T}(x)||^2\mathcal{P}_\epsilon^{\mathrm{OT,N}}(\mathrm{d}x,\mathrm{d}\tilde{x})\right)^{\frac{1}{2}},\end{aligned}$$

where the final inequality follows from the fact that for any random vector V the mapping $v\mapsto\mathbb{E}[\|V-v\|^2]$ is minimized at $v=\mathbb{E}[V]$. The stated result is then obtained using Lemma C.2. □

D. Proof of Proposition 4.3

For technical reasons, we analyse here a slightly modified PF algorithm where

$$\alpha_N^{(t)}=\frac{1}{N}\sum_{j=1}^N\delta_{X_t^j},\qquad X_t^j\overset{\text{i.i.d.}}{\sim}\tilde{\beta}_N^{(t-1)}f=\frac{1}{N}\sum_{j=1}^N f\left(\cdot\middle|\tilde{X}_{t-1}^j\right). \tag{30}$$

instead of the standard version where one has

$$\alpha_N^{(t)}=\frac{1}{N}\sum_{j=1}^N\delta_{X_t^j},\qquad X_t^j\sim f\left(\cdot\middle|\tilde{X}_{t-1}^j\right).$$

This slightly modified version of the bootstrap PF was analyzed for example in (Del Moral & Guionnet, 2001). The analysis does capture the additional error arising from the use of DET instead of resampling. Similar results should hold for the standard PF algorithm. The main technical reason for analysing this modified algorithm is our reliance on Theorem 2 of (Fournier & Guillin, 2015); analysing the standard PF algorithm requires a version of (Fournier & Guillin, 2015) for stratified sampling and will be done in future work.

Proposition D.1. *Suppose that Assumptions B.1, B.2 and B.3 hold. Suppose also that given $\tilde{\beta}_N^{(t-1)}$, $\alpha_N^{(t)}$ is defined through (30). Define the functions*

$$\begin{aligned}\mathcal{F}(x)&:=x+\sqrt{\mathfrak{d}K_1(\Delta,\mathfrak{d})x}\\ f_d(x)&:=\begin{cases}x, & d<4\\ \frac{x}{\log(2+1/x)}, & d=4\\ x^{d/2}, & d>4.\end{cases}\\ \mathcal{F}_{N,\epsilon,\delta,d}(x)&:=\mathcal{F}\left(\kappa x+\sqrt{f_d^{-1}\left(\frac{\log(C/\delta)}{cN}\right)}\right),\\ \frac{1}{\mathfrak{d}}\mathfrak{G}^2_{\epsilon,\delta,N,d}(x)&:=2\lambda^{1/2}\left[\mathcal{F}_{N,\epsilon,\delta,d}(x)+\sqrt{2\epsilon\log N}\right]^{1/2}\left[\mathfrak{d}^{1/2}+\mathcal{F}_{N,\epsilon,\delta,d}(x)+\sqrt{2\epsilon\log N}\right]^{1/2}\\ &\quad+\lambda\kappa\mathcal{F}_{N,\epsilon,\delta,d}(x)+\max\{\lambda,1\}\mathcal{F}_{N,\epsilon,\delta,d}(x).\end{aligned}\tag{31}$$

Then for any $\epsilon,\delta>0$ we have with probability at least $1-\delta$, over the sampling step in (30), that

$$\mathcal{W}_2\left(\tilde{\beta}_N^{(t)},\beta^{(t)}\right)\le\mathfrak{G}_{\epsilon,\delta,N,d}\left[\mathcal{W}_2\left(\tilde{\beta}_N^{(t-1)},\beta^{(t-1)}\right)\right] \tag{32}$$

In particular if $\mathcal{W}_2(\tilde{\beta}_N^{(t-1)},\beta^{(t-1)})\to0$ and $\epsilon_N=o(1/\log(N))$ as $N\to\infty$ we have that

$$\mathcal{W}_2(\tilde{\beta}_N^{(t)},\beta^{(t)})\to0,$$

in probability.

Proof of Proposition D.1. To keep notation concise we write for $N \geq 1$

$$\alpha_N := \alpha_N^{(t)}, \quad \alpha'_N := \tilde{\beta}_N^{(t-1)} f, \quad \beta_N := \beta_N^{(t)}, \quad \tilde{\beta}_N := \tilde{\beta}_N^{(t-1)}.$$

Controlling $\mathcal{W}_1(\beta_N, \beta)$. Let ψ be 1-Lipschitz. Without loss of generality we may assume that $\psi(0) = 0$ since otherwise we can remove a constant.

$$\begin{aligned}
|\beta_N(\psi) - \beta(\psi)| &= \left|\frac{\alpha_N(\omega\psi)}{\alpha_N(\omega)} - \frac{\alpha(\omega\psi)}{\alpha(\omega)}\right| \\
&\leq \left|\frac{\alpha_N(\omega\psi)}{\alpha_N(\omega)} - \frac{\alpha(\omega\psi)}{\alpha_N(\omega)}\right| + \left|\frac{\alpha(\omega\psi)}{\alpha_N(\omega)} - \frac{\alpha(\omega\psi)}{\alpha(\omega)}\right| \\
&\leq \Delta^{-1}\left|\alpha_N(\omega\psi) - \alpha(\omega\psi)\right| + \Delta^{-2}\alpha(\omega\psi)|\alpha_N(\omega) - \alpha(\omega)|.
\end{aligned}$$

At this stage notice that

$$|(\omega\psi)'| \leq |\omega'\psi| + |\omega\psi'| \leq \|\psi\|_\infty + \|\omega\|_\infty.$$

Notice that

$$|\psi(x)| = |\psi(x) - \psi(0)| \leq |x - 0| \leq \mathfrak{d}.$$

Therefore we have that

$$|(\omega\psi)'| \leq \mathfrak{d} + \Delta^{-1},$$

and thus $\omega\psi$ is $(\mathfrak{d} + \Delta^{-1})$-Lipschitz. It follows that

$$\begin{aligned}
|\beta_N(\psi) - \beta(\psi)| &\leq \Delta^{-1}\left|\alpha_N(\omega\psi) - \alpha(\omega\psi)\right| + \Delta^{-2}\alpha(\omega\psi)|\alpha_N(\omega) - \alpha(\omega)| \\
&\leq \Delta^{-1}(\mathfrak{d} + \Delta^{-1})\mathcal{W}_1(\alpha_N, \alpha) + \Delta^{-3}\mathfrak{d}\mathcal{W}_1(\alpha_N, \alpha) \\
&=: K_1(\Delta, \mathfrak{d})\mathcal{W}_1(\alpha_N, \alpha).
\end{aligned}$$

Therefore we have that

$$\mathcal{W}_1(\beta_N, \beta) \leq K_1(\Delta, \mathfrak{d})\mathcal{W}_1(\alpha_N, \alpha). \tag{33}$$

Notice that using the compactness of the state space we easily get also that

$$\mathcal{W}_2(\beta_N, \beta) \leq \sqrt{\mathfrak{d}\mathcal{W}_1(\beta_N, \beta)} \leq \sqrt{\mathfrak{d}K_1(\Delta, \mathfrak{d})\mathcal{W}_1(\alpha_N, \alpha)} \leq \sqrt{\mathfrak{d}K_1(\Delta, \mathfrak{d})\mathcal{W}_2(\alpha_N, \alpha)}, \tag{34}$$

since clearly $\mathcal{W}_1(\rho, \sigma) \leq \mathcal{W}_2(\rho, \sigma)$ for any two probability measures ρ, σ.

Controlling $\mathcal{W}_1(\tilde{\beta}_{N,\epsilon}, \beta)$. Again supposing ψ is 1-Lipschitz, and $\psi(0) = 0$, consider

$$\begin{aligned}
\left|\tilde{\beta}_N(\psi) - \tilde{\beta}(\psi)\right| &= \left|\int \psi(\mathbf{T}_{N,\epsilon}(x))\alpha_N(\mathrm{d}x) - \int \psi(\mathbf{T}(x))\alpha(\mathrm{d}x)\right| \\
&\leq \left|\int \psi(\mathbf{T}_{N,\epsilon}(x))\alpha_N(\mathrm{d}x) - \int \psi(\mathbf{T}(x))\alpha_N(\mathrm{d}x)\right| \\
&\quad + \left|\int \psi(\mathbf{T}(x))\alpha_N(\mathrm{d}x) - \int \psi(\mathbf{T}(x))\alpha(\mathrm{d}x)\right|
\end{aligned}$$

For the second term, using the fact that $\mathbf{T}$ and ψ are λ- and 1-Lipschitz respectively, we have that $\psi \circ \mathbf{T}$ is λ-Lipschitz and therefore

$$\left|\int \psi(\mathbf{T}(x))\alpha_N(\mathrm{d}x) - \int \psi(\mathbf{T}(x))\alpha(\mathrm{d}x)\right| \leq \lambda\mathcal{W}_1(\alpha_N, \alpha) \leq \lambda\mathcal{W}_2(\alpha_N, \alpha),$$

where we used Assumption B.2 for that last inequality For the first term recall that using Cauchy-Schwarz and Jensen we get

$$\begin{aligned}
&\left|\int \psi(\mathbf{T}_{N,\epsilon}(x))\alpha_N(\mathrm{d}x) - \int \psi(\mathbf{T}(x))\alpha_N(\mathrm{d}x)\right| \\
&\leq \int \left|\mathbf{T}_{N,\epsilon}(x) - \mathbf{T}(x)\right| \alpha_N(\mathrm{d}x) \\
&\leq \int \left|\int y\mathcal{P}_{N,\epsilon}(x,\mathrm{d}y) - \mathbf{T}(x)\right| \alpha_N(\mathrm{d}x) \\
&\leq \iint \left|y - \mathbf{T}(x)\right| \alpha_N(\mathrm{d}x)\mathcal{P}_{N,\epsilon}(x,\mathrm{d}y) \\
&\leq \left[\iint \left|y - \mathbf{T}(x)\right|^2 \alpha_N(\mathrm{d}x)\mathcal{P}_{N,\epsilon}(x,\mathrm{d}y)\right]^{1/2}.
\end{aligned}$$

Here we can directly apply Lemma C.2 to obtain

$$\begin{aligned}
&\left[\iint \left|y - \mathbf{T}(x)\right|^2 \alpha_N(\mathrm{d}x)\mathcal{P}_{N,\epsilon}(x,\mathrm{d}y)\right]^{1/2} \\
&\leq 2\lambda^{1/2}\mathcal{E}^{1/2}\left[\mathfrak{d}^{1/2} + \mathcal{E}\right]^{1/2} + \max\{\lambda, 1\}\left[\mathcal{W}_2(\alpha_N,\alpha) + \mathcal{W}_2(\beta_N,\beta)\right],
\end{aligned}$$

where

$$\mathcal{E} := \mathcal{E}(n,\epsilon,\alpha,\beta) := \mathcal{W}_2(\alpha_N,\alpha) + \mathcal{W}_2(\beta_N,\beta) + \sqrt{2\epsilon\log(N)}.$$

From (34) we have that

$$\mathcal{W}_2(\alpha_N,\alpha) + \mathcal{W}_2(\beta_N,\beta) \leq \mathcal{W}_2(\alpha_N,\alpha) + \sqrt{\mathfrak{d}K_1(\Delta,\mathfrak{d})\mathcal{W}_2(\alpha_N,\alpha)}.$$

Next we want to bound $\mathcal{W}_2(\alpha_N,\alpha)$. Notice first that

$$\mathcal{W}_2(\alpha_N,\alpha) \leq \mathcal{W}_2(\alpha_N,\alpha'_N) + \mathcal{W}_2(\alpha'_N,\alpha) \leq \mathcal{W}_2(\alpha_N,\alpha'_N) + \kappa\mathcal{W}_2\left(\tilde{\beta}_N^{(t-1)},\beta^{(t-1)}\right),$$

by Assumption B.2.

To control the other term we use (Fournier & Guillin, 2015) to obtain a high probability bound on $\mathcal{W}_2(\alpha_N,\alpha'_N)$. In particular, using Theorem 2 from (Fournier & Guillin, 2015), with $\alpha=\infty$ since we are in a compact domain, that for some positive constants C, c we have

$$\mathbb{P}\left[\mathcal{W}_2^2(\alpha_N,\alpha'_N) \geq x\right] \leq C\exp\left[-cNf_d^2(x)\right], \tag{35}$$

where

$$f_d(x) := \begin{cases} x, & d < 4 \\ \frac{x}{\log(2+1/x)}, & d = 4 \\ x^{d/2}, & d > 4. \end{cases} \tag{36}$$

In particular, for any $\delta > 0$, with probability at least $1-\delta$ over the sampling step in F_N we have that

$$\mathcal{W}_2(\alpha_N,\alpha'_N) \leq \sqrt{f_d^{-1}\left(\frac{\log(C/\delta)}{cN}\right)}. \tag{37}$$

Assuming that $d \geq 4$ the rate then is of order $N^{-1/d}$ as expected.

Therefore with probability at least $1-\delta$ over the sampling step we have that

$$\mathcal{W}_2(\alpha_N,\alpha) + \mathcal{W}_2(\beta_N,\beta) \leq \mathcal{F}_{N,\epsilon,\delta,d}\left(\mathcal{W}_2\left(\tilde{\beta}_N^{(t-1)},\beta^{(t-1)}\right)\right),$$

where

$$\mathcal{F}_{N,\epsilon,\delta,d}(x) = \mathcal{F}\left(\kappa x + \sqrt{f_d^{-1}\left(\frac{\log(C/\delta)}{cN}\right)}\right), \qquad \mathcal{F}(x) := x + \sqrt{\mathfrak{d}K_1(\Delta,\mathfrak{d})x} \tag{38}$$

Thus overall we have with probability at least $1-\delta$ over the sample

$$\mathcal{W}_2(\tilde{\beta}_{N,\epsilon},\tilde{\beta}) \leq \sqrt{\mathfrak{d}\mathcal{W}_1(\tilde{\beta}_{N,\epsilon},\tilde{\beta})} \leq \mathfrak{G}_{\epsilon,\delta,N,d}\left(\mathcal{W}_2\left(\tilde{\beta}_N^{(t-1)},\beta^{(t-1)}\right)\right),$$

where

$$\frac{1}{\mathfrak{d}}\mathfrak{G}^2_{\epsilon,\delta,N,d}(x) := 2\lambda^{1/2}\left[\mathcal{F}_{N,\epsilon,\delta,d}(x)+\sqrt{2\epsilon\log N}\right]^{1/2}\left[\mathfrak{d}^{1/2}+\mathcal{F}_{N,\epsilon,\delta,d}(x)+\sqrt{2\epsilon\log N}\right]^{1/2} + \lambda\kappa\mathcal{F}_{N,\epsilon,\delta,d}(x)+\max\{\lambda,1\}\mathcal{F}_{N,\epsilon,\delta,d}(x).$$

In particular notice that if we set $\epsilon_N = o(1/\log N)$ and $x_N = o(1)$ we have

$$\mathfrak{G}_{\epsilon_N,\delta,N,d}(x_N) \to 0.$$

Therefore, notice that if $\epsilon_N = o(1/\log N)$ and $\mathcal{W}_2(\mu_N,\mu)\to 0$, then for any $x>0$ we have that

$$\mathbb{P}\left[\mathcal{W}_2(\tilde{\beta}_{N,\epsilon},\tilde{\beta})\geq x\right] \leq \mathbb{P}[\mathcal{W}_2(\alpha'_N,\alpha_N)\geq x'],$$

for some x' that does not depend on N, where the probability is over the sampling step. The convergence in probability follows. □

Proposition D.2. *Let* $\mu_N = \frac{1}{N}\sum_{i=1}^N \delta_{X_1^i}$ *where* $X_1^i \overset{\text{i.i.d.}}{\sim} \mu := q(\cdot|y_1)$ *for* $i\in[N]$ *and suppose that for* $t\geq 1$, $\alpha_N^{(t)}$ *is defined through* (30). *Under Assumptions B.1, B.2, B.3 and B.4, for any* $\delta>0$, *with probability at least* $1-2\delta$ *over the sampling steps, for any bounded 1-Lipschitz* ψ, *for any* $t\in[1:T]$, *the approximations of the filtering distributions and log-likelihood computed by DPF satisfy*

$$|\tilde{\beta}_N^{(t)}(\psi)-\beta^{(t)}(\psi)| \leq \mathfrak{G}^{(t)}_{\epsilon,\delta/T,N,d}\left(\sqrt{f_d^{-1}\left(\frac{\log(CT/\delta)}{cN}\right)}\right) \tag{39}$$

$$\left|\log\frac{\hat{p}_N(y_{1:T})}{p(y_{1:T})}\right| \leq \frac{\kappa}{\Delta}\max_{t\in[1:T]}\mathrm{Lip}\left[g(y_t\mid\cdot)\right]\sum_{t=1}^T \mathfrak{G}^{(t)}_{\epsilon,\delta/T,N,d}\left(\sqrt{f_d^{-1}\left(\frac{\log(CT/\delta)}{cN}\right)}\right) \tag{40}$$

where C *is a finite constant independent of* T, $\mathfrak{G}_{\epsilon,\delta/T,N,d}$, f_d *are defined in* (31), *and* $\mathrm{Lip}[f]$ *is the Lipschitz constant of the function* f. $\mathfrak{G}^{(t)}_{\epsilon,\delta/T,N,d}$ *denotes the t-repeated composition of function* $\mathfrak{G}_{\epsilon,\delta/T,N,d}$. *In particular, if we set* $\epsilon_N = o(1/\log N)$

$$\left|\log\frac{\hat{p}_N(y_{1:T})}{p(y_{1:T})}\right| \to 0,$$

in probability.

Proof of Proposition D.2. Following the proof of Proposition D.1, we define $\alpha_N^{(t)'} = \tilde{\beta}_N^{(t-1)}f$ and for $t\in[1:T]$, the events

$$A_t := \mathcal{W}_2\left(\alpha_N^{(t)},\alpha_N^{(t)'}\right) \leq \sqrt{f_d^{-1}\left(\frac{\log(CT/\delta)}{cN}\right)}.$$

We know from Theorem 2 in (Fournier & Guillin, 2015) that $\mathbb{P}(A_t)\geq 1-\delta/T$, where the probability is over the sampling step. In particular we have that

$$\mathbb{P}\left[\bigcap_{t=1}^T A_t\right] = 1-\mathbb{P}\left[\bigcup_{t=1}^T A_t^{\complement}\right] \geq 1-\sum_{t=1}^N \mathbb{P}\left[A_t^{\complement}\right] \geq 1 - T\frac{\delta}{T} = 1-\delta.$$

Notice that on the event $\cap_{t=1}^T A_t$, iterating the bound (32) we have

$$\mathcal{W}_2\left(\tilde{\beta}_N^{(t)},\beta^{(t)}\right) \leq \mathfrak{G}^{(t)}_{\epsilon,\delta/T,N,d}\left(\mathcal{W}_2\left(\mu_N,\mu\right)\right),$$

with probability at least $1-\delta$. Again by Theorem 2 in (Fournier & Guillin, 2015) we have that with probability at least $1-\delta$

$$\mathcal{W}_2(\mu_N,\mu) \leq \sqrt{f_d^{-1}\left(\frac{\log(CT/\delta)}{cN}\right)}.$$

Therefore with probability at least $1-2\delta$ we have

$$\mathfrak{G}^{(t)}_{\epsilon,\delta/T,N,d}\left(\sqrt{f_d^{-1}\left(\frac{\log(CT/\delta)}{cN}\right)}\right).$$

It remains to prove (40). Note that $|\log(x)-\log(y)| \leq \frac{|x-y|}{\min\{x,y\}}$ for any $x,y>0$ so

$$\begin{aligned}\left|\log\hat{p}(y_{1:T})-\log p(y_{1:T})\right| &\leq \sum_{t=1}^{T}\left|\log\hat{p}(y_t|y_{1:t-1})-\log p(y_t|y_{1:t-1})\right|\\ &\leq \sum_{t=1}^{T}\left|\frac{\hat{p}(y_t|y_{1:t-1})-p(y_t|y_{1:t-1})}{\min(\hat{p}(y_t|y_{1:t-1}),p(y_t|y_{1:t-1}))}\right|\\ &\leq \Delta^{-1}\sum_{t=1}^{T}\left|\hat{p}(y_t|y_{1:t-1})-p(y_t|y_{1:t-1})\right| \qquad (41)\end{aligned}$$

where Δ is defined in Assumption B.3.

The term in line (41) may be written as follows

$$\begin{aligned}&\hat{p}(y_t|y_{1:t-1})-p(y_t|y_{1:t-1})\\ =&\iint g(y_t|x_t)f(\mathrm{d}x_t|\tilde{x}_{t-1})\tilde{\beta}_N^{(t-1)}(\mathrm{d}\tilde{x}_{t-1})-\iint g(y_t|x_t)f(\mathrm{d}x_t|\tilde{x}_{t-1})\tilde{\beta}^{(t-1)}(\mathrm{d}\tilde{x}_{t-1})\\ =&\tilde{\beta}_N^{(t-1)}(h)-\beta^{(t-1)}(h)\end{aligned}$$

for $\Delta^2 \leq h(x) := \int g(y_t|x')f(x'|x)\mathrm{d}x' \leq \Delta^{-2}$. At this point notice also that

$$\begin{aligned}h(x)-h(x') &= \int f(\mathrm{d}w|x)g(y_t\mid w)-\int f(\mathrm{d}w|x')g(y_t\mid w)\\ &= \int \delta_x(\mathrm{d}z)\int f(\mathrm{d}w|z)g(y_t\mid w)-\int \delta_{x'}(\mathrm{d}z)\int f(\mathrm{d}w|z)g(y_t\mid w)\\ &= [\delta_x f][g(y_t\mid\cdot)]-[\delta_{x'}f][g(y_t\mid\cdot)]\\ &\leq \mathrm{Lip}\left[g(y_t\mid\cdot)\right]\mathcal{W}_1\left(\delta_x f,\delta_{x'}f\right) \leq \kappa\mathrm{Lip}\left[g(y_t\mid\cdot)\right]\mathcal{W}_1\left(\delta_x,\delta_{x'}\right)=\kappa\mathrm{Lip}\left[g(y_t\mid\cdot)\right]|x-x'|,\end{aligned}$$

by Assumption B.2. It follows therefore that h is Lipschitz and therefore that

$$\hat{p}(y_t|y_{1:t-1})-p(y_t|y_{1:t-1})=\tilde{\beta}_N^{(t-1)}(h)-\beta^{(t-1)}(h)\leq \kappa\mathrm{Lip}\left[g(y_t\mid\cdot)\right]\mathcal{W}_1\left(\beta_N^{(t-1)},\beta^{(t-1)}\right).$$

Combining (39) and (41), and using the fact that $\mathcal{W}_1\leq\mathcal{W}_2$, we thus get

$$\begin{aligned}\left|\log\hat{p}(y_{1:T})-\log p(y_{1:T})\right| &\leq \Delta^{-1}\kappa\sum_{t=1}^{T}\mathrm{Lip}\left[g(y_t\mid\cdot)\right]\mathcal{W}_1\left(\tilde{\beta}_N^{(t-1)},\beta^{(t)}\right)\\ &\leq \Delta^{-1}\kappa\max_{t\in[1:T]}\mathrm{Lip}\left[g(y_t\mid\cdot)\right]\sum_{t=1}^{T}\mathfrak{G}^{(t)}_{\epsilon,\delta/T,N,d}\left(\sqrt{f_d^{-1}\left(\frac{\log(CT/\delta)}{cN}\right)}\right),\end{aligned}$$

where the last inequality holds with probability at least $1-\delta$ over the sampling steps.

The convergence in probability follows from the corresponding statement of Proposition D.1. □

E. Additional Experiments and Details

E.1. Linear Gaussian model

We first consider the following 2-dimensional linear Gaussian SSM for which exact inference can be carried out using Kalman techniques:

$$X_t|\{X_{t-1}=x\} \sim \mathcal{N}\left(\text{diag}(\theta_1\ \theta_2)x, 0.5\mathbf{I}_2\right), \quad Y_t|\{X_t=x\} \sim \mathcal{N}(x, 0.1\cdot\mathbf{I}_2). \tag{42}$$

We simulate $T = 150$ observations using $\theta = (\theta_1, \theta_2) = (0.5, 0.5)$. As a result, we expect in these scenarios that the filtering distribution $p_\theta(x_t|y_{1:t})$ is not too distinct from the smoothing distribution $p_\theta(x_t|y_{1:T})$ as the latent process is mixing quickly. From Proposition 4.1, this is thus a favourable scenario for methods ignoring resampling terms in the gradient as the bias should not be very large. Figure 1, displayed earlier, shows $\ell(\theta)$ obtained by Kalman and $\hat{\ell}(\theta;\mathbf{u})$ computed regular PF and DPF for the same number $N = 25$ of particles using $q_\phi(x_t|x_{t-1}, y_t) = f_\theta(x_t|x_{t-1})$. The corresponding gradient vector fields are given in Figure 1, where the gradient is computed using the biased gradient from (Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2018) for regular PF.

We now compare the performance of the estimators $\hat{\theta}_{\text{SMLE}}$ (for DPF) and $\hat{\theta}_{\text{ELBO}}$ (for both regular PF and DPF) learned using gradient with learning rate 10^{-4} on 100 steps, using $N = 25$ for DPF and $N = 500$ for regular PF, to $\hat{\theta}_{\text{MLE}}$ computed using Kalman derivatives. We simulate $M = 50$ realizations of $T = 150$ observations using $\theta = (\theta_1, \theta_2) = (0.5, 0.5)$. The ELBO stochastic gradient estimates are computed using biased gradient estimates of $\ell_{\text{ELBO}}(\theta)$ ignoring the contributions of resampling steps as in (Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2018) (we recall that unbiased estimates suffer from very high variance) and unbiased gradients of $\ell^{\text{ELBO}}(\theta)$ using DPF. We average B parallel PFs to reduce the variance of these gradients of the ELBO and also B PFs (with fixed random seeds) to compute the gradient of $\hat{\ell}_{\text{SMLE}(\theta;\mathbf{u}_{1:B})} := \frac{1}{B}\sum_{b=1}^{B}\hat{\ell}(\theta;\mathbf{u}_b)$. The results are given in Table 4. For this example, $\hat{\theta}^{\text{DPF}}_{\text{ELBO}}$ maximizing $\ell^{\text{ELBO}}_{\text{DPF}}(\theta)$ outperforms $\hat{\theta}^{\text{PF}}_{\text{ELBO}}$ and $\hat{\theta}_{\text{SMLE}}$. However, as B increases, $\hat{\theta}_{\text{SMLE}}$ gets closer to $\hat{\theta}^{\text{DPF}}_{\text{ELBO}}$ which is to be expected as $\hat{\ell}_{\text{SMLE}}(\theta;\mathbf{u}_{1:B}) \longrightarrow \ell^{\text{ELBO}}(\theta)$. In Table 4, the Root Mean Square Error (RMSE) is defined as $\sqrt{\frac{1}{M}\sum_{i=1}^{2}\sum_{k=1}^{M}(\hat{\theta}_i^k - \hat{\theta}^k_{\text{MLE},i})^2}$.

Table 4. $10^3\times$ RMSE[4] over 50 datasets - lower is better

B	$\hat{\theta}^{\text{PF}}_{\text{ELBO}}$	$\hat{\theta}^{\text{DPF}}_{\text{ELBO}}$	$\hat{\theta}_{\text{SMLE}}$
1	1.94	1.30	7.94
4	2.40	1.35	3.28
10	2.80	1.37	2.18

E.2. Variational Recurrent Neural Network

$N = 32$ particles were used for training, with a regularization parameter of $\epsilon = 0.5$. The ELBO (scaled by sequence length) was used as the training objective to maximise for each resampling/ DET procedure. The ELBO evaluated on test data using $N = 500$ particles and multinomial resampling. Resampling / DET operations were carried out when effective sample (ESS) size fell below $N/2$. Learning rate 0.001 was used with the Adam optimizer.

Recall the state-space model is given by

$$\begin{aligned}(R_t, O_t) &= \text{RNN}_\theta(R_{t-1}, Y_{1:t-1}, E_\theta(Z_{t-1})),\\ Z_t &\sim \mathcal{N}(\mu_\theta(O_t), \sigma_\theta(O_t)),\\ \hat{p}_t &= h_\theta(E_\theta(Z_t), O_t),\\ Y_t|X_t &\sim \text{Ber}(\hat{p}_t).\end{aligned}$$

Network architectures and data preprocessing steps were based loosely on (Maddison et al., 2017). Given the low volume of data and sparsity of the observations, relatively small neural networks were considered to prevent overfitting, larger neural

[4]The Root Mean Square Error (RMSE) is defined as $\sqrt{\frac{1}{M}\sum_{i=1}^{2}\sum_{k=1}^{M}(\hat{\theta}_i^k - \hat{\theta}^k_{\text{MLE},i})^2}$.

networks are considered in the more complex robotics experiments. R_t is of dimension $d_r = 16$, Z_t is of dimension $d_z = 8$. E_θ is a single layer fully connected network with hidden layer of width 16, output of dimension 16 and RELU activation.

μ_θ and σ_θ are both fully connected neural networks with two hidden layers, each of 16 units and RELU activation, the activation function is not applied to the final output of μ_θ but the softplus is applied to the output of σ_θ, which is the diagonal entries of the covariance matrix of the normal distribution that is used to sample Z_t.

h_θ is a single layer fully connected network with two hidden layers, each of width 16 and RELU activation. The final output is not put through the RELU and is instead used as the logits for the Bernoulli distribution of observations.

E.3. Robot Localization

Similar to the VRNN example, $N = 32$ particles were used for training, with a regularization parameter of $\epsilon = 0.5$ and resampling / DET operations were carried out when ESS size fell below $N/2$. Learning rate 0.001 was used with the Adam optimizer.

Network architectures and data preprocessing were based loosely on (Jonschkowski et al., 2018). There are 3 neural networks being considered:

- Encoder E_θ maps RBG 24×24 pixel images, hence dimension $3 \times 24 \times 24$, to encoding of size $d_E = 128$. This network consists of a convolutional network (CNN) of kernel size 3 and a single layer fully connected network of hidden width 128 and RELU activation.
- Decoder D_θ maps encoding back to original image. This consists of a fully connected neural network with three hidden layers of width 128 and RELU activation function. This is followed by a transposed convolution network with matching specification to the CNN in the encoder, to return an output with the same dimension as observation images, $3 \times 24 \times 24$.
- Network G_θ maps the state $S_t = (X_t^{(1)}, X_t^{(2)}, \gamma_t)$ to encoding of dimension 128. First angle γ_t was converted to $\sin(\gamma_t), \cos(\gamma_t)$. Then the augmented state $(X_t^{(1)}, X_t^{(2)}, \sin(\gamma_t), \cos(\gamma_t))$ was passed to a 3 layer fully connected network with hidden layers of dimensions $16, 32, 64$ and RELU activation function, with final output of dimension 128.

Publication II

Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä. Parallel Iterated Extended and Sigma-Point Kalman Smoothers. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Pages 5350–5354, June 2021.

ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413364

PARALLEL ITERATED EXTENDED AND SIGMA-POINT KALMAN SMOOTHERS

Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, Simo Särkkä

Department of Electrical Engineering and Automation, Aalto University, Finland

ABSTRACT

The problem of Bayesian filtering and smoothing in nonlinear models with additive noise is an active area of research. Classical Taylor series as well as more recent sigma-point based methods are two well-known strategies to deal with this problem. However, these methods are inherently sequential and do not in their standard formulation allow for parallelization in the time domain. In this paper, we present a set of parallel formulas that replace the existing sequential ones in order to achieve lower time (span) complexity. Our experimental results done with a graphics processing unit (GPU) illustrate the efficiency of the proposed methods over their sequential counterparts.

Index Terms— parallel computing, nonlinear estimation, iterated extended Kalman smoother, sigma-point smoother

1. INTRODUCTION

In recent years, the rapid advancements in hardware technologies such as graphics processing units (GPUs) and tensor processing units (TPUs) allow compute-intensive workloads to be offloaded from the central processing units (CPUs) by introducing parallelism [1–3]. There is a wide variety of areas that can benefit from parallelization [4], one of which is state estimation.

State estimation is a common task that arises in various areas of science and engineering [5–7]. It aims at combining the noisy measurements and the model to estimate the hard-to-measure states. A frequent and classical method for solving this problem is based on Bayesian filtering and smoothing [5] which inherently provides a sequential solution with linear complexity in the number of time steps.

In order to tackle the computational burden of Kalman type of filters and smoothers, [8,9] provide sub-linear computational methods by taking advantage of the sparse structures of the matrices appearing in the batch forms of the problems. In other works, using an ensemble formulation of Kalman filter has been used to speed up the matrix computations through parallelization [10,11]. The primary focus of these works was the efficient computation of the covariance matrices either by introducing sparse or sample covariance matrices rather than considering the temporal state-space structure per se. While in the aforementioned works, parallelization of the sub-problems in the area of Bayesian filtering and smoothing were considered, [12] presented a general parallelizable formulations specifically designed for parallelizing state-estimation problems in the temporal direction. Moreover, for the special case of linear Gaussian model, parallel equations for computing Kalman filter and Raugh–Tung–Striebel smoother solutions were derived.

The authors would like to thank Academy of Finland for funding.

Overcoming the computational burden in the case of nonlinear dynamical systems with additive Gaussian noise is also of paramount importance. In these types of models, various linearization approaches can be used. Taylor series expansion based iterated extended Kalman smoother (IEKS) methods [13–15] and sigma-point based methods [5] are well-established techniques in literature. Iterated sigma-point methods have been proposed, for example, in [16, 17]. Despite the capabilities of the aforementioned methods in state estimation in nonlinear Gaussian models, they lack a framework which enables the computations in a more efficient way when using parallelization.

The contribution of this paper is to present a set of parallelizable formulas for filtering and smoothing in nonlinear Gaussian systems, in particular, IEKS and sigma-point based methods using a scan algorithm [12,18]. The proposed methods reduce the linear span complexity of the state estimation methods to logarithmic with respect to the number of measurements.

This paper is organized as follows: Section 2 briefly reviews the generic parallel framework for Bayesian filters and smoothers. Sections 3 and 4 are concerned with presenting the formulation of the problem and proposing our method. Section 5 analyzes the efficiency and the computational complexity of the proposed method through one numerical example, and Section 6 concludes the paper.

2. GENERAL PARALLEL FRAMEWORK FOR BAYESIAN FILTERS AND SMOOTHERS

It is shown in [12] that the computation of sequential Bayesian filtering and smoothing can be converted to general parallel formulas in terms of associative operations. This allows for the use of the parallel scan method [18] which is a common algorithm used to speed-up sequential computations, for example, on GPU-based computing systems. In the rest of this section, we review the general parallel algorithms provided in [12] which we then extend to nonlinear Gaussian models.

Given a state space model of the following form:

$$x_k \sim p(x_k \mid x_{k-1}), \quad y_k \sim p(y_k \mid x_k), \tag{1}$$

the goal of the filtering problem is to find the posterior distributions $p(x_k \mid y_{1:k})$ for $k = 1, \ldots, n$. This distribution is a probabilistic representation of the available statistical information on the state $x_k \in \mathbb{R}^{n_x}$ given the measurements $y_{1:k} = \{y_1, \ldots, y_k\}$ with $y_k \in \mathbb{R}^{n_y}$. Having acquired the filtering results for $k = 1, \ldots, n$, and using all the n measurements, the Bayesian smoother can be used to compute the posterior distributions $p(x_k \mid y_{1:n})$. The following strategies are used in [12] so as to particularize a_k and the binary associative operator $\otimes$ which provide a parallel framework for solving the aforementioned sequential filtering and smoothing problem.

Filtering. Given two positive functions $g'_i(y), g'_j(y)$ and two conditional densities $f'_i(x \mid y), f'_j(x \mid y)$, the authors of [12] proved that the binary operation $(f'_i, g'_i) \otimes (f'_j, g'_j) = (f'_{ij}, g'_{ij})$ defined by

$$\begin{aligned} f'_{ij}(x|z) &= \frac{\int g'_i(y) f'_j(x \mid y) f'_i(y \mid z) dy}{\int g'_j(y) f'_i(y \mid z) dy}, \\ g'_{ij}(z) &= g'_i(z) \int g'_j(y) f'_i(y \mid z) dy, \end{aligned} \tag{2}$$

is associative and by selecting $a_k = (f'_k, g'_k)$ as follows:

$$\begin{aligned} f'_k(x_k \mid x_{k-1}) &= p(x_k \mid y_k, x_{k-1}), \\ g'_k(x_{k-1}) &= p(y_k \mid x_{k-1}), \end{aligned} \tag{3}$$

where $p(x_1 \mid y_1, x_0) = p(x_1 \mid y_1)$ and $p(y_1 \mid x_0) = p(y_1)$, the Bayesian map $\binom{p(x_k|y_{1:k})}{p(y_{1:k})}$ can be rewritten as the k-th prefix sum, $a_1 \otimes \cdots \otimes a_k$.

Smoothing. Similarly [12], for any conditional densities $f'_i(x \mid y)$ and $f'_j(x \mid y)$ the binary operation $f'_i \otimes f'_j := \int f'_i(x|y) f'_j(y|z) dy$ is associative and by selecting $a_k = p(x_k \mid y_{1:k}, x_{k+1})$ with $a_n = p(x_n \mid y_{1:n})$, the Bayesian smoothing solution can then be calculated as $p(x_k \mid y_{1:n}) = a_k \otimes a_{k+1} \otimes \cdots \otimes a_n$.

Having considered the aforementioned general formulations, in this paper, we aim to extend the element a_k and the binary associative operator $\otimes$ to linear approximations of non-linear Gaussian systems, specifically, to the extended Kalman filter and smoother, and sigma-points methods.

3. PROBLEM FORMULATION

We consider the following model:

$$\begin{aligned} x_k &= f_{k-1}(x_{k-1}) + q_{k-1}, \\ y_k &= h_k(x_k) + r_k, \end{aligned} \tag{4}$$

where $f_{k-1}(.)$ and $h_k(.)$ are nonlinear functions. The q_k and r_k are the process and measurement noises, which are assumed to be zero-mean, independent Gaussian noises with known covariance matrices, Q_k and R_k, respectively. Furthermore, the initial state is Gaussian $x_0 \sim N(m_0, P_0)$ with known mean m_0 and covariance P_0. This paper is concerned with the computing approximate posterior distributions of the states $x_{0:n} = \{x_0, x_1, \ldots, x_n\}$ given all the measurements $y_{1:n} = \{y_1, x_1, \ldots, y_n\}$ in parallel form, or more precisely, the corresponding filtering and smoothing distributions.

Since the filtering and smoothing problems are not solvable in closed-form in the general non-linear case, one needs to resort to approximations. Here we follow the Gaussian filtering and smoothing frameworks [5] and form linear approximations of the system (4) in the following form:

$$\begin{aligned} f_{k-1}(x_{k-1}) &\approx F_{k-1} x_{k-1} + c_{k-1} + e_{k-1}, \\ h_k(x_k) &\approx H_k x_k + d_k + v_k, \end{aligned} \tag{5}$$

where $F_k \in \mathbb{R}^{n_x \times n_x}$, $c_k \in \mathbb{R}^{n_x}$, $H_k \in \mathbb{R}^{n_y \times n_x}$, $d_k \in \mathbb{R}^{n_y}$, $e_k \in \mathbb{R}^{n_x}$ and $v_k \in \mathbb{R}^{n_y}$ are zero mean Gaussian noises with covariance matrices Λ_k and Ω_k, respectively.

There are different strategies to effectively select the parameters of (5). In this paper, we will consider two such strategies widely-used in the Gaussian filtering literature, namely iterated sigma-point and extended Kalman smoothers [14–16]. In these approaches, the linearized-filter-smoother method is repeated M times, with the linearization parameters leveraging the results of the previous smoothing pass instead of the previous step. We can therefore see our successive linear approximations as being parametrized by the following vectors and matrices:

$$F^{(i)}_{0:n-1}, c^{(i)}_{0:n-1}, \Lambda^{(i)}_{0:n-1}, H^{(i)}_{0:n-1}, d^{(i)}_{1:n}, \Omega^{(i)}_{1:n}. \tag{6}$$

In the rest of this section, we will discuss how to acquire the linearized parameters of (6) using these methods. Also, for the sake of notational simplicity, we drop the index i from these parameters.

Iterated sigma-point method. In this approach, we select the parameters $(F_{k-1}, c_{k-1}, \Lambda_{k-1})$ and (H_k, d_k, Ω_k) using sigma-point-based statistical linear regression (SLR) method [16] as follows. First, we select m sigma points $\mathcal{X}^{(i)}_{1,k}, \ldots, \mathcal{X}^{(i)}_{m,k}$ and their associated weights $w^{(i)}_{1,k}, \ldots, w^{(i)}_{m,k}$ according to the posterior moments $\bar{x}^{(i-1)}_k$ and $\bar{P}^{(i-1)}_k$ of the previous iteration, which are the best available estimates for the means and covariances of the smoothing distribution. Then, in order to find the parameters $(F_{k-1}, c_{k-1}, \Lambda_{k-1})$, transformed sigma-points are obtained as $\mathcal{Z}_j = f_{k-1}(\mathcal{X}^{(i)}_{j,k-1})$ for $j = 1, \ldots, m$, and the linearization parameters are then given by:

$$\begin{aligned} F_{k-1} &= \Psi^\top \bar{P}_{k-1}^{-1}, \\ c_{k-1} &= \bar{z} - F_{k-1}\bar{x}_{k-1}, \\ \Lambda_{k-1} &= \Phi - F_{k-1}\bar{P}_{k-1}F_{k-1}^\top. \end{aligned} \tag{7}$$

If we now write $\bar{x} = \bar{x}_{k-1}$ and $w_j = w_{j,k-1}^{(i)}$, the required moment approximations for Equation (7) are [19]:

$$\begin{aligned} \bar{z} &\approx \sum_{j=1}^{m} w_j \mathcal{Z}_j, \\ \Psi &\approx \sum_{j=1}^{m} w_j (\mathcal{X}_j - \bar{x})(\mathcal{Z}_j - \bar{z})^\top, \\ \Phi &\approx \sum_{j=1}^{m} w_j (\mathcal{Z}_j - \bar{z})(\mathcal{Z}_j - \bar{z})^\top. \end{aligned} \tag{8}$$

Similarly, reusing Equations (8) with $\bar{x} = \bar{x}_k$, $w_j = w_{j,k}^{(i)}$, and $\mathcal{Z}_j = h_k(\mathcal{X}_{j,k}^{(i)})$ the parameters (H_k, d_k, Ω_k) can be calculated as follows:

$$\begin{aligned} H_k &= \Psi^\top \bar{P}_k^{-1}, \\ d_k &= \bar{z} - H_k\bar{x}_k, \\ \Omega_k &= \Phi - H_k\bar{P}_kH_k^\top. \end{aligned} \tag{9}$$

The iterated posterior linearization smoother (IPLS) [16] now consists in iterating Equations (7) and (9) with updated approximate means and covariances of the posterior distribution at each iteration.

Iterated extended Kalman smoother. In this case, Ω and Λ are selected as zeros, and (F_{k-1}, c_{k-1}) and (H_k, d_k) are obtained by analytical linearization at the previous posterior (smoother) mean estimate of $x_{0:N}$. This approach is recognized as Gauss–Newton method when computing the MAP estimates [14] and it can also be extended to correspond to Levenberg–Marquardt method [15]. Here, we aim to obtain the linearized parameters according to this method which will be used in the next section to get parallel formulas.

By expanding $f_{k-1}(x_{k-1})$ and $h_k(x_k)$ in the first-order Taylor series utilizing the previous posterior means $\bar{x}_k$, the parameters of (6) are:

$$\begin{aligned} F_{k-1} &= \nabla f(\bar{x}_{k-1}), \\ c_k &= f(\bar{x}_{k-1}) - F_{k-1}\bar{x}_{k-1}, \\ H_k &= \nabla h(\bar{x}_k), \\ d_k &= h(\bar{x}_k) - H_k\bar{x}_k, \end{aligned} \tag{10}$$

where ∇f and ∇h are the Jacobians of f and h, respectively. Please note that in this paper computation of parameters in (7) and (9), and (10) is performed offline, which means that we have all measurements as well as the results of previous trajectory, that is, $\bar{x}_{1:n}$ and $\bar{P}_{1:n}$ for all n data points.

Having obtained the linearized parameters, the remaining task is to find the parallel formulas which will be discussed in the next section.

4. THE PROPOSED METHOD

Probability densities for the model of form (4) with linearization parameters of form (6) can be formulated as follows:

$$\begin{aligned} p(x_k \mid x_{k-1}) &\approx N(x_k; F_{k-1}x_{k-1} + c_{k-1}, Q'_{k-1}), \\ p(y_k \mid x_k) &\approx N(y_k; H_kx_k + d_k, R'_k), \end{aligned} \tag{11}$$

where $Q'_{k-1} = Q_{k-1} + \Lambda_{k-1}$ and $R'_k = R_k + \Omega_k$. The goal here is to obtain the parallel nonlinear Gaussian filter and smoother for the model (11). To meet this goal, similar to the method used in [12], we define a_k and binary operator $\otimes$ for our new linearized model.

Nonlinear Gaussian filtering. Aiming to specify the element a_k for obtaining parallel filtering equations according to (3), we apply Kalman filter update step to the density $p(x_k \mid x_{k-1})$ with measurement y_k. The results of the matching terms are as follows:

$$\begin{aligned} f'_k(x_k \mid x_{k-1}) &= p(x_k \mid y_k, x_{k-1}) \\ &= N(x_k; A_kx_{k-1} + b_k, C_k), \end{aligned} \tag{12}$$

where:

$$\begin{aligned} A_k &= (I_{n_x} - K_kH_k)F_{k-1}, \\ b_k &= c_{k-1} + K_k(y_k - H_kc_{k-1} - d_{k-1}), \\ C_k &= (I_{n_x} - K_kH_k)Q'_{k-1}, \\ K_k &= Q'_{k-1}H_k{}^\top S_k^{-1}, \\ S_k &= H_kQ'_{k-1}H_k^\top + R'_k. \end{aligned} \tag{13}$$

It is worth noticing that in order to find parameters of (13) at $k = 1$ and given m_0 and P_0, conventional formulations of the Kalman filter method with the linearized parameters are applied directly for prediction and update steps.

Also, using the information form of Kalman filter [20], the distribution $g'_k(x_{k-1}) = p(y_k \mid x_{k-1}) \propto N_I(x_{k-1}; \eta_k, J_k)$ can be obtained as follows:

$$\begin{aligned} J_k &= (H_kF_{k-1})^\top S_k^{-1}H_kF_{k-1} \\ \eta_k &= (H_kF_{k-1})^\top S_k^{-1}H_k(y_k - H_kc_{k-1} - d_k). \end{aligned} \tag{14}$$

Equations (13) and (14) provide the parameters of element $a_k = (A_k, b_k, C_k, \eta_k, J_k)$ in the filtering step, and they can be computed in parallel. Also, given a_i and a_j with the mentioned parameters, the binary associative operator $a_i \otimes a_j =$

a_{ij} can then be calculated with the following parameterization [12, lemma 8]:

$$\begin{aligned} A_{ij} &= A_j(I_{n_x} + C_i J_j)^{-1} A_i, \\ b_{ij} &= A_j(I_{n_x} + C_i J_j)^{-1}(b_i + C_i \eta_j) + b_j, \\ C_{ij} &= A_j(I_{n_x} + C_i J_j)^{-1} C_i A_j^\top + C_j, \\ \eta_{ij} &= A_i^\top (I_{n_x} + J_j C_i)^{-1}(\eta_j - J_j b_i) + \eta_i, \\ J_{ij} &= A_i^\top (I_{n_x} + J_j C_i)^{-1} J_j A_i + J_i. \end{aligned} \tag{15}$$

The proof for Equations (15) can be found in [12].

Nonlinear Gaussian smoothing. Assume that the filtering means x_k^* and covariance matrices P_k^* for the model (11) have been acquired as described above. We now get the following parameters for the smoothing step:

$$p(x_k \mid y_{1:k}, x_{k+1}) = N(x_k; E_k x_{k+1} + g_k, L_k) \tag{16}$$

for $k < n$:

$$\begin{aligned} E_k &= P_k F_k^\top (F_k P_k^* F_k^\top + Q'_{k-1})^{-1}, \\ g_k &= x_k^* - E_k(F_k x_k^* + c_k), \\ L_k &= P_k^* - E_k F_k P_k^*, \end{aligned} \tag{17}$$

and for $k = n$:

$$\begin{aligned} E_n &= 0, \\ g_n &= x_n^*, \\ L_n &= P_n^*. \end{aligned} \tag{18}$$

In the smoothing step, the parameters $a_k = (E_k, g_k, L_k)$ can be calculated in parallel. Now, given two elements a_i and a_j, the binary associative operator defined by $a_i \otimes a_j = a_{ij}$ can be parametrized as follows [12, lemma 10]:

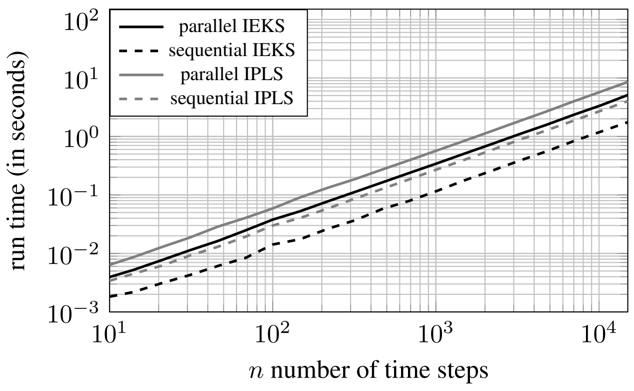
$$\begin{aligned} E_{ij} &= E_i E_j, \\ g_{ij} &= E_i g_j + g_i, \\ L_{ij} &= E_i L_j E_i^\top + L_i. \end{aligned} \tag{19}$$

5. EXPERIMENTAL RESULTS

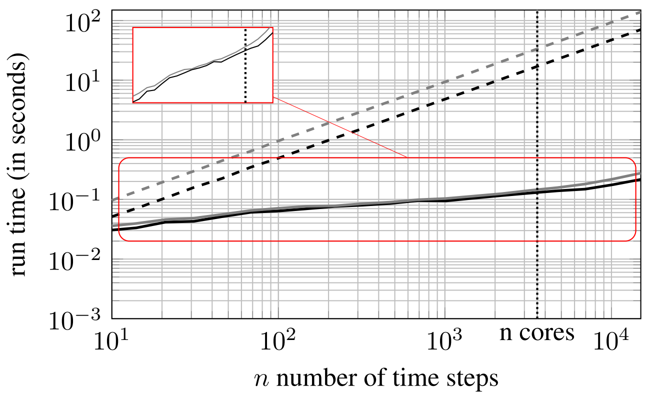
In this section, we evaluate the performance of the proposed methods on a simulated coordinated turn model with a bearings only measurement model [21] which was also used in [15]. To this end, we compare the effective average run time of the parallel versions of the extended (IEKS) and cubature integration [5] based sigma-point iterated smoothers (IPLS) with $M = 10$ iterations, as described in Section 4, with their sequential counterparts both on a CPU (Intel® Xeon® running at 2.30GHz) and on a GPU (Nvidia® Tesla® P100 PCIe 16 GB with 3584 cores). For our experiments we leverage the JAX framework [22] which implements the Blelloch parallel-scan algorithm [18] natively[1].

[1]The code can be found here: `https://github.com/EEA-sensors/parallel-non-linear-gaussian-smoothers`

In Figures 1a and 1b we observe that while the total computational cost of the parallel implementation of the iterated smoothers is higher than that of their sequential counterparts (Figure 1a), the parallelization properties of our proposed algorithms prove beneficial on a distributed environment such as a GPU (Figure 1b). Moreover, as outlined by the medallion in Figure 1b, our experiments indeed exhibit the theoretical logarithmic span complexity - derived in [12] for a linear Gaussian state space model - up to the parallelization capabilities of our GPU (3584 cores).



(a) CPU run time



(b) GPU run time

Fig. 1: Run time comparison of the parallel and sequential versions of the IEKS and IPLS on CPU (a) and GPU (b)

6. CONCLUSION

In this paper, parallel formulations for two kinds of nonlinear smoothers, namely, iterated sigma-point-based smoothers and iterated extended Kalman smoothers, have been presented. The proposed algorithms have the capability of diminishing the span-complexity from linear to logarithmic. Furthermore, the experimental results, which were conducted on a GPU, showed the benefits of the proposed methods over classical sequential methods.

7. REFERENCES

[1] T. Rauber and G. Rünger, *Parallel Programming: For multicore and cluster systems*, Springer, 2013.

[2] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[3] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al., "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 1–12.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and S. Clifford, *Introduction to Algorithms*, MIT Press, 2009.

[5] S. Särkkä, *Bayesian Filtering and Smoothing*, Cambridge University Press, 2013.

[6] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, Wiley, 2001.

[7] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, 1970.

[8] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression," in *Robotics: Science and Systems*, 2014, vol. 10.

[9] A. Grigorievskiy, N. Lawrence, and S. Särkkä, "Parallelizable sparse inverse formulation Gaussian processes (SpInGP)," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1–6.

[10] H. Ghorbanidehno, A. Kokkinaki, J. Lee, and E. Darve, "Recent developments in fast and scalable inverse modeling and data assimilation methods in hydrology," *Journal of Hydrology*, p. 125266, 2020.

[11] G. Evensen, "The ensemble Kalman filter: Theoretical formulation and practical implementation," *Ocean Dynamics*, vol. 53, no. 4, pp. 343–367, 2003.

[12] S. Särkkä and Á. F. García-Fernández, "Temporal parallelization of Bayesian smoothers," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 299–306, 2021.

[13] B. M. Bell and F. W. Cathey, "The iterated Kalman filter update as a Gauss-Newton method," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 294–297, 1993.

[14] B. M. Bell, "The iterated Kalman smoother as a Gauss–Newton method," *SIAM Journal on Optimization*, vol. 4, no. 3, pp. 626–636, 1994.

[15] S. Särkkä and L. Svensson, "Levenberg-Marquardt and line-search extended Kalman smoothers," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 5875–5879.

[16] Á. F. García-Fernández, L. Svensson, and S. Särkkä, "Iterated posterior linearization smoother," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 2056–2063, 2016.

[17] Á. F. García-Fernández, L. Svensson, M. R. Morelande, and S. Särkkä, "Posterior linearization filter: Principles and implementation using sigma points," *IEEE transactions on signal processing*, vol. 63, no. 20, pp. 5561–5573, 2015.

[18] G. E. Blelloch, "Scans as primitive parallel operations," *IEEE Transactions on Computers*, vol. 38, no. 11, pp. 1526–1538, 1989.

[19] I. Arasaratnam, S. Haykin, and R. J. Elliott, "Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 953–977, 2007.

[20] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, 1979.

[21] Y. Bar-Shalom and X.-R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, vol. 19, YBS, 1995.

[22] J. Bradbury, R. Frostig, P Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, and S. Wanderman-Milne, "JAX: composable transformations of Python+NumPy programs," `http://github.com/google/jax`, 2018.

Publication III

Adrien Corenflos, Zheng Zhao, and Simo Särkkä. Temporal Gaussian Process Regression in Logarithmic Time. In *Proceedings of the 2022 International Conference on Information Fusion (FUSION)*, Linköping, Sweden, Pages 1–5, July 2022.

Temporal Gaussian Process Regression in Logarithmic Time

Adrien Corenflos*, Zheng Zhao†, and Simo Särkkä*
*Department of Electrical Engineering and Automation
Aalto University, Finland
†Department of Information Technology
Uppsala University, Sweden

***Abstract*—The aim of this article is to present a novel parallelization method for temporal Gaussian process (GP) regression problems. The method allows for solving GP regression problems in logarithmic $O(\log N)$ time, where N stands for the number of observations and test points. Our approach uses the state-space representation of GPs which, in its original form, allows for linear $O(N)$ time GP regression by leveraging Kalman filtering and smoothing methods. By using a recently proposed parallelization method for Bayesian filters and smoothers, we are able to reduce the linear computational complexity of the temporal GP regression problems into logarithmic span complexity. This ensures logarithmic time complexity when parallel hardware such as a graphics processing unit (GPU) are employed. We experimentally show the computational benefits of our approach on simulated and real datasets via our open-source implementation leveraging the GPflow framework.**

***Index Terms*—Gaussian process, state space, parallelization, logarithmic time, Kalman filter and smoother**

I. Introduction

Gaussian processes (GPs) are a family of function-space priors used to solve regression and classification problems arising in machine learning [1]. In their native form their complexity scales as $O(N^3)$, where N is the number of training data points, which is problematic for large datasets. For a large class of covariance functions, the associated GP regression problem can be reformulated as a smoothing problem for a linear state-space model [2], [3]. This reduces the GP regression problem into a Kalman smoothing algorithm with linear time complexity $O(N)$. This improvement comes at the cost of a loss of precision for all covariance functions that do not have an exact state-space representation [4]. The linear complexity is optimal on single-threaded computational architectures, as processing data needs to be done sequentially. However, it is suboptimal on hardware where parallelization is possible, such as multi-core central processing units (CPUs) or, more importantly, on massively threaded architectures such as graphics processing units (GPUs). The aim of this letter is therefore to develop parallel state-space GP (PSSGP) methods which reduce the computational complexity (in the sense of parallel span complexity) of state-space GPs to logarithmic $O(\log N)$ (see Fig. 1 in experiments). To do so, we leverage the parallel Bayesian filtering and smoothing methodology presented in [5].

Over the recent years, several other approaches to parallelization of GPs have been proposed. For instance, in [6], [7] the authors consider mini-batching the dataset to form mixtures of local GP experts. This incurs a cubic cost only in the size of the batches, and achieves additional problem decomposition that could potentially be combined with our approach. More closely related to this letter are the works in [8], [9] which proposed to leverage the sparse Markovian structure of Markovian and state-space GPs (SSGPs). Specifically, they use parallel matrix computations, thereby reaching $O(\log N)$ span complexity in the dataset size in some special cases. However, the methods outlined in [8], [9] effectively require computations with large (albeit sparse) matrices, and their logarithmic span complexity is hard to guarantee for all the subproblems [8]. Orthogonally to these parallelization efforts, different approximation methods have been introduced in order to reduce the computational complexity of GPs. These include, for example, inducing points, spectral sampling, and basis function methods (see, e.g., [1], [10]–[13]).

The contribution of our paper is three-fold:

1) We combine the state-space formulation of GPs with parallel Kalman filters and smoothers [5].
2) We extend the parallel formulation to missing measurements to allow for predicting with state-space GPs.
3) We experimentally show the computational gains of our proposed methods on simulated and real datasets[1].

II. Gaussian processes in state-space form

In this section, we quickly recall results about the state-space formulation of Gaussian processes before we present their temporal parallelization formulation. Given a covariance function $C(t,t')$ and a set of observations $\{y_k \colon k = 1,\dots,N\}$, a temporal GP regression problem of the form

$$\begin{aligned} f(t) &\sim \mathrm{GP}(0, C(t,t')), \\ y_k &= f(t_k) + e_k, \qquad e_k \sim \mathcal{N}(0, \sigma_k^2), \end{aligned} \tag{1}$$

This work was supported Academy of Finland (projects 321900 and 321891) and Aalto ELEC doctoral school. Email: adrien.corenflos@aalto.fi.

[1]We implemented the method as an open-source extensible library. The code can be found at https://github.com/EEA-sensors/parallel-gps.

can be converted into a smoothing problem for an n_x-dimensional continuous-discrete state-space model

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{G}\,\mathbf{x}(t) + \mathbf{L}\,\mathbf{w}(t), \quad y_k = \mathbf{H}\,\mathbf{x}(t_k) + e_k, \tag{2}$$

where $\mathbf{x}$ is the state, y_k is the measurement, $\mathbf{w}$ is a white noise process with a constant spectral density matrix $\mathbf{Q}$, and e_k is the Gaussian measurement noise [2], [3]. The dimension n_x of the state, as well as the matrices $\mathbf{G}$, $\mathbf{L}$, $\mathbf{H}$, and $\mathbf{Q}$ in the model, depend on (and define) the covariance function at hand.

In the state-space formulation (2), the Gaussian process in Equation (1) has the representation $f(t) = \mathbf{H}\,\mathbf{x}(t)$. In the case of Matérn covariance functions, this representation is exact and available in closed form [2]. Other stationary covariance functions, such as the squared exponential, can be approximated up to an arbitrary precision by using Taylor series or Páde approximants [2], [4], [14]–[17] in the spectral domain.

The continuous-time state-space model (2) can be discretized into an equivalent discrete-time linear Gaussian state-space model (LGSSM, e.g., [18]) of the form

$$\mathbf{x}_k = \mathbf{F}_{k-1}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1} \quad y_k = \mathbf{H}\,\mathbf{x}_k + e_k, \tag{3}$$

where $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$. Then, the GP regression problem can be solved by applying Kalman filtering and smoothing algorithms on model (3) in $O(N)$ time [2].

In the rest of the article, we show how the sequential Kalman filters and smoothers used in SSGP can be replaced by their parallel versions [5]. This reduces the computational complexity of SSGP regression to $O(\log N)$. Additionally, by combining these with automatic differentiation softwares (e.g., TensorFlow [19]), we also show how this parallelization benefits GP parameter learning.

III. Handling Missing Observations in Parallel Kalman Filter

In [5], the authors introduce an equivalent formulation of Kalman filters and smoothers in terms of an associative operator. This enables them to leverage distributed implementations of scan (prefix-sum) algorithms, such as [20] and [21], in order to reduce the time complexity of Kalman filtering and smoothing down to $O(\log N)$. However, this formulation does not take into account missing observations. This prevents its application for inference in state-space GP models, where test points are treated as missing data [3].

The method introduced in [5] consists in writing the filtering step in terms of an associative operator of a sequence of five elements $(\mathbf{A}_k, \mathbf{b}_k, \mathbf{C}_k, \boldsymbol{\eta}_k, \mathbf{J}_k)$, which are first initialized in parallel and then combined using parallel associative scan [20], [21]. At the initialization step of the original algorithm, these elements need to be computed so as to correspond to the following quantities:

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, y_k) = \mathcal{N}(\mathbf{x}_t \mid \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{b}_k, \mathbf{C}_k), \tag{4}$$
$$p(y_k \mid \mathbf{x}_{k-1}) = \mathcal{N}_I(\mathbf{x}_{k-1} \mid \boldsymbol{\eta}_k, \mathbf{J}_k), \tag{5}$$

where $\mathcal{N}_I$ denotes the information form of the Gaussian distribution. However, when no observation is available at step k, these equations do not hold directly and need to be modified.

By redoing the original derivation, it turns out that, in the case of missing measurements, the posterior density $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, y_k)$ should be replaced by the transition density $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k \mid \mathbf{F}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1})$ for $k > 1$ and $p(\mathbf{x}_1)$ for $k = 1$. Specifically, in the case of missing measurements, the initialization equations for $\mathbf{A}_k$, $\mathbf{b}_k$, $\mathbf{C}_k$, $\boldsymbol{\eta}_k$, and $\mathbf{J}_k$ can be written as $\boldsymbol{\eta}_k = \mathbf{0}$, $\mathbf{J}_k = \mathbf{0}$, for all k, and

$$\mathbf{A}_k = \mathbf{F}_{k-1}, \quad \mathbf{b}_k = \mathbf{0}, \quad \mathbf{C}_k = \mathbf{Q}_{k-1}, \tag{6}$$

for $k > 1$, while, for $k = 1$, they become

$$\mathbf{A}_1 = \mathbf{0}, \quad \mathbf{b}_1 = \mathbf{0}, \quad \mathbf{C}_1 = \mathbf{P}_\infty. \tag{7}$$

When the quantities $\mathbf{A}_k$, $\mathbf{b}_k$, $\mathbf{C}_k$, $\boldsymbol{\eta}_k$, and $\mathbf{J}_k$ have been initialized for time steps with and without observations, they can be combined using parallel scan, with the associative operator $\otimes$ defined in the same way as in [5]:

$$\begin{aligned}
\mathbf{A}_{ij} &= \mathbf{A}_j \left(\mathbf{I}_{n_x} + \mathbf{C}_i \mathbf{J}_j\right)^{-1} \mathbf{A}_i, \\
\mathbf{b}_{ij} &= \mathbf{A}_j \left(\mathbf{I}_{n_x} + \mathbf{C}_i \mathbf{J}_j\right)^{-1} \left(\mathbf{b}_i + \mathbf{C}_i \boldsymbol{\eta}_j\right) + \mathbf{b}_j, \\
\mathbf{C}_{ij} &= \mathbf{A}_j \left(\mathbf{I}_{n_x} + \mathbf{C}_i \mathbf{J}_j\right)^{-1} \mathbf{C}_i \mathbf{A}_j^\top + \mathbf{C}_j, \\
\boldsymbol{\eta}_{ij} &= \mathbf{A}_i^\top \left(\mathbf{I}_{n_x} + \mathbf{J}_j \mathbf{C}_i\right)^{-1} \left(\boldsymbol{\eta}_j - \mathbf{J}_j \mathbf{b}_i\right) + \boldsymbol{\eta}_i, \\
\mathbf{J}_{ij} &= \mathbf{A}_i^\top \left(\mathbf{I}_{n_x} + \mathbf{J}_j \mathbf{C}_i\right)^{-1} \mathbf{J}_j \mathbf{A}_i + \mathbf{J}_i.
\end{aligned}$$

Then, running a parallel scan algorithm on the elements above with the operator $\otimes$ produces a sequence of "prefix-sum" elements $\{(\mathbf{A}_k^*, \mathbf{b}_k^*, \mathbf{C}_k^*, \boldsymbol{\eta}_k^*, \mathbf{J}_k^*) \colon k = 1, \dots, N\}$. Finally, the terms $\overline{\mathbf{x}}_k \triangleq \mathbf{b}_k^*$ and $\mathbf{P}_k \triangleq \mathbf{C}_k^*$ will correspond to the filtering mean and covariance at time step k, respectively.

Proposition 1 (Equivalence of sequential and parallel Kalman filters)**.** *For any $k = 1, 2, \dots, N + M$, the Kalman filter means and covariances are given by $\overline{\mathbf{x}}_k = \mathbf{b}_k^*$ and $\mathbf{P}_k = \mathbf{C}_k^*$, respectively.*

Proof. The detailed proof is omitted due to space limitations, but the result follows by explicitly writing down the forward recursion for the elements $(\mathbf{A}_k^*, \mathbf{b}_k^*, \mathbf{C}_k^*, \boldsymbol{\eta}_k^*, \mathbf{J}_k^*)$, and checking that equations for $\mathbf{b}_k^*$ and $\mathbf{P}_k = \mathbf{C}_k^*$ coincide with the Kalman filter equations. □

On the other hand, the parallel smoothing algorithm needs no modifications with respect to [5] in order to handle missing observations, as it only relies on the result of the filtering algorithm and not directly on the observations.

IV. Temporal parallelization of Gaussian Processes

An immediate consequence of the equivalence of the parallel and sequential Kalman filters and smoothers [5] is the fact that the parallel and sequential versions of SSGP are equivalent too. In this section, we provide the details of the steps needed to create the linear Gaussian state-space model (LGSSM) representation of SSGPs. The resulting end-to-end algorithm is automatically differentiable and has a total span complexity of $O(\log N)$, from training to inference.

A. Computation of the steady-state covariance

To represent a stationary GP, one must start the state-space model SDE from the stationary initial state $\mathbf{x}(t_0) \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_\infty)$ given by the Lyapunov equation [2]. The complexity of this step is independent of the number of time steps and does not need time-parallelization. There exist a number of iterative methods for solving this kind of algebraic equations [22]. However, in order to make automatic differentiation efficient, in this work, we use the closed-form vectorization solution given in [23] (p. 229). This relies on matrix algebra, and does not need any explicit looping. This solution is feasible because we only need to numerically solve the Lyapunov equation for small state dimensions. Furthermore, as this solution only involves matrix inversions and multiplications, it is easily parallelizable on GPU architectures.

B. Balancing of the state space model

In practice, the state-space model (3) obtained via discretization is often numerically unstable due to the transition matrix having a poor conditioning number. This in turn results in inaccuracies in computing both the GP predictions and the marginal log-likelihood of the observations. To alleviate this issue, we need to resort to balancing algorithms [24] in order to obtain a transition matrix $\mathbf{F}$ which has rows and columns that have approximately equal norms, thereby obtaining a more stable state-space model. Formally, for any diagonal matrix $\mathbf{D} \in \mathbb{R}^{n_x \times n_x}$, the continuous-discrete model

$$\begin{aligned} \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} &= \mathbf{D}^{-1}\,\mathbf{G}\,\mathbf{D}\,\mathbf{z}(t) + \mathbf{D}^{-1}\,\mathbf{L}\,\mathbf{w}(t), \\ y_k &= \mathbf{H}\,\mathbf{D}\,\mathbf{z}_k + e_k, \end{aligned} \tag{8}$$

with its initial condition given by $\mathbf{z}(t_0) \sim \mathcal{N}(\mathbf{0}, \mathbf{D}^{-1}\,\mathbf{P}_\infty \mathbf{D})$, is equivalent to the state-space model (2) started at $\mathbf{x}(t_0) \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_\infty)$, in the sense that for all $t \geq t_0$ we have $f(t) = \mathbf{H}\,\mathbf{D}\,\mathbf{z}(t)$.

In particular, this means that the gradient of the log-likelihood $\log p(y_1, \dots, y_N \mid \boldsymbol{\theta})$ with respect to the parameter $\boldsymbol{\theta}$ is left unchanged by the choice of the scaling matrix $\mathbf{D}$. This property allows us to condition our state-space representation of the original GP using a scaling matrix $\mathbf{D}$ computed with the iterative methods described in [24]. It also allows us to compute the gradient of the log-likelihood of the observations with respect to the GP parameters as if $\mathbf{D}$ did not depend on $\mathbf{F}$ and, therefore, on the parameter $\boldsymbol{\theta}$. This is crucial to obtain a stable gradient by avoiding to unroll the gradient through the iteration necessary to compute $\mathbf{D}$.

C. Converting GPs into discrete-time state space

In order to use the parallel formulation of Kalman filters and smoothers in Section III, we need to first form the continuous state-space model representation from the initial Gaussian process definition. This operation is independent of the number of measurements and, therefore, has a complexity of $O(1)$. When it has been formed, we then need to transform it into a discrete-time LGSSM as given by Equation (3). In practice, the discretization can be implemented using, for example, matrix fractions or various other methods [18], [25]. These operations are fully parallelizable across the time dimension and, therefore, have a span complexity of $O(1)$ when run on a parallel architecture.

It is worth noting that, in the parameters learning phase, the discretization needs only to happen for the training data points. However, when predicting, it is necessary to insert the M requested prediction times at the right location in the training data so as to be able to run the Kalman filter and smoother routines. When done naively, this operation has complexity $O(M + N)$. However, it can be done in parallel with span complexity $O(\log(\min(M, N)))$ by using merging operation [26]. In addition, for some GP models, such as Matérn GPs the discretization can be done offline, as it admits closed-form solutions [18].

D. End-to-end complexity of parallelized state-space GPs

The complexity analysis of the six stages for running the parallellized state-space GPs are the following:

1) Forming the continuous state-space model has both $O(1)$ work and span complexities.
2) Discretizing the state-space model has $O(N)$ work complexity and $O(1)$ span complexity.
3) At training time, the parallel Kalman filtering operations have $O(N)$ total work complexity and $O(\log N)$ total span complexity.
4) At training time, automatic differentiation shares the same computational graph structure as the parallel Kalman filter. Therefore, it has the same work and span complexities: $O(N)$ and $O(\log N)$, respectively.
5) At prediction time, merging the training and test data has work complexity $O(N + M)$ and span complexity $O(\log(\min(M, N)))$.
6) At prediction time, the parallel Kalman filtering and smoothing operations have $O(N + M)$ total work complexity and $O(\log(N + M))$ total span complexity.

Putting together the above we can conclude that the total work and span complexities of doing end-to-end inference to prediction in parallelized state-space GPs are $O(N + M)$ and $O(\log(N + M))$, respectively. The memory complexity is similar, although higher, than that of the sequential algorithm, whereby we need to store all the parameters $(\mathbf{A}_k, \mathbf{b}_k, \mathbf{C}_k, \boldsymbol{\eta}_k, \mathbf{J}_k)$ at each time step, resulting in $O(N n_x^2)$ memory complexity.

V. Experiments

In this section, we show the benefits of our approach for inference and prediction on simulated and real datasets. Because GPUs are inherently massively parallelized architectures, they are not optimized for sequential data processing, and have a lower clock speed than cost-comparable CPUs. This makes running the standard state-space GPs on a GPU less attractive than running them on a CPU, contrarily to standard GPs which can leverage GPU-enabled linear algebra routines. In order to offer a fair comparison between our proposed methodology, the standard GPs, and the standard state-space GPs, we have therefore chosen to run the sequential implementation of state-space GP on a CPU while we run the standard GP and our proposed parallel state-space GP on a GPU. We verified empirically that running the standard state-space GP on the same GPU architecture resulted in a tremendous performance loss for it ($\sim 100\times$ slower), justifying running it on a CPU for benchmarking. All the results were obtained using an AMD® Ryzen Threadripper 3960X CPU with 128 GB DDR4 RAM, and an Nvidia® GeForce RTX 3090 GPU with 24GB memory.

A. Simulation model

We first study the behavior of our proposed methodology on a simple noisy sinusoidal model given by

$$\begin{aligned} f(t) &= \sin(\pi\, t) + \sin(2\,\pi\, t) + \sin(3\,\pi\, t), \\ y_k &= f(t_k) + r_k, \end{aligned} \tag{9}$$

with observations and prediction times being equally spaced on $(0, 4)$. By increasing the number of training points we can measure the empirical time complexity (in terms of wall clock) of our proposed parallel state-space GP (PSSGP) method compared to the standard GP and state-space GP (SSGP).

We have taken the covariance function to be the squared exponential (approximated to the 6th order for the state-space GPs), corresponding to $n_x = 6$. The training dataset size ranges from 2^{12} to 2^{15} points, while the test dataset contains 10 000 points. As it can be inferred from Figure 1, our proposed method consistently outperforms standard GP and SSGP across the chosen range of dataset sizes with standard GP eventually running out of memory for larger datasets.

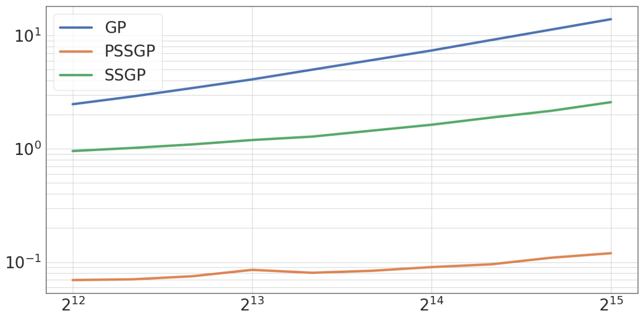


Figure 1. Average run time in seconds required to predict $M = 10\,000$ test points for noisy sinusoidal data with RBF covariance function for standard GP, SSGP, and PSSGP (ours). The number of training data points N is given on the x-axis.

B. Sunspots dataset

In this section, we compare regression and parameter learning via likelihood maximization using L-BFGS [27], [28] on the monthly sunspot activity dataset provided by World Data Center SILSO, Royal Observatory of Belgium, Brussels[2]. We learn the GP parameters on the whole dataset which contains $N = 3\,200$ points. Then, we interpolate the data on every single day from 1749-01-31 to 2018-07-31. This results in 96 000 prediction points.

[2]The data is available at http://www.sidc.be/silso/datafiles.

Table I
RUNNING TIME OF LEARNING THE GP PARAMETERS ON THE SUNSPOT DATASET RELATIVE TO PSSGP. PSSGP TOOK 39, 46, AND 48 MILLISECONDS PER FUNCTION/GRADIENT EVALUATION WHEN $N = 1\,200, 2\,200$, AND $3\,200$, RESPECTIVELY.

N	GP	SSGP	PSSGP
1200	1.03	12.08	**1**
2200	3.82	25.7	**1**
3200	10.1	43.86	**1**

The running times of the different algorithms are shown in Table I. PSSGP is respectively 10 and 43 times faster than GP and SSGP, when $N = 3\,200$. Interpolating daily took 0.14s for PSSGP, while SSGP on our CPU and standard GP on our GPU were respectively 23 and 33 times slower.

C. CO2 concentration dataset

In order to understand the impact of the dimensionality of the SDE, we finally consider the Mauna Loa carbon dioxide concentration dataset[3]. Specifically, to model the periodic pattern of the data, we use the composite covariance function $C_{\text{co2}}(t - t') = C_{\text{Per.}}(t - t')\, C_{\text{Mat.}}(t - t') + C_{\text{Mat.}}(t - t')$ and convert its periodic component to its state-space form using its Taylor expansion [14][4] up to order 1, 2, and 3. This results in SDEs of dimensions $n_x = 10$, 14, and 18, respectively. Then we perform HMC sampling (see, e.g., [29]) on the GP regression model parameters. We selected monthly and weekly data from year 1974 to 2021, which contains 3192 training points.

Table II
RELATIVE TIME OF SAMPLING FROM THE PARAMETERS POSTERIOR DISTRIBUTION USING HMC WITH 10 LEAPFROG STEPS ON THE CO2 DATASET. THE GP TOOK 3 SECONDS PER SAMPLE.

Order	GP	SSGP	PSSGP
1	1	4.5	**0.55**
2	**1**	5.73	1.36
3	**1**	6.9	2.55

As it can be inferred from Table II, while PSSGP is still competitive compared to SSGP for high dimensional SDE representations, its complexity increases with the dimension to the point where it eventually does not outperform the standard GP anymore.

VI. DISCUSSION

We have presented a sublinear algorithm for learning and inference in state space Gaussian processes, leveraging and extending the parallel Kalman filter and smoother introduced in [5]. This has allowed us to dramatically reduce the training time for regression problems on large datasets as evidenced

[3]The data is available at https://www.esrl.noaa.gov/gmd/ccgg/trends/.

[4]Our choice of covariance function is slightly different from the one suggested in [14] where the authors also add an RBF covariance function term to C_{co2}. However, we did not see any improvement from adding this supplementary degree of freedom and therefore left it out.

by our experiments on synthetic and real data. However, our final experiment has also revealed that PSSGP scales worse than SSGP as the dimension of the state in the state-space representation of the GP regression problem increases. This is due to the necessity of solving a system of n_x equations in the current parallel form of Kalman filtering, whereas the sequential form only requires to solve $n_y < n_x$ ones. Finally, recent works [30] show that similar parallelization techniques can also be used for non-linear state-space models, which could then make it possible to exploit the present methodology for Gaussian process classification and deep state-space Gaussian processes [31].

References

[1] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[2] J. Hartikainen and S. Särkkä, "Kalman filtering and smoothing solutions to temporal Gaussian process regression models," in *Proceedings of the 2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2010, pp. 379–384.

[3] S. Särkkä, A. Solin, and J. Hartikainen, "Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 51–61, 2013.

[4] S. Särkkä and R. Piché, "On convergence and accuracy of state-space approximations of squared exponential covariance functions," in *Proceedings of the 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2014, pp. 1–6.

[5] S. Särkkä and Á. F. García-Fernández, "Temporal parallelization of Bayesian smoothers," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 299–306, 2021.

[6] H. Liu, J. Cai, Y. Wang, and Y. S. Ong, "Generalized robust Bayesian committee machine for large-scale Gaussian process regression," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 3131–3140.

[7] M. M. Zhang and S. A. Williamson, "Embarrassingly parallel inference for Gaussian processes," *Journal of Machine Learning Research*, vol. 20, no. 169, pp. 1–26, 2019.

[8] A. Grigorievskiy, N. Lawrence, and S. Särkkä, "Parallelizable sparse inverse formulation Gaussian processes (SpInGP)," in *Proceedings of the 2017 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1–6.

[9] F. Lindgren, H. Rue, and J. Lindström, "An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 4, pp. 423–498, 2011.

[10] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[11] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum Gaussian process regression," *Journal of Machine Learning Research*, vol. 11, pp. 1865–1881, 2010.

[12] J. Hensman, N. Durrande, and A. Solin, "Variational Fourier features for Gaussian processes," *Journal of Machine Learning Research*, vol. 8, no. 151, pp. 1–52, 2017.

[13] A. Solin and S. Särkkä, "Hilbert space methods for reduced-rank Gaussian process regression," *Statistics and Computing*, vol. 30, no. 2, pp. 419–446, 2020.

[14] ——, "Explicit link between periodic covariance functions and state space models," in *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 33, 2014, pp. 904–912.

[15] ——, "Gaussian quadratures for state space approximation of scale mixtures of squared exponential covariance functions," in *Proceedings of the 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2014, pp. 1–6.

[16] T. Karvonen and S. Särkkä, "Approximate state-space Gaussian processes via spectral transformation," in *Proceedings of the 2016 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016, pp. 1–6.

[17] F. Tronarp, T. Karvonen, and S. Särkkä, "Mixture representation of the Matérn class with applications in state space approximations and Bayesian quadrature," in *Proceedings of the 2018 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2018, pp. 1–6.

[18] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[20] G. E. Blelloch, "Scans as primitive parallel operations," *IEEE Transactions on Computers*, vol. 38, no. 11, pp. 1526–1538, 1989.

[21] ——, "Prefix sums and their applications," School of Computer Science, Carnegie Mellon University, Tech. Rep. CMU-CS-90-190, 1990.

[22] A. S. Hodel, B. Tenison, and K. R. Poolla, "Numerical solution of the Lyapunov equation by approximate power iteration," *Linear Algebra and its Applications*, vol. 236, pp. 205–230, 1996.

[23] W. L. Brogan, *Modern Control Theory*, 3rd ed. Pearson, 2011.

[24] E. E. Osborne, "On pre-conditioning of matrices," *Journal of the ACM*, vol. 7, no. 4, pp. 338–345, 1960.

[25] P. Axelsson and F. Gustafsson, "Discrete-time solutions to the continuous-time differential Lyapunov equation with applications to Kalman filtering," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 632–643, 2014.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT press, 2009.

[27] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, p. 550–560, 1997.

[28] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.

[29] R. M. Neal, "MCMC using Hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011, ch. 5.

[30] F. Yaghoobi, A. Corenflos, S. Hassan, and S. Särkkä, "Parallel iterated extended and sigma-point Kalman smoothers," in *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5350–5354.

[31] Z. Zhao, M. Emzir, and S. Särkkä, "Deep state-space Gaussian processes," *Statistics and Computing*, vol. 31, no. 6, p. 75, 2021.

Publication IV

Adrien Corenflos, Nicolas Chopin, and Simo Särkkä. De-Sequentialized Monte Carlo: a parallel-in-time particle smoother. *Journal of Machine Learning Research*, Volume 23, Number 283, Pages 1–39, August 2022.

Journal of Machine Learning Research 23 (2022) 1-39 Submitted 2/22; Revised 7/22; Published 8/22

De-Sequentialized Monte Carlo: a parallel-in-time particle smoother

Adrien Corenflos ADRIEN.CORENFLOS@AALTO.FI
Department of Electrical Engineering and Automation, Aalto University

Nicolas Chopin NICOLAS.CHOPIN@ENSAE.FR
ENSAE, Institut Polytechnique de Paris

Simo Särkkä SIMO.SARKKA@AALTO.FI
Department of Electrical Engineering and Automation, Aalto University

Editor: Anthony Lee

Abstract

Particle smoothers are SMC (Sequential Monte Carlo) algorithms designed to approximate the joint distribution of the states given observations from a state-space model. We propose dSMC (de-Sequentialized Monte Carlo), a new particle smoother that is able to process T observations in $\mathcal{O}(\log_2 T)$ time on parallel architectures. This compares favorably with standard particle smoothers, the complexity of which is linear in T. We derive $\mathcal{L}_p$ convergence results for dSMC, with an explicit upper bound, polynomial in T. We then discuss how to reduce the variance of the smoothing estimates computed by dSMC by (i) designing good proposal distributions for sampling the particles at the initialization of the algorithm, as well as by (ii) using lazy resampling to increase the number of particles used in dSMC. Finally, we design a particle Gibbs sampler based on dSMC, which is able to perform parameter inference in a state-space model at a $\mathcal{O}(\log_2 T)$ cost on parallel hardware.

Keywords: Sequential Monte Carlo; Parallel methods; Particle smoothing; Particle Gibbs

1. Introduction

State-space models (SSM), or hidden Markov models, are a class of statistical models that comprise unobserved (latent) Markovian states $X_t \in \mathcal{X}$ for $t \in \{0, 1, \dots, T\}$, and conditionally independent observations $Y_t \in \mathcal{Y}$. The models can be written in the form

$$\begin{aligned} X_t \mid x_{t-1} &\sim P_t(\mathrm{d}x_t \mid x_{t-1}), \\ Y_t \mid x_t &\sim P_t(\mathrm{d}y_t \mid x_t), \end{aligned} \tag{1}$$

with $X_0 \sim \mathbb{P}_0(\mathrm{d}x_0)$, where $P_t(\mathrm{d}x_t \mid x_{t-1})$ is the transition kernel of the Markov sequence X_t modeling the dynamics of the system, $P_t(\mathrm{d}y_t \mid x_t)$ is the conditional distribution of measurements Y_t, and $\mathbb{P}_0(\mathrm{d}x_0)$ is the prior distribution of the initial state X_0. For simplicity, we assume that there exist h_t such that $P_t(\mathrm{d}y_t \mid x_t) = h_t(y_t \mid x_t)\,\mathrm{d}y_t$, where $\mathrm{d}y_t$ refers to a dominating measure over $\mathcal{Y}$, for example, the Lebesgue measure if $\mathcal{Y} = \mathbb{R}^{d_y}$.

In this paper, we consider the state-estimation problem which refers to the problem of inferring the states X_t from the measurements Y_t. In particular, we concentrate on the smoothing problem, where the aim is to infer the distribution of the whole trajectory of states $X_{0:T}$ given the whole set of measurements $Y_{0:T}$. A typical application of state estimation consists in target tracking, where the state X_t models the position (and possibly other physical quantities such as the speed) of a moving target, and the observation Y_t corresponds to some noisy and partial or indirect measurement of X_t (Jazwinski, 1970; Bar-Shalom et al., 2001). Additionally, we consider the parameter-estimation problem of inferring the unknown parameters appearing in the model. In addition to target tracking, state and/or parameter estimation problems also arise in various other applications such as in biomedicine, epidemiology, finance, audio signal analysis, and imaging. For an in-depth review of state-space models and their applications, see the books of Särkkä (2013) and Chopin and Papaspiliopoulos (2020).

In the signal processing setting, the solutions to the smoothing problem are focused on computing the marginal conditional distributions of X_t for $t = 0, \dots, T$ given all the measurements $Y_{0:T}$. However, in the context of Monte Carlo methods – which we also concentrate on here – it is more natural to directly consider the joint distribution of all the states and measurements which can be written as

$$\mathbb{P}(\mathrm{d}x_{0:T}\,, \mathrm{d}y_{0:T}) \coloneqq \mathbb{P}_0(\mathrm{d}x_0) \left\{ \prod_{t=0}^{T} h_t(y_t \mid x_t)\,\mathrm{d}y_t \right\} \left\{ \prod_{t=1}^{T} P_t(\mathrm{d}x_t \mid x_{t-1}) \right\}. \tag{2}$$

In this notation, smoothing consists in representing the posterior distribution of the states conditionally on the observations, $\mathbb{Q}_T(\mathrm{d}x_{0:T}) \coloneqq \mathbb{P}(\mathrm{d}x_{0:T} \mid y_{0:T})$, and in particular being able to approximate expectations such as $\mathbb{Q}_T(\varphi) \coloneqq \mathbb{E}_{\mathbb{Q}_T}[\varphi(X_{0:T})]$ for some function φ of interest. When $\mathbb{P}_0(\mathrm{d}x_0) = \mathbb{P}_0(\mathrm{d}x_0 \mid \theta)$, $P_t(\mathrm{d}x_t \mid x_{t-1}) = P_t(\mathrm{d}x_t \mid x_{t-1}, \theta)$, and $h_t(y_t \mid x_t) = h_t(y_t \mid x_t, \theta)$ depend on a parameter θ, parameter estimation consists in computing estimates θ either as point estimates or in the form of a posterior distribution of the parameter. Formally, if θ is given a prior distribution $p(\mathrm{d}\theta)$, one can represent its posterior distribution as

$$p(\mathrm{d}\theta \mid y_{0:T}) \propto p(\mathrm{d}\theta) p(y_{0:T} \mid \theta), \tag{3}$$

where

$$p(y_{0:T} \mid \theta) = \int_{\mathcal{X}^{T+1}} \mathbb{P}_0(\mathrm{d}x_0 \mid \theta) \left\{ \prod_{t=0}^{T} h_t(y_t \mid x_t, \theta) \right\} \left\{ \prod_{t=1}^{T} P_t(\mathrm{d}x_t \mid x_{t-1}, \theta) \right\}. \tag{4}$$

Except in the case of finite-state SSMs (e.g. Rabiner, 1989), linear Gaussian SSMs (LGSSMs) (Kalman, 1960; Rauch et al., 1965), and certain other special cases, neither the smoothing nor the parameter estimation problems admit a closed-form solution, and we need to resort to approximations. A successful class of such approximations comprise Gaussian approximation based filtering and smoothing approximations such as extended (Jazwinski, 1970), unscented (Julier et al., 2000), and cubature Kalman filters (Ito and Xiong, 2000; Arasaratnam and Haykin, 2009), as well as their corresponding smoothers (for a review, see, e.g., Särkkä, 2013). Another class of methods is sequential Monte Carlo (SMC) algorithms (see, e.g., Gordon et al., 1993; Doucet et al., 2000; Chopin and Papaspiliopoulos, 2020) such as particle filters and smoothers which are based on Monte Carlo sampling from the filtering and smoothing distributions. These algorithms can, more generally, also sample from the full distribution of Feynman–Kac models (Del Moral, 2004; Chopin and Papaspiliopoulos, 2020) given as a product of Markov kernels and potentials h_t as follows:

$$\mathbb{Q}_T(\mathrm{d}x_{0:T}) \propto \mathbb{P}_0(\mathrm{d}x_0) \left\{ \prod_{t=0}^{T} h_t(x_t) \right\} \left\{ \prod_{t=1}^{T} P_t(\mathrm{d}x_t \mid x_{t-1}) \right\}, \tag{5}$$

which recovers the case of (2) by setting $h_t(x_t) = h_t(y_t \mid x_t)$ in a slight abuse of notation.

The aforementioned finite-state methods, Gaussian approximations, and SMC methods are based on sequential forward and backward recursions which allow for computationally efficient algorithms that scale linearly in the number of time steps $\mathcal{O}(T)$. Although this computational complexity is (in a sense) optimal in classical single-core computers, it is not optimal in multi-core parallel computers which are capable of sub-linear time-complexity in terms of span-complexity (Cormen et al., 2009) – span-complexity referring here to the actual wall-clock time taken by a method when run on a parallel computer which can be less than $\mathcal{O}(T)$ even when the size of data is T. The sequential approximations for filtering and smoothing, in their standard formulation, have a linear time complexity in T even when run on a parallel computer, which is due to the inherent sequential nature of the computations.

However, it was recently shown in Särkkä and García-Fernández (2021) that Bayesian filtering and smoothing recursions (including, e.g., the Kalman filter and smoother) can be reformulated in terms of associative operators that can be time-parallelized to $\mathcal{O}(\log T)$ span-complexity by using a parallel scan algorithm. In Hassan et al. (2021), similar methods were developed for finite-state models, and Yaghoobi et al. (2021) developed Gaussian approximation based parallel methods for non-linear SSMs. These methods reduce the computational cost from linear to logarithmic in the number of time steps T on highly parallel hardware such as graphics processing units (GPUs). Unfortunately, the general formulation of Särkkä and García-Fernández (2021) is not directly applicable to SMC-based particle filters and smoothers, as propagating the associative operator appearing in Särkkä and García-Fernández (2021) is exactly what SMC offers to do in the first place. The aim of this article is to fix this shortcoming by proposing a parallel-in-time (PIT) formulation of SMC, the de-Sequentialized Monte Carlo (dSMC) method, that can be used – either as a standalone method, or in combination with Gaussian approximations – in order to perform Monte Carlo inference in general SSMs. However, instead of using an associative operator formulation as in Särkkä and García-Fernández (2021), the method uses parallel merging of blocks in a tree structure.

1.1 Related work

Temporal parallelization of general Bayesian filters and smoothers have recently been discussed in Särkkä and García-Fernández (2021), Hassan et al. (2021), and Yaghoobi et al. (2021), but only in the contexts of Gaussian approximations and finite-state models. Parallelization methods for Kalman type of (ensemble) filters via parallel matrix computations over the state dimension are presented in Lyster et al. (1997) and Evensen (2003). In the context of SMC methods, parallelization over particles has been considered in Lee et al. (2010), Rosen and Medvedev (2013), and Murray et al. (2016), however, these methods do not address the time dimension and their computational complexity is still linear in T on parallel hardware. In the context of variational inference (see, e.g. Blei et al., 2017), it was also noted in Aitchison (2019) that operations akin to sequential importance sampling could easily be written as chaining matrix multiplications, allowing to parallelize these on a GPU, both in the time and particle dimensions. Singh et al. (2017) consider blocking strategies for particle Gibbs algorithm, using the Markov property to allow the treatment of non-contiguous time blocks in parallel. Their method, however, works better for larger blocks with a significant overlap, thereby reducing its parallelization properties, and they also do not consider parallelization of particle smoothing. Orthogonally to these direction, coupled smoothing methods, introduced in Jacob et al. (2019) and further developed in Middleton et al. (2019) and Lee et al. (2020), allow to compute unbiased estimates of particle smoothers. This allows to parallelize calculation of smoothing expectations by aggregating many unbiased smoothers together.

Closest to our work is Lindsten et al. (2017) who consider the case of already formed graphical models. In fact, once the tree structure of dSMC is built, our algorithm can be seen as a direct instance of divide and conquer SMC for one dimensional lattices (Lindsten et al., 2017, Section 3.4), which propagates and merges particle samples from children nodes to a parent node. In their article, Lindsten et al. (2017) show the consistency of their algorithm in terms of convergence in probability. This was further improved by Kuntz et al. (2021b) who derived additional theoretical properties of estimates computed from divide-and-conquer SMC. These results can be applied to dSMC as well. However, our method differs from both these articles in several ways. First, Lindsten et al. (2017) do not consider *modifying* the structure of a pre-existing graphical model to be able to parallelize it. Second, the bounds for $\mathcal{L}_p$ errors we derive in this article depend explicitly (and polynomially) on T. These results are specific to dSMC as a parallel algorithm. Third, we derive a parallel-in-time particle Gibbs algorithm for dSMC which can be more generally applied to Lindsten et al. (2017). Lastly, we introduce parallel-in-time initialization of the algorithm and lazy resamplings as ways to speed up the algorithm and allow for better scalability in the number of particles used.

Finally, we note that Ding and Gandy (2018) introduced a smoothing algorithm leveraging the same binary tree. However, their method differs from ours in the following aspects. The main goal of Ding and Gandy (2018) is to reduce the variance of smoothing algorithms by computing adapted target distributions at each node of the tree. As a consequence, they do not directly address parallelization in time (our main motivation), and, in fact, do not allow for it as their algorithm requires to run a particle filter and a particle smoother *a priori*. They also do not discuss approximated LGSSM (PIT) initialization, lazy schemes, or particle Gibbs extensions.

1.2 Contributions

In Section 2, we introduce a formal divide-and-conquer formulation of the smoothing distribution for a class of Feynman–Kac models, which is then used to define dSMC. We then proceed to study the properties of dSMC and, in particular, we derive $\mathcal{L}_p$ error bounds that only scale polynomially in T for balanced tree representations of the smoothing distribution. Section 3 is concerned with introducing the conditional formulation of dSMC. This is then used to define a PIT particle Gibbs algorithm. In Section 4, we discuss how to construct adapted proposals without breaking the logarithmic scaling in T, and then show how parallel resampling methods can be used to lazily increase the number of particles used in dSMC. Finally, in Section 5, we experimentally demonstrate the statistical and computational properties of our method on a suite of examples. The article concludes with a discussion of the limitations and possible improvements of the de-Sequentialized Monte Carlo method.

2. De-Sequentialized Monte Carlo

We first introduce the core components required for building a parallel-in-time (PIT) particle smoother algorithm that we call de-Sequentialized Monte Carlo (dSMC). Our method relies on a divide-and-conquer approach, where we recursively stitch together partial smoothing distributions $\mathbb{Q}_{a:b}(\mathrm{d}x_{a:b})$ to form the final estimate. In order to do this, we first present the tree structure associated with smoothing in state-space models, and then we discuss how importance sampling-resampling can be leveraged to create joint samples from marginal ones. Finally, we describe the resulting algorithm and derive convergence bounds for it. For the sake of generality, we will consider the potential formulation $h_t(x_t)$ in (5), which possibly depends on y_t, but, by a slight abuse of language, we will still refer to $\mathbb{Q}_T$ as the smoothing distribution. To ensure that the model (5) is not degenerate, we will for simplicity assume that these potentials h_t are positive.

2.1 Tree structure

The recursive expressions for the smoothing distribution

$$\mathbb{Q}_T(\mathrm{d}x_{0:T}) = \frac{1}{L_T}\left[\mathbb{P}_0(\mathrm{d}x_0)\prod_{t=1}^{T} P_t(\mathrm{d}x_t \mid x_{t-1})\right]\prod_{t=0}^{T} h_t(x_t), \tag{6}$$

where L_T is a normalizing constant, are given by the forward Feynman–Kac recursion (see, e.g., Del Moral, 2004; Chopin and Papaspiliopoulos, 2020)

$$\mathbb{Q}_{t+1}(\mathrm{d}x_{0:t+1}) \propto \mathbb{Q}_t(\mathrm{d}x_{0:t})h_{t+1}(x_{t+1})P_t(\mathrm{d}x_{t+1} \mid x_t),$$

or the backward one

$$\mathbb{Q}_T(\mathrm{d}x_{t:T}) \propto h_{t+1}(x_{t+1})p_t(x_{t+1} \mid x_t)\,\mathrm{d}x_t\,\mathbb{Q}_T(\mathrm{d}x_{t+1:T}),$$

when $P_t(\mathrm{d}x_{t+1} \mid x_t)$ admits a density $p_t(x_{t+1} \mid x_t)$ with respect to a fixed (x_t-independent) measure $\mathrm{d}x_{t+1}$. Leveraging these recursions respectively corresponds to particle filtering and particle smoothing algorithms, and results in algorithms for sampling from $\mathbb{Q}_T$ that scale computationally in $\mathcal{O}(T)$.

In this section we instead propose a divide-and-conquer recursive construction of the smoothing density $\mathbb{Q}_T$. In order to do so, we introduce the concept of partial smoothing distributions.

Definition 1 *Let $(\nu_c(\mathrm{d}x_c))_{c=0}^{T}$ be a collection of probability measures, such that for all $c > 0$ and all $x_{c-1} \in \mathcal{X}$, $P_c(\mathrm{d}x_c \mid x_{c-1})$ is absolutely continuous with respect to $\nu_c(\mathrm{d}x_c)$. Then for any $0 \le a \le b \le T$, we can define*

$$\mathbb{Q}^{\nu}_{a:b}(\mathrm{d}x_{a:b}) \coloneqq \frac{1}{L^{\nu}_{a:b}}\left[\nu_a(\mathrm{d}x_a)\prod_{t=a+1}^{b} P_t(\mathrm{d}x_t \mid x_{t-1})\right]\prod_{t=a+1}^{b} h_t(x_t), \tag{7}$$

where $L^{\nu}_{a:b}$ is a normalizing constant (assumed to be positive), and by convention the product over an empty set is 1, so that, for any a, $\mathbb{Q}^{\nu}_{a:a}(\mathrm{d}x_a) = \nu_a(\mathrm{d}x_a)$ and $L^{\nu}_{a:a} = 1$.

Provided that ν_0 defines the filtering posterior distribution of x_0, we can then recover the original $\mathbb{Q}_T$ from $\mathbb{Q}^{\nu}_{0:T}$. This corresponds to the following proposition.

Proposition 2 *For any family $\nu_{0:T}$ given by Definition 1, and such that the initial distribution verifies $\nu_0(\mathrm{d}x_0) \propto h_0(x_0)\mathbb{P}_0(\mathrm{d}x_0)$, we have*

$$\mathbb{Q}_T(\mathrm{d}x_{0:T}) = \mathbb{Q}^{\nu}_{0:T}(\mathrm{d}x_{0:T}). \tag{8}$$

The partial smoothing distributions $(\mathbb{Q}^{\nu}_{a:b})_{0 \le a < b, \le T}$ can then be stitched together, forming a recursive structure for the smoothing operation.

Proposition 3 *For any $0 \leq a < c < b \leq T$, we have*

$$\mathbb{Q}^{\nu}_{a:b}(\mathrm{d}x_{a:b}) = \frac{L^{\nu}_{a:c-1} L^{\nu}_{c:b}}{L^{\nu}_{a:b}} \omega^{\nu}_{c}(x_{c-1}, x_c) \mathbb{Q}^{\nu}_{a:c-1}(\mathrm{d}x_{a:c-1}) \mathbb{Q}^{\nu}_{c:b}(\mathrm{d}x_{c:b}), \tag{9}$$

where ω^{ν}_{c} is defined as the following Radon–Nikodym derivative:

$$\omega^{\nu}_{c}(x_{c-1}, x_c) \coloneqq \frac{P_c(\mathrm{d}x_c \mid x_{c-1}) h_c(x_c)}{\nu_c(\mathrm{d}x_c)}. \tag{10}$$

Proof For all $0 \leq a < c < b \leq T$, we have:

$$\begin{aligned}
&\frac{L^{\nu}_{a:c-1} L^{\nu}_{c:b}}{L^{\nu}_{a:b}} \omega^{\nu}_{c}(x_{c-1}, x_c) \mathbb{Q}^{\nu}_{a:c-1}(\mathrm{d}x_{a:c-1}) \mathbb{Q}^{\nu}_{c:b}(\mathrm{d}x_{c:b}) \\
&= \frac{1}{L^{\nu}_{a:b}} \frac{P_c(\mathrm{d}x_c \mid x_{c-1}) h_c(x_c)}{\nu_c(\mathrm{d}x_c)} \\
&\times \left[\nu_a(\mathrm{d}x_a) \prod_{t=a+1}^{c-1} P_t(\mathrm{d}x_t \mid x_{t-1}) \right] \prod_{t=a+1}^{c-1} h_t(x_t) \\
&\times \left[\nu_c(\mathrm{d}x_c) \prod_{t=c+1}^{b} P_t(\mathrm{d}x_t \mid x_{t-1}) \right] \prod_{t=c+1}^{b} h_t(x_t) \\
&= \frac{1}{L^{\nu}_{a:b}} \left[\nu_a(\mathrm{d}x_a) \prod_{t=a+1}^{b} P_t(\mathrm{d}x_t \mid x_{t-1}) \right] \prod_{t=a+1}^{b} h_t(x_t) \\
&= \mathbb{Q}^{\nu}_{a:b}(\mathrm{d}x_{a:b}).
\end{aligned} \tag{11}$$

■

The recursive property exhibited by Proposition 3 allows us to construct an arbitrary tree structure on the smoothing distribution. This construction is illustrated in Figure 1. In practice, we could use any ordered binary tree structure on $\{0, 1, \dots, T\}$ to define a well-posed recursive representation of $\mathbb{Q}_T$, but, as we will see in Sections 2.2 and 2.3, balanced representations offer better statistical and computational properties.

In practice, the stitching operation described by Proposition 3 is not tractable in closed-form, and we need to resort to Monte Carlo integration instead.

2.2 Sample stitching

For notational simplicity, in this section and all subsequent ones we do not emphasize the dependency of our estimates on ν. Suppose that we have two independent Monte Carlo

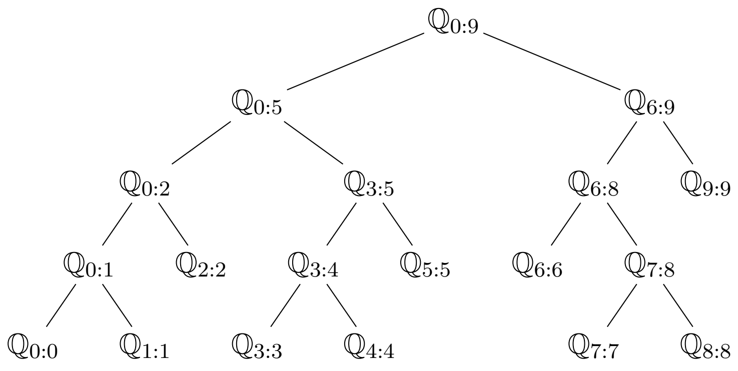


Figure 1: Example of a recursive tree structure for $\mathbb{Q}_{0:9}$.

approximations

$$\begin{aligned} \mathbb{Q}_{a:c-1} &\approx \mathbb{Q}^N_{a:c-1} \coloneqq \sum_{n=1}^N w^n_{c-1} \delta_{X^n_{a:c-1}} \\ \mathbb{Q}_{c:b} &\approx \mathbb{Q}^N_{c:b} \coloneqq \sum_{n=1}^N w^n_{c} \delta_{X^n_{c:b}}, \end{aligned} \tag{12}$$

following Lindsten et al. (2017) and Kuntz et al. (2021a), we can then form the "product-form" importance empirical density

$$\widetilde{\mathbb{Q}}^N_{a:b} \coloneqq \sum_{m,n=1}^N W^{m,n}_c \delta_{[X^m_{a:c-1}, X^n_{c:b}]}, \tag{13}$$

where

$$W^{m,n}_c = \frac{w^m_{c-1} w^n_c \omega_c(X^m_{c-1}, X^n_c)}{\sum_{i,j=1}^N w^i_{c-1} w^j_c \omega_c(X^i_{c-1}, X^j_c)}. \tag{14}$$

As described in Kuntz et al. (2021a), this estimator exhibits better statistical properties than the "naive" estimator

$$\sum_{n=1}^N \frac{w^n_{c-1} w^n_c \omega_c(X^n_{c-1}, X^n_{c-1})}{\sum_{m=1}^N w^m_{c-1} w^m_c \omega_c(X^m_{c-1}, X^m_{c-1})} \delta_{[X^n_{a:c-1}, X^n_{c:b}]}$$

when the $(X^n_{c-1})^N_{n=1}$, and $(X^n_c)^N_{n=1}$ have been sampled independently.

Moreover, the denominator of (14) directly provides us with an estimate of the normalizing constant increment, that is, if $L^N_{a:c-1}$ and $L^N_{c:b}$ are estimates of the normalizing constants of $\mathbb{Q}_{a:c-1}$ and $\mathbb{Q}_{c:b}$, respectively, then, following Proposition 3, we know that

$$\begin{aligned} L_{a:b} &= L_{a:c-1} L_{c:b} \iint \omega_c(x_{c-1}, x_c) \mathbb{Q}_{a:c-1}(\mathrm{d}x_{c-1}) \mathbb{Q}_{c:b}(\mathrm{d}x_c) \\ &\approx L^N_{a:c-1} L^N_{c:b} \sum_{i,j=1}^{N} w^i_{c-1} w^j_c \omega_c(X^i_{c-1}, X^j_{c-1}). \end{aligned} \tag{15}$$

Algorithm 1: Block combination

```
// All operations on indices $m, n$ are done in parallel
Function Combine$\left(X^{1:N}_{a:c-1}, w^{1:N}_{c-1}, L^N_{a:c-1}, X^{1:N}_{c:b}, w^{1:N}_c, L^N_{c:b}\right)$
    // In all but the initial step, the weights $w^m_{c-1}$ and $w^n_c$ are $1/N$.
    $L^N_c \leftarrow \sum_{m,n=1}^{N} \omega_c(X^m_{c-1}, X^n_c) w^m_{c-1} w^n_c$
    $W^{m,n}_c \leftarrow \omega_c(X^m_{c-1}, X^n_c) w^m_{c-1} w^n_c / L^N_c$
    // In parallel, using, e.g., multinomial or systematic resampling:
    Sample $N$ times from $\sum_{m,n} W^{m,n}_c \delta_{X^m_c}(\mathrm{d}x_c) \delta_{X^n_{c-1}}(\mathrm{d}x_{c-1})$ to get
     $(X^{l^n}_{c-1}, X^{r^n}_c)_{1 \le n \le N}$
    // The two loops below can be done in parallel:
    for $c' = a, \ldots, c-1$ in parallel do
        $\widetilde{X}^n_{c'} \leftarrow X^{l^n}_{c'}$
    for $c' = c, \ldots, b$ in parallel do
        $\widetilde{X}^n_{c'} \leftarrow X^{r^n}_{c'}$
    return $\widetilde{X}^{1:N}_{a:b}$, $L^N_{a:c-1} L^N_{c:b} L^N_c$
```

When the importance estimator in (13) has been formed, we can then resample N pairs of partial smoothing paths $(l^n, r^n)_{n=1}^N$ according to the normalized weights $W^{m,n}_c$ to obtain a stitched Monte Carlo approximation $\mathbb{Q}^N_{a:b} \coloneqq \frac{1}{N} \sum_{n=1}^{N} \delta_{[X^{l^n}_{a:c-1}, X^{r^n}_{c:b}]}$. This construction is summarized in Algorithm 1, from which we can also compute the normalizing constant increment as a by-product. On the other hand, for all $t = 0, \ldots, T$, we can define the initial self-normalized importance approximation $\mathbb{Q}^N_{t:t} \coloneqq \sum_{n=1}^{N} W^n_t \delta_{X^n_t}$, where, for all t, the X^n_t's are i.i.d. sampled from q_t, and the $W^n_t \propto \frac{\nu_t(X^n_t)}{q_t(X^n_t)}$ sum to 1.

Under this construction, we can show that the resulting Monte Carlo $\mathcal{L}_p$ error is well-behaved.

Proposition 4 *Let $p \geq 1$ be an integer, suppose that the $(l^n, r^n)_{n=1}^N$ are sampled according to a categorical distribution (i.e. multinomial resampling), that φ is a bounded measurable function, and that ω_c is bounded. If for any measurable bounded functions $\varphi_{a:c-1}$ and $\varphi_{c:b}$*

we have

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:c-1}(\varphi_{a:c-1}) - \mathbb{Q}^N_{a:c-1}(\varphi_{a:c-1})\right|^p\right]^{1/p} \leq C^p_{a:c-1}\frac{\|\varphi_{a:c-1}\|_\infty}{N^{1/2}}, \tag{16}$$

$$\mathbb{E}\left[\left|\mathbb{Q}_{c:b}(\varphi_{c:b}) - \mathbb{Q}^N_{c:b}(\varphi_{c:b})\right|^p\right]^{1/p} \leq C^p_{c:b}\frac{\|\varphi_{c:b}\|_\infty}{N^{1/2}}, \tag{17}$$

for some constants $C^p_{a:c-1}, C^p_{c:b}$ independent of N, $\varphi_{a:c-1}$, and $\varphi_{c:b}$, then,

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \mathbb{Q}^N_{a:b}(\varphi)\right|^p\right]^{1/p} \leq \left(4\min(C^p_{a:c-1}, C^p_{c:b})\|\bar{\omega}_c\|_\infty + 2^{(p+1)/p}\right)\frac{\|\varphi\|_\infty}{N^{1/2}}, \tag{18}$$

where $\bar{\omega}_c \coloneqq \omega_c / Z_c$ and $Z_c \coloneqq L_{a:b}/(L_{a:c-1}L_{c:b})$.

The proof of Proposition 4 may be found in Appendix C. Proposition 4 allows us to derive upper bounds to the total $\mathcal{L}_p$ error as a consequence.

Corollary 5 *Suppose that the ω_c/Z_c's and the $\frac{\nu_c}{q_c}$'s are uniformly bounded by some constant Ω independent of c, and let $\mathbb{Q}^N_{c:c} = \frac{1}{N}\sum_{n=1}^N W^n_c \delta_{X^n_c}$, where, for all c, the X^n_c's are i.i.d. sampled from q_c, then*

$$C^p_{0:T} = O\left((4\Omega)^D\right), \tag{19}$$

where D is the depth of the tree structure chosen for the smoothing operation (see Figure 1). In particular, if the tree is balanced, that is, if $D = \lceil \log_2 T \rceil$,

$$C^p_{0:T} = O\left(T^{2+\log\Omega}\right). \tag{20}$$

Proof The initial case comes from (Del Moral, 2004, Lemma 7.3.3) so that for all c, $C^p_{c:c} \leq 2^{(p+1)/p}$. The result then proceeds by using inequalities on the progression of arithmetico-geometric sequences (see, e.g., Riley et al., 2006, Section 4.2.3). ■

Remark 6 *The uniform bounding of the quantities*

$$\frac{\omega_c}{Z_c} = \frac{\omega_c}{\iint \omega_c(x_{c-1}, x_c)\mathbb{Q}_{a:c-1}(\mathrm{d}x_{c-1})\mathbb{Q}_{c:b}(\mathrm{d}x_c)}, \tag{21}$$

for all c, assumed in Corollary 5, is, for example, true as soon as the ω_c's are uniformly bounded below and above. This hypothesis, albeit strong, is typically assumed in proofs of the uniform convergence of particle filtering algorithms (Del Moral and Guionnet, 2001).

Following Crisan and Doucet (2000, Proof of Lemma 5), for $p > 2$, Chebyshev's inequality and Borel–Cantelli lemma also provides the following corollary.

Corollary 7 *Under the same hypotheses, $\mathbb{Q}^N_{0:T}(\varphi)$ converges almost surely to $\mathbb{Q}_{0:T}(\varphi)$.*

An interesting point to notice is that for very unbalanced trees with depth of order T, Proposition 4 recovers the usual exponential scaling in T (Andrieu et al., 2001) of the mean squared error, instead of the polynomial scaling obtained when the tree is balanced.

Algorithm 2: Smoother initialization

```
// All operations on indices $m, n$ are done in parallel
for $t = 0, \dots, T$ in parallel do
    $X_t^n \leftarrow$ sample from $q_t(\mathrm{d}x_t)$
    if $t = 0$ then
        $w_0^n \leftarrow h_0(X_0^n) \frac{\mathbb{P}_0(\mathrm{d}x_0)}{q_0(\mathrm{d}x_0)}(X_0^n)$
    else
        $w_t^n \leftarrow \frac{\nu_t(\mathrm{d}x_t)}{q_t(\mathrm{d}x_t)}(X_t^n)$
    $W_t^n \leftarrow w_t^n / \sum_{m=1}^N w_t^m$
    $L_t^N \leftarrow \frac{1}{N} \sum_{m=1}^N w_t^m$
return $X_{0:T}^{1:N}, W_{0:T}^{1:N}, L_{0:T}^N$
```

2.3 Algorithm

The Monte Carlo approximation of $\mathbb{Q}_{0:T}$ can be computed using a recursive algorithm which can be parallelized across all operations happening at each level of the tree depth. At initialization, we simply need to sample independently N times from $T+1$ proposal distributions q_t, $t = 0, \dots, T$, and then form the resulting importance sampling representation of all the distributions $\mathbb{Q}_{t:t}$, $t = 0, \dots, T$, following Definition 1 and Proposition 2. This is summarized in Algorithm 2. In order to obtain a balanced tree, we can recursively split at the midpoint of the partial smoothing interval, essentially recovering a binary tree when $T+1$ is a power or 2. This results in Algorithm 3.

The smoothing algorithm then simply consists in passing the output of Algorithm 2 to Algorithm 3. It is worth noting that while Algorithm 3 is correct, its recursive nature makes its implementation on parallel devices tedious if one wants to benefit from hardware acceleration. Moreover it does not consist in a tail recursion (see, e.g. Muchnick, 1997, Chap. 15), so that it cannot easily be transformed into a loop that would be easier to parallelize. However, the split-combine operations can be reformulated as a series of tensor reshaping operations, which is more amenable to parallelization. We provide this equivalent, albeit parallelizable, formulation of the algorithm in Appendix A.

Consider now the choice of tree partitioning given in Figure 1. The nodes correspond to the combination operation, while the edges correspond to the split happening in Algorithm 3. All the operations at a given depth can be run fully in parallel, each of them being entirely parallelizable too with respect to the particle samples, except for the resampling operation. The resampling operation requires normalizing the weights and running parallel search operations, which can be done with span complexity (Cormen et al., 2009) of $\mathcal{O}(\log N)$ on parallel architectures (using prefix-sum operations, see, e.g., Murray et al., 2016), so that each level of the tree has span complexity $\mathcal{O}(\log N)$. This results in a parallelized algorithm run time that globally scales linearly with the depth of the smoothing

Algorithm 3: Recursion

```
Function Recursion$(X^{1:N}_{a:c-1}, w^{1:N}_{a:c-1}, L^N_{a:c-1}, X^{1:N}_{c:b}, w^{1:N}_{c:b}, L^N_{c:b})$
    if $a = c - 1$ and $b = c$ then
        return Combine$(X^{1:N}_{a:a}, w^{1:N}_a, L^N_{a:a}, X^{1:N}_{b:b}, w^{1:N}_b, L^N_{b:b})$
    else if $c - 1 > a$ and $b = c$ then
        $c' \leftarrow \lfloor \frac{a+c-1}{2} \rfloor$
        $X^{1:N}_{a:c-1}, L^N_{a:c-1} \leftarrow$
          Recursion$(X^{1:N}_{a:c'-1}, w^{1:N}_{a:c'-1}, L^N_{a:c'-1}, X^{1:N}_{c':c-1}, w^{1:N}_{c':c-1}, L^N_{c':c-1})$
        return Combine$(X^{1:N}_{a:c-1}, (1/N)^N_{n=1}, L^N_{a:c-1}, X^{1:N}_{b:b}, w^{1:N}_b, L^N_{b:b})$
    else if $a = c - 1$ and $b > c$ then
        $c' \leftarrow \lfloor \frac{c+b}{2} \rfloor$
        $X^{1:N}_{c:b}, L^N_{c:b} \leftarrow$ Recursion$(X^{1:N}_{c:c'-1}, w^{1:N}_{c:c'-1}, L^N_{c:c'-1}, X^{1:N}_{c':b}, w^{1:N}_{c':b}, L^N_{c':b})$
        return Combine$(X^{1:N}_{a:a}, w^{1:N}_a, L^N_{a:a}, X^{1:N}_{c:b}, (1/N)^N_{n=1}, L^N_{c:b})$
    else
        $c' \leftarrow \lfloor \frac{a+c-1}{2} \rfloor$
        $X^{1:N}_{a:c-1}, L^N_{a:c-1} \leftarrow$
          Recursion$(X^{1:N}_{a:c'-1}, w^{1:N}_{a:c'-1}, L^N_{a:c'-1}, X^{1:N}_{c':c-1}, w^{1:N}_{c':c-1}, L^N_{c':c-1})$
        $c' \leftarrow \lfloor \frac{c+b}{2} \rfloor$
        $X^{1:N}_{c:b}, L^N_{c:b} \leftarrow$ Recursion$(X^{1:N}_{c:c'-1}, w^{1:N}_{c:c'-1}, L^N_{c:c'-1}, X^{1:N}_{c':b}, w^{1:N}_{c':b}, L^N_{c':b})$
        return Combine$(X^{1:N}_{a:c-1}, (1/N)^N_{n=1}, L^N_{a:c-1}, X^{1:N}_{c:b}, (1/N)^N_{n=1}, L^N_{c:b})$
```

tree considered, and logarithmically in the number of particles. As a consequence, we have the following proposition.

Proposition 8 *The total span complexity of dSMC is $\mathcal{O}(\log_2(T)\log(N))$.*

Remark 9 *It is worth noting that some alternative resampling methods exist that allow to parallelize the resampling operation, at the cost of biasing it, or at the cost of random execution time (Murray et al., 2016). We discuss these methods and the additional benefits they provide for dSMC in Section 4.2.*

3. Parallel-in-time particle Gibbs

We now focus on deriving a conditional formulation of dSMC (that we call c-dSMC) that we then use to build a PIT particle Gibbs algorithm. We quickly discuss its degeneracy properties, and in particular the fact that it may mix well even without the addition of a backward sampling step.

3.1 Conditional dSMC sampler

Particle Gibbs methods were introduced in Andrieu et al. (2010) in order to sample from the joint posterior $\mathbb{Q}_{0:T}(\mathrm{d}x_{0:T}\,, \mathrm{d}\theta)$ of a state-space model. It consists in successively applying two conditional sampling steps: (i) sampling θ conditionally on a given smoothing trajectory $x^*_{0:T}$, and (ii) sampling a smoothing trajectory $x'_{0:T}$ conditionally on $x^*_{0:T}$ and θ. Step (ii) needs to be understood as "conditionally to one of the trajectories sampled by the SMC algorithm being $x^*_{0:T}$".

Due to the arbitrary tree representation of the smoothing operation, it is complicated to manipulate the complete expression for the distribution of all the random variables generated during the course of the smoothing algorithm[1]. However, we can still provide a natural recursive expression that will serve as a support for understanding the behavior of the conditional distributions. In order to make notations simpler, we write $\sigma_{a:c-1}(k)$, $k = a, a+1, \dots, c-1$ for the resampling array (i.e., the array of the resampling indices) applied to node k, and we write $\sigma^n_{a:c-1}(k)$ for its n-th element (and similarly for $\sigma_{c:b}(k)$).

Remark 10 *$\sigma_{a:c-1}$ is a function of the left-right resampling indices $l^{1:N}_{a+1:c-1}, r^{1:N}_{a+1:c-1}$ generated deeper in the recursion tree, and similarly for $\sigma_{c:b}$ via the recursion*

$$\begin{aligned} \sigma^m_{a:b}(k) &= \sigma^{l^m_c}_{a:c-1}(k), \quad \textit{for all } a \le k < c, \\ \sigma^m_{a:b}(k) &= \sigma^{r^m_c}_{c:b}(k), \quad \textit{for all } c \le k \le b, \end{aligned} \tag{22}$$

and initial values

$$\sigma^m_{a:a}(a) = m, \tag{23}$$

for all $m \in \{1, 2, \dots, N\}$.

Under this notation, if the initial Monte Carlo approximation for the $\mathbb{Q}_{c:c}$'s are given by

$$\mathbb{Q}^N_{c:c} = \frac{1}{N} \sum_{n=1}^{N} W^n_c \delta_{X^n_c},$$

then for all $a \le k \le b$, $a < b$, we have

$$\mathbb{Q}^N_{a:b}(\mathrm{d}x_k) = \frac{1}{N} \sum_{n=1}^{N} \delta_{X_k^{\sigma_{a:b}(k)}}(\mathrm{d}x_k).$$

That is, $\sigma_{a:b}$ encodes the subset (with repetitions) of particles that survived from initialization down to the partial smoothing distribution approximation $\mathbb{Q}^N_{a:b}$.

1. Although this was done for instance in Lindsten et al. (2017) to prove the unbiasedness of their resulting likelihood estimate.

For $a < c < b$, let $\psi_{a:c-1}(\mathrm{d}x_{a:c-1}^{1:N}, l_{a+1:c-1}^{1:N}, r_{a+1:c-1}^{1:N})$ and $\psi_{c:b}(\mathrm{d}x_{c:b}^{1:N}, l_{c+1:b}^{1:N}, r_{c+1:b}^{1:N})$ be the full distributions of all the random variables generated by dSMC for the partial smoothing distributions $\mathbb{Q}_{a:c-1}$ and $\mathbb{Q}_{c:b}$, respectively. The full distribution of all the random variables generated to form the resampled approximation $\mathbb{Q}_{a:b}^N$ of $\mathbb{Q}_{a:b}$ is given by

$$\begin{aligned}\psi_{a:b}(\mathrm{d}x_{a:b}^{1:N}, l_{a+1:b}^{1:N}, r_{a+1:b}^{1:N}) &= \psi_{a:c-1}(\mathrm{d}x_{a:c-1}^{1:N}, l_{a+1:c-1}^{1:N}, r_{a+1:c-1}^{1:N}) \\ &\quad\times \psi_{c:b}(\mathrm{d}x_{c:b}^{1:N}, l_{c+1:b}^{1:N}, r_{c+1:b}^{1:N}) \times \left\{\prod_{n=1}^{N} W_c^{l_c^n, r_c^n}\right\}, \end{aligned} \tag{24}$$

where for all $m, n \in \{1, \dots, N\}$, we have

$$W_c^{m,n} \propto \omega_c\left(x_{c-1}^{\sigma^m_{a:c-1}(c-1)}, x_c^{\sigma^n_{c:b}(c)}\right) \tag{25}$$

so that $\sum\limits_{m,n=1}^{N} W_c^{m,n} = 1$, and the initial distributions are given by $\psi_{a:a}(\mathrm{d}x_a^{1:N}) = \prod_{n=1}^N \nu_a(\mathrm{d}x_a^n)$ (for the sake of clarity, we restrict to the case $\nu_a = q_a$).

Equation (24) further allows to define the full distribution of all the random variables generated to form the weighted approximation $\widetilde{\mathbb{Q}}_{a:b}^N$ (13) as

$$\begin{aligned}\widetilde{\psi}_{a:b}(\mathrm{d}x_{a:b}^{1:N}, l_{a+1:b}^{1:N}, r_{a+1:b}^{1:N}) &= \psi_{a:c-1}(\mathrm{d}x_{a:c-1}^{1:N}, l_{a+1:c-1}^{1:N}, r_{a+1:c-1}^{1:N}) \\ &\quad\times \psi_{c:b}(\mathrm{d}x_{c:b}^{1:N}, l_{c+1:b}^{1:N}, r_{c+1:b}^{1:N}).\end{aligned} \tag{26}$$

Similarly, the related estimate of the normalizing constant

$$L_{a:b}^N = L_{a:b}^N(x_{a:b}^{1:N}, l_{a+1:c-1}^{1:N}, l_{c+1:b}^{1:N}, r_{a+1:c-1}^{1:N}, r_{c+1:b}^{1:N}) \tag{27}$$

follows the recursion

$$L_{a:b}^N = L_{a:c-1}^N L_{c:b}^N \left\{N^{-2} \sum_{i,j=1}^{N} \omega_c\left(x_{c-1}^{\sigma^i_{a:c-1}(c-1)}, x_c^{\sigma^j_{c:b}(c)}\right)\right\}. \tag{28}$$

Putting these together allows us to characterize recursively the invariant distribution of our specific version of the particle Gibbs kernel (Andrieu et al., 2010), that we will then use in order to express the related conditional dSMC distribution

$$\pi_{a:b}(\mathrm{d}x_{a:b}^{1:N}, l_{a+1:b}^{1:N}, r_{a+1:b}^{1:N}) = \frac{L_{a:b}^N}{L_{a:b}} \widetilde{\psi}_{a:b}(\mathrm{d}x_{a:b}^{1:N}, l_{a+1:b}^{1:N}, r_{a+1:b}^{1:N}). \tag{29}$$

Following Andrieu et al. (2010), define

$$\pi_{a:b}(\mathrm{d}x_{a:b}^{1:N}, l_{a+1:b}^{1:N}, r_{a+1:b}^{1:N}, m, n) = \pi_{a:b}(\mathrm{d}x_{a:b}^{1:N}, l_{a+1:b}^{1:N}, r_{a+1:b}^{1:N}) W_c^{m,n}, \tag{30}$$

corresponding to sampling once from (13). We have

$$
\begin{aligned}
&\pi_{a:b}(\mathrm{d}x_{a:b}^{1:N}, l_{a+1:b}^{1:N}, r_{a+1:b}^{1:N}, m, n) \qquad (31)\\
&\quad = \frac{L_{a:c-1}^{N} L_{c:b}^{N} \left\{ N^{-2} \sum_{i,j=1}^{N} \omega_c \left(x_{c-1}^{\sigma^{i}_{a:c-1}(c-1)}, x_{c}^{\sigma^{j}_{c:b}(c)} \right) \right\}}{L_{a:c-1} L_{c:b} \int \omega_c(x_{c-1}, x_c) \mathbb{Q}_{a:c-1}(\mathrm{d}x_{c-1}) \mathbb{Q}_{c:b}(\mathrm{d}x_c)} W_c^{m,n}\\
&\qquad \times \psi_{a:c-1}(\mathrm{d}x_{a:c-1}^{1:N}, l_{a+1:c-1}^{1:N}, r_{a+1:c-1}^{1:N}) \times \psi_{c:b}(\mathrm{d}x_{c:b}^{1:N}, l_{c+1:b}^{1:N}, r_{c+1:b}^{1:N})\\
&\quad \propto \pi_{a:c-1}(\mathrm{d}x_{a:c-1}^{1:N}, l_{a+1:c-1}^{1:N}, r_{a+1:c-1}^{1:N}) \times \left\{ \prod_{i=1}^{N} W_{c_l}^{l_{c_l}^i, r_{c_l}^i} \right\} \qquad (32)\\
&\qquad \times \pi_{c:b}(\mathrm{d}x_{c:b}^{1:N}, l_{c+1:b}^{1:N}, r_{c+1:b}^{1:N}) \times \left\{ \prod_{i=1}^{N} W_{c_r}^{l_{c_r}^i, r_{c_r}^i} \right\}\\
&\qquad \times \omega_c \left(x_{c-1}^{\sigma^{m}_{a:c-1}(c-1)}, x_{c}^{\sigma^{n}_{c:b}(c)} \right),
\end{aligned}
$$

where we assumed that we have $a < c-1$ and $c < b$ (as we otherwise recover the base case for $\psi_{a:c-1}$ and $\psi_{c:b}$ and the recursion can be stopped), that c_l and c_r are the indices for the stitching that formed $\psi_{a:c-1}$ and $\psi_{c:b}$ in Proposition 3, respectively, and where the constant of normalization is $N^2 \int \omega_c(x_{c-1}, x_c) \mathbb{Q}_{a:c-1}(\mathrm{d}x_{c-1}) \mathbb{Q}_{c:b}(\mathrm{d}x_c)$.

This, in turn, can be rewritten to isolate a "star trajectory": developing line (32) we obtain

$$
\begin{aligned}
&\pi_{a:c-1}(\mathrm{d}x_{a:c-1}^{1:N}, l_{a+1:c-1}^{1:N}, r_{a+1:c-1}^{1:N}) \times \left\{ \prod_{i \neq m} W_{c_l}^{l_{c_l}^i, r_{c_l}^i} \right\} \times W_{c_l}^{l_{c_l}^m, r_{c_l}^m}\\
&\quad = \frac{L_{a:c_l-1}^{N} L_{c_l:c-1}^{N} \left\{ N^{-2} \sum_{i,j=1}^{N} \omega_{c_l} \left(x_{c_l-1}^{\sigma^{i}_{a:c_l-1}(c_l-1)}, x_{c_l}^{\sigma^{j}_{c_l:c-1}(c_l)} \right) \right\}}{L_{a:c_l-1} L_{c_l:c-1} \iint \omega_{c_l}(x_{c_l-1}, x_{c_l}) \mathbb{Q}_{a:c_l-1}(\mathrm{d}x_{c_l-1}) \mathbb{Q}_{c_l:c-1}(\mathrm{d}x_{c_l})}\\
&\qquad \times \psi_{a:c_l-1}(\mathrm{d}x_{a:c_l-1}^{1:N}, l_{a+1:c_l-1}^{1:N}, r_{a+1:c_l-1}^{1:N}) \times \psi_{c_l:c-1}(\mathrm{d}x_{c_l:c-1}^{1:N}, l_{c_l+1:c-1}^{1:N}, r_{c_l+1:c-1}^{1:N}),
\end{aligned}
$$

which can be further decomposed in

$$
\begin{aligned}
&\frac{L_{a:c_l-1}^{N}}{L_{a:c_l-1}} \psi_{a:c_l-1}(\mathrm{d}x_{a:c_l-1}^{1:N}, l_{a+1:c_l-1}^{1:N}, r_{a+1:c_l-1}^{1:N})\\
&\qquad \times \frac{L_{c_l:c-1}^{N}}{L_{c_l:c-1}} \psi_{c_l:c-1}(\mathrm{d}x_{c_l:c-1}^{1:N}, l_{c_l+1:c-1}^{1:N}, r_{c_l+1:c-1}^{1:N})\\
&\qquad\qquad \times \left\{ \prod_{i \neq m} W_{c_l}^{l_{c_l}^i, r_{c_l}^i} \right\} \omega_{c_l} \left(x_{c_l-1}^{\sigma^{l_{c_l}^m}_{a:c_l-1}(c_l-1)}, x_{c_l}^{\sigma^{r_{c_l}^m}_{c_l:c-1}(c_l)} \right). \qquad (33)
\end{aligned}
$$

The final structure of (33) mirrors that of (31), with the decomposition $\left\{\prod_{i\neq m} W_{c_l}^{l^i_{c_l}, r^i_{c_l}}\right\} \times \omega_{c_l}\left(x_{c_l-1}^{\sigma_{a:c_l-1}^{l^m_{c_l}}(c_l-1)}, x_{c_l}^{\sigma_{c_l:c-1}^{r^m_{c_l}}(c_l)}\right)$. This ensures that we can recursively decompose $\pi_{a:b}$ in a star trajectory and a remainder by defining $\sigma^*_{a:c-1} = \sigma^I_{a:c-1}$, $\sigma^*_{c:b} = \sigma^J_{c:b}$, where (I, J) is distributed according to a categorical distribution on $W_c^{m,n}$, and $\sigma^*_{a:b} = \left[\sigma^*_{a:c-1}, \sigma^*_{c:b}\right]$. This in turn defines $X^*_{a:b} \coloneqq X_{a:b}^{\sigma^*_{a:b}} = \left[X^*_{a:c-1}, X^*_{c:b}\right]$, and the related l^*_c, r^*_c, which correspond to the resampling indices pairs that eventually lead to the star trajectory. Following the recursive construction of (31) and (33), we are able to isolate the star trajectory from the rest of the variables appearing in $\pi_{a:b}$ to form the marginal distribution

$$\pi_{a:b}(\mathrm{d}x^*_{a:b}) \propto \prod_{c=a}^{b} \nu_c(\mathrm{d}x^*_c) \prod_{c=a}^{b} \omega_c(x^*_{c-1}, x^*_c), \tag{34}$$

which corresponds exactly to $\mathbb{Q}_{a:b}$. This allows us to formulate the following proposition.

Proposition 11 (Conditional dSMC) *Under $\pi_{a:b}$, the star trajectory $X^*_{a:b}$ is marginally distributed according to $\mathbb{Q}_{a:b}$, and the remaining variables admit as a conditional distribution, given the star trajectory, the distribution defined by Algorithm 4 (discussed in next section).*

Remark 12 *While we are concerned here with parallel-in-time smoothing, the construction above generalizes to the algorithm of Lindsten et al. (2017) which considers stitching independent SMC samplers by means of an operation akin to that of Proposition 3. This means that, provided that one is able to implement a conditional SMC for each individual SMC sampler in the tree structure – possibly in the form of an ancestor sampling algorihtm (Lindsten and Schön, 2012; Whiteley, 2010) – the conditional SMC properties can be preserved by a construction similar to the one we developed in this section for dSMC.*

3.2 Parallel-in-time particle Gibbs

The resulting algorithm resembles the classical conditional SMC algorithm of Andrieu et al. (2010), in that, similarly, we can implement it by simply enforcing that the first trajectory be preserved throughout the course of the recursion. In particular, only Algorithms 1 and 2 need to be modified. The conditional version of Algorithm 2 simply consists in prepending the star trajectory to the sampled proposal trajectories before computing the resulting weights. In other words, for a star trajectory $x^*_{0:T}$, if, for a given t, we are given $N-1$ i.i.d. samples $(X^n_t)_{n=2}^N$ from q_t, we set $X^1_t = x^*_t$ and then proceed with computing the weights in Algorithm 2 *as if x^*_t had simply been sampled from q_t too.* On the other hand the conditional version of Algorithm 1 consists in preserving said star trajectory throughout the resampling steps and is given by Algorithm 4.

Algorithm 4: Conditional Block combination

```
// All operations on indices $m, n$ are done in parallel
Result: Combine conditional particle representation of partial smoothing
        distributions
Function ConditionalCombine$(X^{1:N}_{a:c-1}, w^{1:N}_{c-1}, X^{1:N}_{c:b}, w^{1:N}_c)$
    Set $W^{m,n}_c \propto \omega_c(X^m_{c-1}, X^n_c) w^m_{c-1} w^n_c$
    Sample independently $N-1$ times from $\sum_{m,n} W^{m,n}_c \delta_{X^m_c}(\mathrm{d}x_c)\delta_{X^n_{c-1}}(\mathrm{d}x_{c-1})$ to
        get $(X^{l_n}_{c-1}, X^{r_n}_c)_{1 \le n \le N}$ // in parallel, using multinomial resampling
    // The two loops below can be done in parallel:
    for $c' = a, \dots, c-1$ in parallel do
        $\widetilde{X}^1_{c'} \leftarrow X^1_{c'}$
        $\widetilde{X}^n_{c'} \leftarrow X^{l_n}_{c'}$
    for $c' = c, \dots, b$ in parallel do
        $\widetilde{X}^1_{c'} \leftarrow X^1_{c'}$
        $\widetilde{X}^n_{c'} \leftarrow X^{r_n}_{c'}$
    return $\widetilde{X}^{1:N}_{a:b}$
```

Andrieu et al. (2010) considered implementing the conditional SMC step using a particle filter only, which resulted in lower mixing speeds for time steps further away from the last time step T. This was corrected by the introduction of the so-called backward sampling step (Whiteley, 2010; Lindsten and Schön, 2012), which enabled rejuvenating the conditional trajectories; see also Lindsten et al. (2014) for a related approach. A noteworthy point is that our proposed PIT particle Gibbs algorithm does not suffer from the classical genealogy degeneracy problem that prompted the development of the ancestor sampling step. This is due to the fact that the degeneracy arising in dSMC is essentially uniform across all time steps thanks to the balanced tree structure. Indeed, instead of the last time steps being resampled just a few times and the initial time steps being resampled around T times, as in standard SMC, all time steps in dSMC are resampled at most $\lceil \log_2 T \rceil$ times. This is also the reason why the $\mathcal{L}_p$ error in Proposition 4 scales as a polynomial of T and not exponentially. In practice, this means that the modified trajectories sampled from our conditional dSMC will mix similarly for initial timesteps and for final ones, provided that our proposal distributions q_t and auxiliary weight functions ν_t are adapted to the model and data at hand.

It is worth noting, however, that the backward sampling step of Whiteley (2010) additionally removes the need for scaling the number of particles, N, with the time horizon, T (at the cost of instead increasing the number of MCMC iterations required to converge, Lee et al., 2020). This property is likely not preserved by c-dSMC, and we expect that N needs to increase with T (at least logarithmically) in order to ensure proper ergodicity.

4. Variance reduction methods

A drawback of our method consists in the necessity to use independent proposals $q_{0:T}(\mathrm{d}x_{0:T}) = \prod_{t=0}^{T} q_t(\mathrm{d}x_t)$. It is well known that using such rough estimates increases the variance of the smoothing distribution estimates in particular in case of "sticky" processes which exhibit a strong time-dependency, or more precisely, when the conditional reverse Markov chain representing the smoothing distribution mixes slowly. However, this problem can be mitigated by using proposal distributions that are adapted to the model at hand. In Section 4.1 we describe how recently developed parallel-in-time Gaussian approximation based smoothing algorithms (Särkkä and García-Fernández, 2021; Yaghoobi et al., 2021) can be used to form such proposals. As these methods are also parallel in time, they do not relinquish the $\mathcal{O}(\log T)$ span complexity of the dSMC algorithm.

More prosaically, a natural way to reduce the variance of the smoothing estimators is to increase the number of particles used in the Monte Carlo representations. However, doing so in Algorithm 2.3 comes at a quadratic cost in memory and threads utilization. In Section 4.2 we discuss how we can leverage ideas from Murray et al. (2016) to lazily resample so as to keep a linear memory cost and reduce the computational burden.

4.1 Parallel-in-time Gaussian approximated smoothing solutions

It is well known that non-linear SSMs for which the state posterior distribution is unimodal can be approximated by LGSSMs. For example, consider an additive Gaussian noise transition model $p_t(x_t \mid x_{t-1}) = \mathcal{N}(x_t; f(x_{t-1}), Q_{t-1})\, \mathrm{d}x_t$. Under the Gaussian approximated assumption $p(x_t \mid y_{1:t}) \approx \mathcal{N}(x_t; m_t, P_t)$, we can use a Taylor linearization of the transition function f around the approximated mean m_t to form the linearized dynamics $x_{t+1} = f(m_t) + J[f](m_t)(x_t - m_t) + \epsilon_t$, where ϵ_t is a Gaussian random variable with mean 0 and covariance $J[f](m_t) Q_t J[f](m_t)^\top$ and $J[f](m_t)$ is the Jacobian of f evaluated at m_t. By repeating this approximation for each time step and for the observation model, we obtain the extended Kalman filter algorithm (Jazwinski, 1970). Similarly, one can use Taylor expansion in order to compute Gaussian approximations of the smoothing distribution marginals $p(x_t \mid y_{1:T})$ for all t, yielding the extended Kalman smoother algorithm. Other linearization techniques exist, such as statistical linearization (Gelb, 1974), sigma-point (unscented) methods (Julier et al., 2000; Särkkä, 2008), and numerical integration based methods (Ito and Xiong, 2000; Särkkä and Hartikainen, 2010). For a review, we refer the reader to Särkkä (2013).

In practice it is worth noting that the reference point used to linearize the system at time t (m_t for the extended Kalman filter example above) is arbitrary, and could be optimized instead of taking the result of the previous time step. This remark led to development of iterated extended Kalman filters (Bell and Cathey, 1993), iterated sigma-point filters (Sibley et al., 2006; Zhan and Wan, 2007), and general iterated statistical linear regression methods called posterior linearization filters (García-Fernández et al., 2015). When considering smoothing problems, it is even better to iteratively linearize with respect to the smoothing trajectory as is done in the iterated extended Kalman smoother (Bell, 1994). A general framework of iterated posterior linearization smoothers using this idea was developed in García-Fernández et al. (2017) and this was further generalized to more general state-space models in Tronarp et al. (2018). These methods result in Gaussian approximations to the marginals $p(x_t \mid y_{1:T}) \approx \mathcal{N}(x_t; m_t^l, P_t^l)$ which are optimal in a Kullback–Leibler sense (García-Fernández et al., 2015).

Recently, Särkkä and García-Fernández (2021) showed that by reformulating Bayesian filters and smoothers (including Kalman filters and smoothers) in terms of associative operators, it is possible to parallelize them along the time dimension by leveraging prefix-sum algorithms (Blelloch, 1989). This leads to logarithmic span-time complexity $\mathcal{O}(\log_2 T)$ instead of the conventional $\mathcal{O}(T)$ of sequential methods. Yaghoobi et al. (2021) then extended this framework to non-linear models by developing parallelized versions of the iterated extended Kalman smoothers as well as the more general iterated posterior linearization smoothers. This framework allows for computing the marginal approximations $p(x_t \mid y_{1:T}) \approx \mathcal{N}(x_t; m_t^l, P_t^l)$ in the $\mathcal{O}(\log_2 T)$ time complexity.

These Gaussian approximations to the smoothing distributions can now be used as proposal distributions q_t and/or weighting distributions ν_t in the proposed dSMC algorithm. The resulting method with $q_t = \nu_t$ is summarized in Algorithm 5.

Algorithm 5: PIT linearized proposal smoother

```
Function LinearizedSmoother($y_{1:T}$)
    for $t = 0, \dots, T$ in parallel do
        Initialize $q_t^0 = \mathcal{N}(x_t; m_t^0, P_t^0)$ // for example, using the stationary distribution
    Set $l \leftarrow 1$
    while convergence criterion not verified do
        Linearize (2) around $q_t^{l-1}$, for $t = 0, 1, \dots, T$ // Done in parallel
        Run parallel Kalman filter and RTS smoothers on the linearized system as per Särkkä and García-Fernández (2021) or Yaghoobi et al. (2021)
        Set $p(x_t \mid y_{1:T}) \approx q_t^l = \mathcal{N}(x_t; m_t^l, P_t^l)$, for $t = 0, 1, \dots, T$ // Done in parallel
        Set $l \leftarrow l + 1$
    Run the parallel smoother defined as per Algorithms 2 and 1
```

Similarly, we can tweak Algorithm 5 in order to define an efficient Gaussian proposal model for PIT pGibbs. Indeed, between two iterations of the d-cSMC described in Section 3.1, pGibbs typically proposes new parameters and we can expect the parameters of the state-space model to not have changed too much. Intuitively, this means that the optimum trajectory for the parallel IPLS method will not change much and we can therefore reuse the optimum of the previous Gibbs iteration as initialization for the next one. The benefit of doing so is shown in the experiment of Section 5.2.

4.2 Parallel resampling for lazy evaluation of the weight matrix

Algorithm 1 presented in Section 2.3 requires to form a $N \times N$ matrix to then sample N elements from it. Doing so limits the scalability of dSMC in at least two ways:

1. The memory cost will increase quadratically with the required number of particles. This is particularly problematic on parallel hardware such as GPUs where the memory available is usually more limited than the main (random-access memory) memory accessible via a CPU. For a large number of time steps or particles, our algorithm may therefore simply fail to return a result.

2. The number of threads available on GPUs, while increasing year-on-year, is still limited, and our algorithm computational scalability, although theoretically logarithmic in both N and T, may be affected by threading bottlenecks. See Section 5.1 for an illustration of this.

In order to mitigate both these issues, we can leverage the parallel resampling schemes proposed by Murray et al. (2016). Indeed, these can be modified in order to sample N entries from a set of $N \times N$ unnormalized weights *without needing to evaluate the whole matrix.* This property, although not discussed in Murray et al. (2016) can crucially be utilized to design lazy resampling schemes for our $N \times N$ size importance density (13). Formally, suppose we want to sample N pairs (I_m, J_m) independently from a categorical distribution $\mathrm{Cat}((W_c^{i,j})_{i,j=1}^N)$, where for all i, j, $W_c^{i,j} \propto \omega_c(X_{c-1}^i, X_c^j)$ for some time index c. This can be done in parallel across the N pairs (I_m, J_m) by considering N independent instances of a Metropolis–Hastings (Code 2 in Murray et al., 2016) algorithm with proposal (after proper flattening of the $N \times N$ matrix) $\mathcal{U}(\{1, \dots, N^2\})$ and target $\propto \omega_c(X_{c-1}^i, X_c^j)$. Similarly, when an upper bound $\overline{\omega}_c$ to ω_c is available, an unbiased rejection sampling equivalent (Code 3 in Murray et al., 2016) can be implemented. Under this perspective, we only need to evaluate the term $\omega_c(X_{c-1}^i, X_c^j)$ for the proposed pairs (i, j). This allows us to never increase the memory and thread utilization beyond $\mathcal{O}(N)$ operations at any point in time. On the other hand, this also means that we may inefficiently re-evaluate the same pair several times. However, as shown in Section 5.3, the parallelization makes this trade-off beneficial. For the sake of completeness, we reproduce the resulting resampling algorithms in Appendix B.

Finally, while using these lazy resampling schemes comes at a price (biasedness in the case of the Metropolis–Hastings variation and random execution time in the case of the rejection sampling one), as discussed in Murray et al. (2016, Sections 3.2 and 3.4), this trade-off becomes better as the variance of the weights $\omega_c(X_{c-1}^i, X_c^j)$ decreases.

5. Experiments

In order to illustrate the computational and statistical properties of our proposed methods, we now consider a set of examples from the literature and compare with the sequential counterparts of our methods. All the results were obtained using an Nvidia® GeForce RTX 3090 GPU with 24GB memory and the code to reproduce them can be found at `https://github.com/AdrienCorenflos/parallel-ps`.

5.1 Comparison with FFBS

In this section, we compare dSMC to the classical forward filtering backward sampling (FFBS) algorithm (Godsill et al., 2004), both in terms of execution time and Monte Carlo error. To make the comparison fairer, we also implement FFBS on GPU; in this way, FFBS scale as $\mathcal{O}(T \log N)$ (see, e.g., the prefix-sum implementation of classical resampling operations in Murray et al., 2016), since the particle operations are parallelizable up to a logarithmic factor (corresponding to computing the sum of the importance weights, which can be done using a prefix-sum algorithm). We consider the same model as in Chopin and Singh (2015) (which is a simplified version of the model in Yu and Meng (2011) for photon emission):

$$\begin{aligned} &x_0 \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{1-\rho^2}\right), \\ &x_t = \mu + \rho(\lambda x_{t-1} - \mu) + \epsilon_{t-1}, \quad \epsilon_{t-1} \sim \mathcal{N}(0, \sigma^2), \quad t \geq 1, \\ &y_t \sim \mathcal{P}(\exp(x_t)), \end{aligned} \tag{35}$$

where $\mathcal{P}(\exp(x_t))$ denotes a Poisson distribution with rate $\exp(x_t)$, and we want to estimate its Fisher score with respect to σ^2, and evaluated at σ^2:

$$\begin{aligned} &\mathbb{E}\left[\nabla_{\sigma^2} \ln p(X_{0:T}, y_{0:T}) \mid y_{0:T}\right] \\ &= \mathbb{E}\left[-\frac{T+1}{2\sigma^2} + \frac{1-\rho^2}{2\sigma^4}(X_0 - \mu)^2 + \frac{1}{2\sigma^4}\sum_{s=1}^{T}\left\{X_s - \mu - \rho(X_{s-1} - \mu)\right\}^2 \mid y_{0:T}\right]. \end{aligned} \tag{36}$$

Because of its additive nature, the variance of this expectation should increase as T increases, making it a good benchmark function to test our algorithm.

The stationary distribution of the underlying dynamics is $x_t \sim \mathcal{N}(\mu, \sigma^2/(1-\rho^2))$, so we take $q_t = \nu_t = \mathcal{N}(\mu, \sigma^2/(1-\rho^2))$ for all t.

In order to study the statistical and numerical properties of our algorithm we then generate 50 datasets $x_{0:T}, y_{0:T}$ from the model for $T = 32, 64, 128, 256, 512$ and repeat 100 dSMC and FFBS smoothing experiments on each dataset generated. The statistics we generate are then averaged over the datasets.

The resulting average running times (and 90% confidence intervals) of the corresponding algorithms are shown in Figure 2. Our algorithm is always faster than its sequential

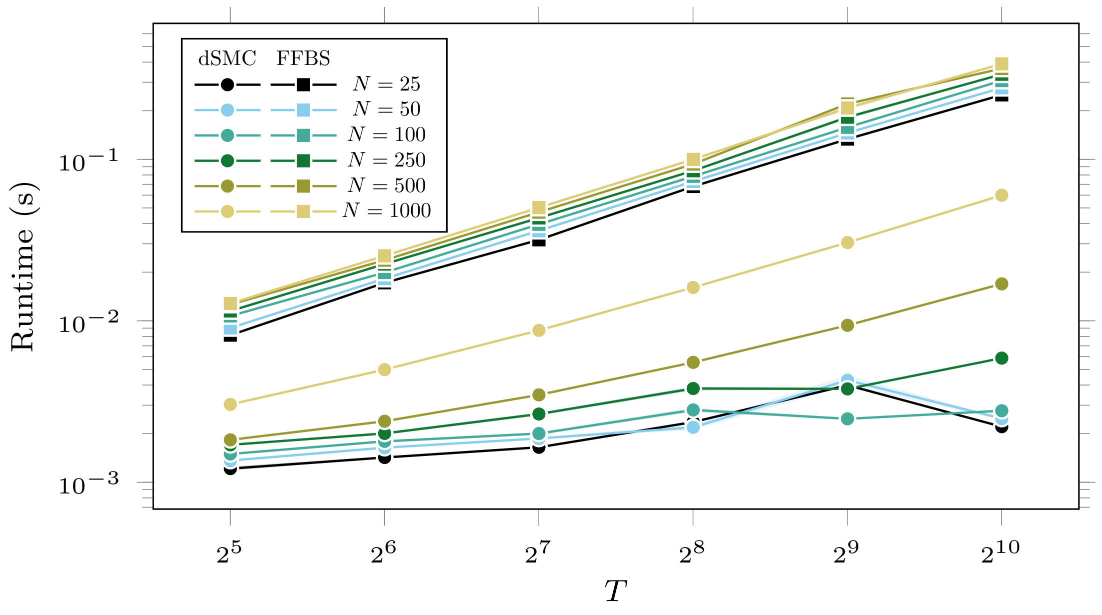


Figure 2: Average clock time of running a sequential FFBS vs dSMC. For T small enough, dSMC scales logarithmically, and then linearly when the parallelization threads have all been utilized. The effect is more pronounced for a higher number of particles. The 90% confidence intervals over the 50 datasets are also reported but hardly visible. The numerical artifact at $T = 2^9$ for $N = 25$ and $N = 50$ is not explained but may be due to the computational framework used.

FFBS counterpart. Due to the limited number of threads on our GPU, the logarithmic complexity scaling of our proposed method reaches a technical upper bound as we increase the number of sampled time steps. In particular the number of time steps that can effectively be parallelized is a decreasing function of the number of particles used. After the parallelization limit has been reached, dSMC scales linearly as further progress is blocked by waiting that a thread becomes free to use.

On the other hand, as can be expected from using independent proposals, our algorithm exhibits a larger error for estimating the Fisher score function, and this error increases with the number of time steps we want to sample. This effect is illustrated by Figure 3 where, for each dataset, we report the average relative error of computing the Fisher score for 100 runs of the particle smoothers, and report the average of this over all the datasets. There therefore exists a natural trade-off between speed and precision, which can be beneficial or not depending on the application. In the next section we show that the increase in variance does not necessarily affect sampling performance in practice.

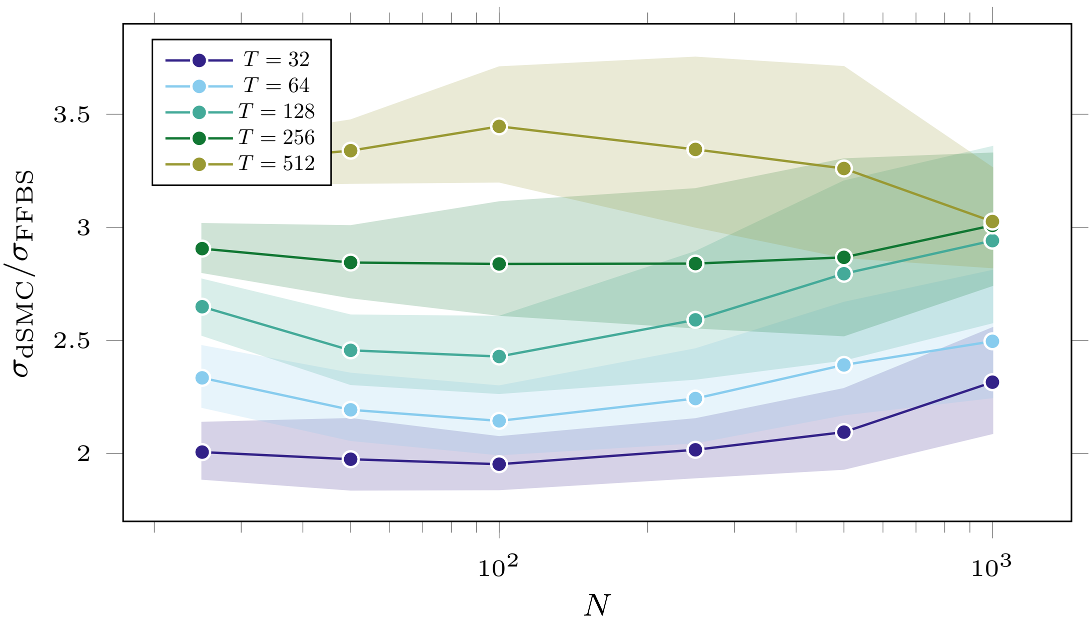


Figure 3: Average relative error (and 90% confidence interval thereof) of running a sequential FFBS vs dSMC. dSMC always exhibits a higher error than FFBS, the ratio between the two increasing as T increases.

5.2 Particle Gibbs sampling of theta-logistic model

The goal of this section is to show how the c-dSMC algorithm can be used to perform particle Gibbs sampling while not reducing its performance compared to the sequential version of cSMC. In order to illustrate the properties of this PIT pGibbs algorithm, we consider the following theta-logistic state-space model:

$$\begin{aligned} x_0 &\sim \mathcal{N}(0,1), \\ x_t &= x_{t-1} + \tau_0 - \tau_1 \exp(\tau_2 x_{t-1}) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,q^2), t \geq 1 \\ y_t &= x_t + \gamma_t, \quad \gamma_t \sim \mathcal{N}(0,r^2), \quad t \geq 0. \end{aligned} \tag{37}$$

This model was originally proposed by Lande et al. (2003) in order to model population dynamics and has been used as a benchmark for PMCMC methods in, for example, Peters et al. (2010), Chopin and Papaspiliopoulos (2020, Chap. 16). We use the same prior and data (nutria, $T + 1 = 120$) as in these references. We run two sets of experiments: one informed, where Kalman approximations are used, and one uninformed.

For c-dSMC, the “informed proposal” method is defined as taking $q_t = \nu_t$ to be “locally adapted”: $q_t(x_t) = p_{\text{EKS}}(x_t \mid y_{0:T}, \tau_0, \tau_1, \tau_2, q, r)$ given by the parallel extended Kalman smoother described in Section 4.1. More precisely, given an initial sample from the prior

$p(\tau_0, \tau_1, \tau_2, q, r)$, we compute the iterated EKS solution with 25 iterations and take the q_t's to be the resulting approximated smoothing marginal. For all subsequent steps, given new parameters, we run a single step of the iterated EKS, starting from the previous iterated EKS approximation, and use the updated Gaussian approximated smoothing marginals as our new proposal distributions q_t's. The "uninformed proposal" on the other hand, is taken to be a Gaussian proposal around the data: $q_t(x_t) = \mathcal{N}(x_t; y_t, r^2 + q^2)$.

For the classical cSMC, the "informed proposal" method is defined as a guided particle filter using the "locally optimal proposal" for (37) (see, e.g., Chopin and Papaspiliopoulos, 2020, Chap. 10.3.2). The "uninformed proposal" on the other hand, is taken to be the bootstrap proposal for the model. We use $N = 50$ particles for both the sequential and PIT versions of cSMC and report the run time of the experiments.

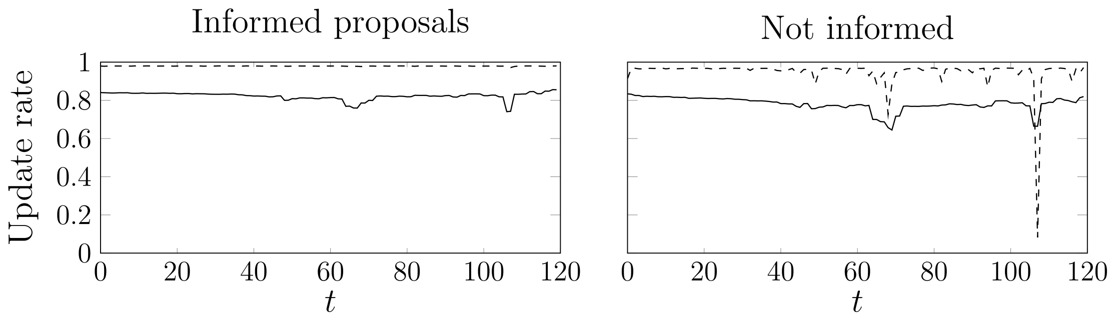


Figure 4: Average update rate of the star trajectory X_t^* for each time t. The average update rate of cSMC with backward sampling (- - -) is higher than that of c-dSMC (——), but in the case of uninformed proposals, the latter is more homogeneous across time steps. Using an adapted proposal marginally improves the resulting update rate in both cases, mostly by smoothing out the "dip" in the model.

In Figure 4, we report the update rate for the sampled trajectory, defined as the empirical probability that the star trajectory X_t^* is updated by running a conditional SMC. It varies between 70% and 80% for the uninformed c-dSMC, and 80% and 85% for the informed version, homogeneously across all time steps *without any explicit backward sampling step*. This is to be compared with the non-uniform renewal rates ($\approx$ 90%) of the uninformed standard pGibbs algorithm, and the almost ideal behavior of the informed standard pGibbs, when a backward sampling step (Whiteley, 2010; Lindsten and Schön, 2012) is implemented.

Moreover, obtaining 10^5 samples from the Gibbs chain took around 400 seconds with both proposal versions of c-dSMC, while it took around 4 000 seconds for both sequential cSMC samplers with backward sampling. Finally, the ACFs (auto-correlation functions) of the Markov chains formed by the parameters posterior samples are virtually identical, as illustrated by Figure 5 (we only report these for the adapted proposals, as there is no major difference with the uninformed ones).

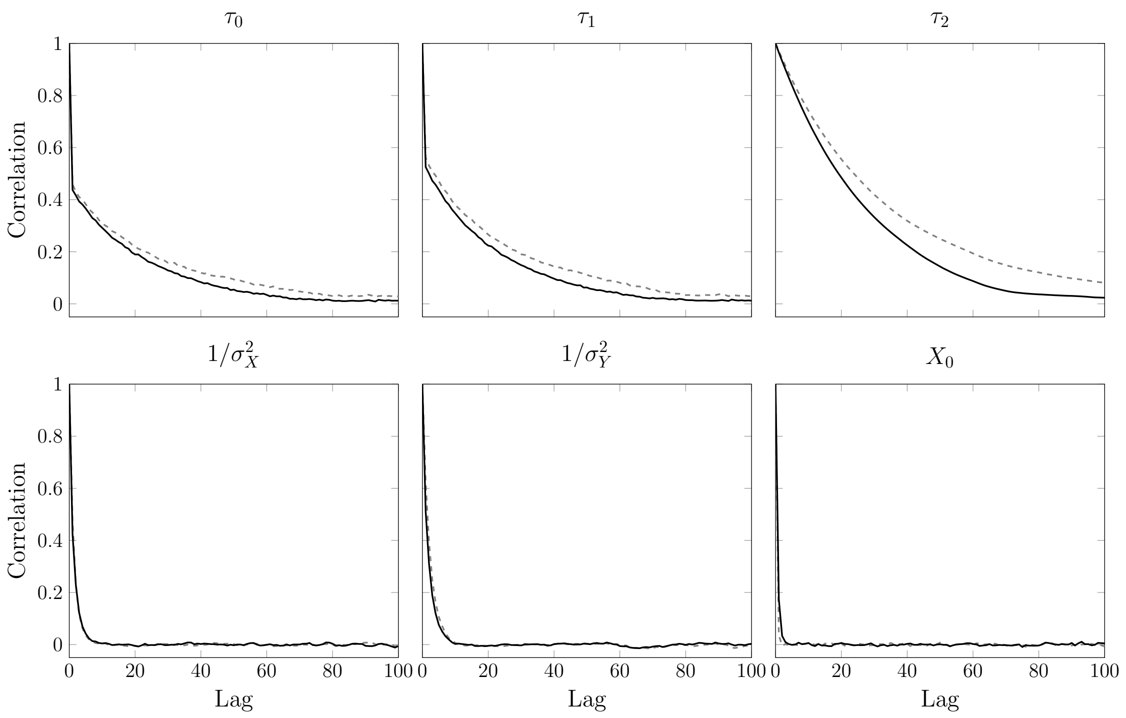


Figure 5: Auto-correlation plots for the parameters τ_0, τ_1, τ_2, $1/\sigma_X^2$, $1/\sigma_Y^2$, and the initial state X_0 posterior samples. Using cSMC with backward sampling (- - -) or c-dSMC (——) results in similar auto-correlation functions for the posterior samples.

5.3 Speed-up and variance reduction via lazy resampling

We now show how the lazy resampling methods introduced in Section 4.2 can help speed up dSMC significantly, while at the same time retaining the same variance as the original method. In order to do so, similarly to Deligiannidis et al. (2020, Section 4.1), we consider a constrained random walk model studied, for example, in Del Moral and Doucet (2004) and Adorisio et al. (2018). While, contrarily to these works, we are not concerned with exact simulation, this model is helpful in understanding the impact of the weights variance on the total runtime and variance of dSMC with lazy resampling. Indeed, the model is controlled by a single parameter σ which represents the noise of the constrained random walk, and directly impacts the variance of the weights in dSMC. Decreasing this parameter will increase the variance of the importance weights, reducing the performance of Monte Carlo methods. This type of behavior is akin to what happens when one increases the dimension of the state (or of the observation). Furthermore, this model is not easily approximated by an LGSSM, and therefore, the variance reduction method of Section 4.1 does not apply here.

Formally, the model is defined as follows:

$$\begin{aligned} &x_0 \sim \mathcal{N}(0,1), \\ &x_t = x_{t-1} + \sigma \epsilon_{t-1}, \quad \epsilon_{t-1} \sim \mathcal{N}(0,1), \end{aligned} \tag{38}$$

and we want to sample from $p(x_{0:T} \mid -1 \leq x_t \leq 1, t = 0, \dots, T)$. This model corresponds to the transition kernel $P_t(\mathrm{d}x_t \mid x_{t-1}) \sim \mathcal{N}(x_{t-1}, \sigma^2)$ with potential function $h_t(x_t) = \mathbb{1}_{[-1,1]}(x_t)$. Following Deligiannidis et al. (2020), we consider the proposal $q_t = \mathcal{U}([-1,1])$; the weights ω_t are then upper-bounded by $(2\pi\sigma)^{-1/2}$. As σ gets higher, we expect the lazy resampling schemes in Section 4.2 to perform better. In order to compare the different smoothing algorithms, we estimate $\mathbb{E}\left[\varphi(X_{0:T}) \mid -1 \leq X_t \leq 1, t = 0, \dots, T\right]$, where

$$\varphi(x_{0:T}) = \log(\sigma) + \frac{1}{\sigma^3} \sum_{t=1}^{T} (x_t - x_{t-1})^2, \tag{39}$$

which corresponds (up to a multiplicative constant) to the expected Fisher's score estimate of this model.

For the sake of simplicity, we only consider the rejection version of our lazy resampling methods. Recall that the Metropolis–Hastings version is biased, and thus our convergence theorems do not apply. On the other hand, Murray et al. (2016) find that it works better than the rejection counterpart in all the examples they consider.

In Figure 6, we take σ to be in $\{0.3, 0.4, 0.5\}$, this set being taken to be around the value when using lazy resampling starts to outperform FFBS, and we report the average (over 100 experiments) run times of FFBS, dSMC with systematic resampling (sys-dSMC), and dSMC with rejection-resampling (rs-dSMC) of computing Fisher's score estimates. In Figure 7, we report the respective variance of the resulting 100 score estimates.

For low N's, sys-dSMC is the fastest, with fairly high variance estimates of the Fisher score, as previously discussed in Section 5.1. However, for larger N values, despite its random run time, rs-dSMC completely outperforms both FFBS and sys-dSMC in terms of speed. Moreover, for $\sigma = 0.4$ and $\sigma = 0.5$ and all T's, the slowest running rs-dSMC ($N = 5000$) is faster than the fastest running FFBS ($N = 25$) and exhibits a lower Fisher score estimate variance than FFBS with more particles than $N = 25$. Finally, this improved performance becomes better as the number of time steps T increases, therefore confirming the appeal of dSMC for high values of T.

6. Discussion

In this article we have introduced de-Sequentialized Monte Carlo, the first fully parallel-in-time particle smoother. This algorithm exhibits $\mathcal{L}_p$ error bounds that scale polynomially in the number of times steps and inverse proportionally to the number of particles used. Futhermore, we have shown how one can build a conditional version of dSMC, to be used,

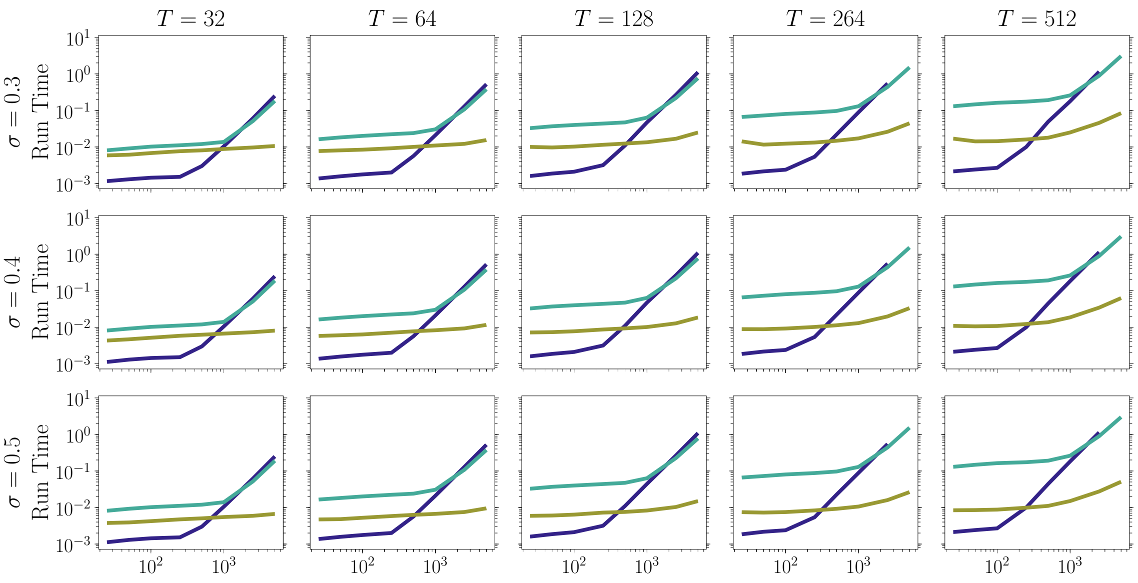


Figure 6: Average run times to compute Fisher's score estimate (39) as a function of the number of particles N for FFBS (🟩), sys-dSMC (🟦), and rs-dSMC (🟨), and different values of T and σ. We can see that for lower variance weights regimes (higher σ), rs-dSMC runs largely faster than both FFBS and sys-dSMC.

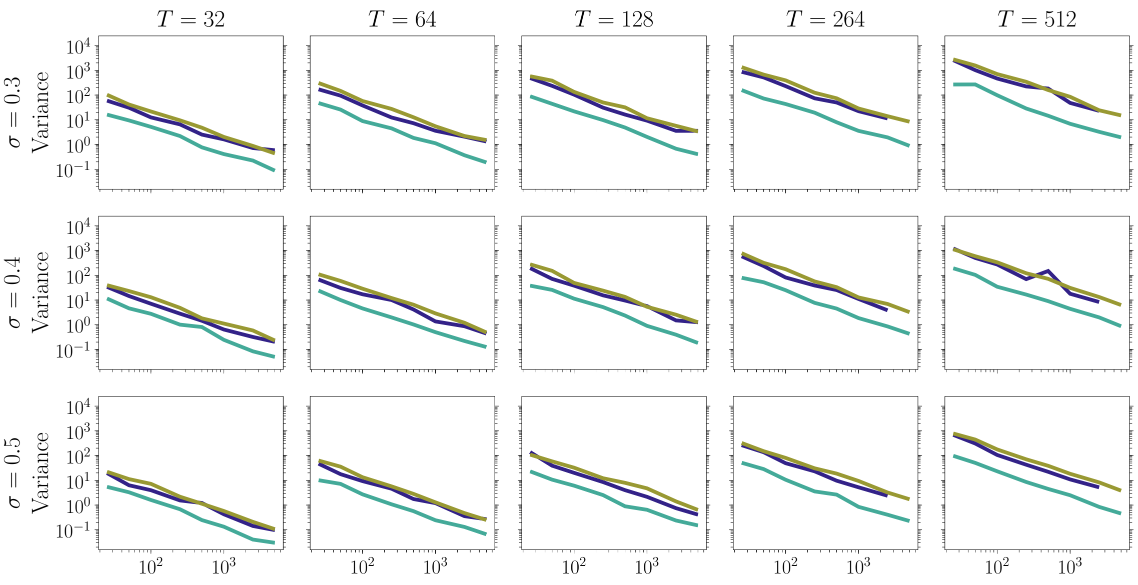


Figure 7: Variances of Fisher's score estimate (39) as a function of the number of particles N for FFBS (🟩), sys-dSMC (🟦), and rs-dSMC (🟨), and different values of T and σ. We can see that rs-dSMC retains roughly the same variance as sys-dSMC throughout.

for example, in particle Gibbs algorithms. Furthermore, we discussed two variance reduction schemes based on parallel-in-time linear Gaussian state-space models approximants, as well as lazy resampling schemes. The resulting algorithms have then been shown to be competitive with standard sequential methods in different non-trivial regimes.

While the Gaussian approximations recover a lot of practical use cases, their nature makes them inadequate to approximate, for example, multi-modal posteriors. Designing proposals with more modeling capacity, and fully utilizing the additional degree of freedom offered by the different roles of ν and q is an important direction of future work. This could be done, for instance, using direct gradient methods (Corenflos et al., 2021; Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018) or more iterative methods (Guarniero et al., 2017; Heng et al., 2020).

Our parallel smoother exhibits good statistical and computational properties in non-trivial regimes, and allows faster inference at the cost of some precision. The loss of precision comes from the need to use independent proposal distributions, and hinders inference in high variance regimes (such as high dimensional spaces). We believe future research should maybe directed towards using pathwise proposals instead, for example by further leveraging the LGSSM approximants of Yaghoobi et al. (2021).

Because we developed a conditional version of dSMC, our algorithm can be used *mutatis mutandis* within the unbiased coupled smoothing framework of Jacob et al. (2019). While (non-lazy) dSMC exhibits higher variance than its sequential counterparts (for the same number of particles), the framework of Jacob et al. (2019) allows to average independent such estimates to increase the precision of the resulting estimate arbitrarily, making the gain of speed particularly attractive in this context.

Another avenue of future work is to study the ergodic properties of the parallel-in-time particle Gibbs we developed in Section 3, in particular, how one needs to choose the number of particles N as a function of T. We believe that, contrary to conditional SMC with backward sampling (Lee et al., 2020), N needs to increase with T. Understanding the exact relationship between T and N in our case requires a careful examination and is an interesting direction of research.

An important technical limitation of our methodology is the necessity, at each level of the recursion, to explicitly form several $N \times N$ matrices. While this does not impact the theoretical logarithmic properties of our algorithm, this clearly limits the number of particles that we can use in at least two ways: the memory footprint will scale quadratically with it, and the number of threads being limited, a processing bottleneck may appear (as illustrated in Figure 2). We mitigated these issues by utilizing the parallel resampling perspective of Murray et al. (2016) as a lazy resampling scheme, never computing more than N weights at once, which allowed us to improve the scalability of dSMC in the low weights variance regime. We believe that this method can be further improved by using non-uniform proposals on the indices pairs (I, J) to target specific pairs that have a higher *a priori* chance of resulting in a high weight. It was also suggested in Corenflos and Särkkä (2022) that using ensemble techniques in parallel resampling schemes may result in an improved performance at the cost of a slightly higher memory consumption. Both these extensions deserve more investigation.

On the computational resource perspective, over the years parallel processing hardware have continually increased both the memory and number of threads, so we expect our algorithm to become increasingly competitive in the future. Similarly, it is also possible to distribute the computations across several processors (be it GPUs or CPUs), which in turn would result in making the algorithm scale better with the number of time steps or particles, provided that the communication cost between processors remains limited. Combining this technical solution with the lazy resampling approach of Section 4.2 in particular would likely result in a very competitive smoothers.

Finally, it was recently suggested in Deligiannidis et al. (2020) that it is possible to perform perfect sampling of SSMs smoothing distributions provided we use independent proposals. While our algorithm does not sample exactly from the same proposal distribution, it is our hope that the methods developed here could be applied to sampling from their proposal distribution too, thereby making their sampling algorithm scale logarithmically in time.

Acknowledgments

Adrien Corenflos and Simo Särkkä gratefully acknowledge the support of Academy of Finland (project 321900). The authors would also like to thank Francesca R. Crucinio for spotting a typo in the proof of Proposition 3, as well as the two anonymous referees and the editor for their useful comments and suggestions on this article.

Individiual contributions

The original idea for this article comes from discussions between Adrien Corenflos and Simo Särkkä. The methodology of dSMC was developed by Adrien Corenflos in collaboration with Nicolas Chopin. The pGibbs and lazy resampling extensions are both due to Adrien Corenflos while the LGSSM approximants are jointly due to Simo Särkkä and Adrien Corenflos. The original proofs of this article's results are due to Adrien Corenflos, the convergence rate of Proposition 4 being subsequently improved with the help of Nicolas Chopin. The experimental results are all due to Adrien Corenflos. The first version of this article was written by Adrien Corenflos, after which all authors contributed to the writing.

Appendix A. Parallel combination algorithm

We now reproduce a parallel equivalent to Algorithm 3. It can generally be thought of a divide-and-conquer algorithm akin to prefix-sum algorithms, but not requiring associativity of the operator. Algorithm 6 is phrased in terms of generic operators and elements which, in the particular case of parallel particle smoothing, need to be taken to be, respectively, the operator defined in Algorithm 1 and the set of particles, weights and partial normalizing constants.

Algorithm 6: Generic parallel combination via array reshaping

```
Result: Combined array
Function ParallelReshapeCombination($Z_{1:K}$, Operator)
    Find $L$ such that $2^{L-1} < K \leq 2^L$
    for $t = 1, \dots, K$ in parallel do
        // Flag that says if we should use the value or not
        $b_t \leftarrow 1$
    // Pad $Z$ to the next power of 2 using some NULL value
    for $t = K+1, \dots, 2^L$ in parallel do
        $Z_t \leftarrow$ NULL
        $b_t \leftarrow 0$
    for $l = 0, \dots L-1$ do
        for $n = 1, \dots 2^{L-l}$ in parallel do
            /* Join the $Z$'s block by block, this corresponds to
               reshaping the array and do not result in creating a new
               array.                                                   */
            $Y_n \leftarrow \left[Z_{1+(n-1)2^l}, Z_{2+(n-1)2^l}, \dots Z_{n2^l}\right]$
        for $n = 1, \dots 2^{L-l}$ in parallel do
            /* Combine the adjacent odd and even $Y$'s if we have not
               reached the padding threshold, otherwise, just leave the
               data unchanged.                                          */
            if $b_{1+n2^l} = 1$ then
                $\left[Z_{1+(n-1)2^l}, Z_{2+(n-1)2^l}, \dots Z_{n2^l}\right], \left[Z_{1+n2^l}, Z_{2+n2^l}, \dots Z_{(n+1)2^l}\right] \leftarrow$
                 CombinationOperator $(Y_n, Y_{n+1})$
    return $Z_{1:K}$
```

It is worth noting that Algorithm 6 and Algorithm 3 are not strictly equivalent. This is because the combination operator used for smoothing is random and depends on the state of a random number generator. In fact two reasons make these two algorithm differ:

1. The order in which the nodes at a given depth of Algorithm 3 are handled is arbitrary. Similarly for the order in which we combine adjacent blocks in Algorithm 6.

2. The splitting of Algorithm 6, although corresponding to a balanced tree, will not correspond to the mid-point splitting of Algorithm 3 except when $T+1$ is a power of 2.

However, both algorithms are consistent and can be analyzed by Proposition 4 in the same way.

Appendix B. Lazy resampling algorithms

We now describe the lazy resampling algorithms introduced in Section 4.2. The Metropolis–Hastings version is given by Algorithm 7, while the rejection sampling one is given by Algorithm 8. In Algorithm 7, B is a user defined parameter corresponding to the assumed number of MCMC steps required for the Markov chain to converge to the categorical distribution $\mathrm{Cat}((W_c^{i,j})_{i,j=1}^N)$ (Murray et al., 2016, Section 2.1). Some guidance on how to choose B is provided in Murray et al. (2016), and directly applies to Algorithm 7.

Algorithm 7: Metropolis–Hastings lazy resampling algorithm

```
Result: Resampling indices $(I_m, J_m)$ for $m = 1, \dots, N$.
Function MHResampling$\big(X_{c-1}^{1:N}, X_c^{1:N}, \omega_c, B\big)$
    for $m = 1, \dots, N$ in parallel do
        $(I_m, J_m) \leftarrow (m, m)$
        for $b = 1, \dots, B$ do
            Sample $u \sim \mathcal{U}([0,1])$
            Sample $I^*, J^* \sim \mathcal{U}(\{1, \dots, N\})$, independently
            if $u < \omega_c(X_{c-1}^{I^*}, X_c^{J^*}) / \omega_c(X_{c-1}^{I_m}, X_c^{J_m})$ then
                $(I_m, J_m) \leftarrow (I^*, J^*)$
    return $I_{1:N}, J_{1:N}$
```

It is worth noting that, contrarily to Algorithm 3 (Code 3) in Murray et al. (2016), the initial proposal in Algorithm 8 is random and not deterministic. This is because the deterministic starting point of Murray et al. (2016) would result in a bias when subsampling N candidates from the $N \times N$ entries in the weight matrix.

Appendix C. Proof of Proposition 4

For simplicity we only consider the case when $w_{c-1}^n = w_c^n = 1/N$, for all $n \in \{1, \dots, N\}$. The general case follows from the same lines. Using Minkowski's inequality, we have

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \mathbb{Q}_{a:b}^N(\varphi)\right|^p\right]^{1/p} \leq \mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \widetilde{\mathbb{Q}}_{a:b}^N(\varphi)\right|^p\right]^{1/p} + \mathbb{E}\left[\left|\widetilde{\mathbb{Q}}_{a:b}^N(\varphi) - \mathbb{Q}_{a:b}^N(\varphi)\right|^p\right]^{1/p}. \tag{40}$$

Algorithm 8: Rejection-sampling lazy resampling algorithm

```
Result: Resampling indices $(I_m, J_m)$ for $m = 1, \dots, N$.
Function RSResampling$\left(X_{c-1}^{1:N}, X_c^{1:N}, \omega_c, \overline{\omega}_c\right)$
    // $\overline{\omega}_c$ is such that $\omega_c(x, y) \leq \overline{\omega}_c$ for all $x, y$.
    for $m = 1, \dots, N$ in parallel do
        Sample $I_m$, $J_m \sim \mathcal{U}(\{1, \dots, N\})$, independently
        Sample $u \sim \mathcal{U}([0, 1])$
        while $u > \omega_c(X_{c-1}^{I_m}, X_c^{J_m})/\overline{\omega}_c$ do
            Sample $I_m, J_m \sim \mathcal{U}(\{1, \dots, N\})$, independently
    return $I_{1:N}, J_{1:N}$
```

The second term of (40), corresponding to the resampling error, can be controlled as a Monte Carlo error via Del Moral (2004, Lemma 7.3.3). Indeed, let us first notice that we have $\mathbb{E}\left[\mathbb{Q}_{a:b}^N(\varphi) \mid X_{a:b}^{1:N}\right] = \sum_{m,n=1}^N W_c^{m,n} \varphi\left(X_{a:c-1}^m, X_{c:b}^n\right)$, and that, given that we are considering the multinomial resampling case, conditionally on $X_{a:b}^{1:N}$, the variables $(l_n, r_n)_{n=1}^N$ are independent. In this case,

$$\mathbb{E}\left[\left|\widetilde{\mathbb{Q}}_{a:b}^N(\varphi) - \mathbb{Q}_{a:b}^N(\varphi)\right|^p \mid X_{a:b}^{1:N}\right]^{1/p} \leq d(p) \frac{\|\varphi\|_\infty}{N^{1/2}} \tag{41}$$

for some constant $d(p) \leq 2^{(p+1)/p}$, so that the tower law ensures that

$$\mathbb{E}\left[\left|\widetilde{\mathbb{Q}}_{a:b}^N(\varphi) - \mathbb{Q}_{a:b}^N(\varphi)\right|^p\right]^{1/p} \leq 2^{(p+1)/p} \frac{\|\varphi\|_\infty}{N^{1/2}} \tag{42}$$

is verified too.

On the other hand, the first term of (40), corresponding to the self-normalization error, requires more attention. In order to simplify notations, let us introduce the following quantities:

$$\begin{aligned} \widehat{\mathbb{Q}}_{c:b}^N(\varphi) &:= \frac{1}{N} \sum_{n=1}^N \int \bar{\omega}_c\left(X_{c-1}^n, x_c\right) \varphi\left(X_{a:c-1}^n, x_{c:b}\right) \mathbb{Q}_{c:b}(\mathrm{d}x_{c:b}), \\ \breve{\mathbb{Q}}_{a:b}^N(\varphi) &:= \frac{1}{N^2} \sum_{m,n=1}^N \bar{\omega}_c\left(X_{c-1}^m, X_c^n\right) \varphi\left(X_{a:c-1}^m, X_{c:b}^n\right). \end{aligned} \tag{43}$$

Using Minkowski's inequality again, twice, we can now decompose the first term of (40) as

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \widetilde{\mathbb{Q}}_{a:b}^N(\varphi)\right|^p\right]^{1/p} \leq \mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \breve{\mathbb{Q}}_{a:b}^N(\varphi)\right|^p\right]^{1/p} + \mathbb{E}\left[\left|\breve{\mathbb{Q}}_{a:b}^N(\varphi) - \widetilde{\mathbb{Q}}_{a:b}^N(\varphi)\right|^p\right]^{1/p}, \tag{44}$$

so that, splitting once more, we have

$$\begin{aligned}\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi)-\breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p} \leq\ & \mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi)-\widehat{\mathbb{Q}}^N_{c:b}(\varphi)\right|^p\right]^{1/p} \\ &+\mathbb{E}\left[\left|\widehat{\mathbb{Q}}^N_{c:b}(\varphi)-\breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p}.\end{aligned} \tag{45}$$

Let us first remark that

$$\mathbb{Q}_{a:b}(\varphi)=\mathbb{Q}_{a:c-1}\left(x_{a:c-1}\mapsto\int\bar{\omega}_c\left(x_{c-1},x_c\right)\varphi\left(x_{a:c-1},x_{c:b}\right)\mathbb{Q}_{c:b}(\mathrm{d}x_{c:b})\right). \tag{46}$$

Then, the integrand $x_{a:c-1}\mapsto\int\bar{\omega}_c\left(x_{c-1},x_c\right)\varphi\left(x_{a:c-1},x_{c:b}\right)\mathbb{Q}_{c:b}(\mathrm{d}x_{c:b})$ is upper bounded by $\|\bar{\omega}_c\|_\infty\|\varphi\|_\infty$, so that we can apply the recursion hypothesis to get

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi)-\widehat{\mathbb{Q}}^N_{c:b}(\varphi)\right|^p\right]^{1/p}\leq C^p_{a:c-1}\|\bar{\omega}_c\|_\infty\frac{\|\varphi\|_\infty}{N^{1/2}}. \tag{47}$$

On the other hand, using the tower law, the second term of (45) becomes

$$\mathbb{E}\left[\left|\widehat{\mathbb{Q}}^N_{c:b}(\varphi)-\breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]=\mathbb{E}\left[\mathbb{E}\left[\left|\widehat{\mathbb{Q}}^N_{c:b}(\varphi)-\breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\mid X^{1:N}_{a:c-1}\right]\right]. \tag{48}$$

Noting that

$$\widehat{\mathbb{Q}}^N_{c:b}(\varphi)=\mathbb{Q}_{c:b}\left(x_{c:b}\mapsto N^{-1}\sum_{n=1}^N\bar{\omega}_c\left(X^n_{c-1},x_c\right)\varphi\left(X^n_{a:c-1},x_{c:b}\right)\right), \tag{49}$$

and that, for all $n=1,\dots,N$ and all $x_{c:b}$, $N^{-1}\sum_{n=1}^N\bar{\omega}_c\left(X^n_{c-1},x_c\right)\varphi\left(X^n_{a:c-1},x_{c:b}\right)\leq\|\bar{\omega}_c\|_\infty\|\varphi\|_\infty$, we can leverage the recursion hypothesis one more time to obtain

$$\mathbb{E}\left[\left|\widehat{\mathbb{Q}}^N_{c:b}(\varphi)-\breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\mid X^{1:N}_{a:c-1}\right]^{1/p}\leq C^p_{a:c-1}\|\bar{\omega}_c\|_\infty\frac{\|\varphi\|_\infty}{N^{1/2}} \tag{50}$$

and, applying the tower law again,

$$\mathbb{E}\left[\left|\widehat{\mathbb{Q}}^N_{c:b}(\varphi)-\breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p}\leq C^p_{a:c-1}\|\bar{\omega}_c\|_\infty\frac{\|\varphi\|_\infty}{N^{1/2}}. \tag{51}$$

This ensures that

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi)-\breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p}\leq 2C^p_{a:c-1}\|\bar{\omega}_c\|_\infty\frac{\|\varphi\|_\infty}{N^{1/2}}. \tag{52}$$

Similarly, instead of introducing $\widehat{\mathbb{Q}}^N_{c:b}(\varphi)$, we could have introduced the similar quantity

$$\widehat{\mathbb{Q}}^N_{a:c-1}(\varphi) = \frac{1}{N}\sum_{n=1}^{N}\int \bar{\omega}_c\left(x_{c-1}, X^n_c\right)\varphi\left(x_{a:c-1}, X^n_{c:b}\right)\mathbb{Q}_{a:c-1}(\mathrm{d}x_{a:c-1})$$

to obtain:

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p} \leq 2C^p_{c:b}\|\bar{\omega}_c\|_\infty\frac{\|\varphi\|_\infty}{N^{1/2}}. \tag{53}$$

This finally ensures that

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \breve{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p} \leq 2\min(C^p_{a:c-1}, C^p_{c:b})\|\bar{\omega}_c\|_\infty\frac{\|\varphi\|_\infty}{N^{1/2}}. \tag{54}$$

Now the term $\mathbb{E}\left[\left|\breve{\mathbb{Q}}^N_{a:b}(\varphi) - \widetilde{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p}$ can be controlled in a way similar to the one used in (Chopin and Papaspiliopoulos, 2020, Lemma 11.2). Indeed we first note that $\widetilde{\mathbb{Q}}^N_{a:b}(\varphi) - \breve{\mathbb{Q}}^N_{a:b}(\varphi) = \widetilde{\mathbb{Q}}^N_{a:b}(\varphi)\left(1 - \breve{\mathbb{Q}}^N_{a:b}(1)\right)$, so that

$$\mathbb{E}\left[\left|\breve{\mathbb{Q}}^N_{a:b}(\varphi) - \widetilde{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p} \leq \|\varphi\|_\infty\mathbb{E}\left[\left|1 - \breve{\mathbb{Q}}^N_{a:b}(1)\right|^p\right]^{1/p}. \tag{55}$$

Moreover, $\mathbb{Q}_{a:b}(1) = 1$ by definition, so that we can rewrite

$$\mathbb{E}\left[\left|1 - \breve{\mathbb{Q}}^N_{a:b}(1)\right|^p\right]^{1/p} = \mathbb{E}\left[\left|\mathbb{Q}_{a:b}(1) - \breve{\mathbb{Q}}^N_{a:b}(1)\right|^p\right]^{1/p} \tag{56}$$

which can be bounded similarly to (54), giving

$$\mathbb{E}\left[\left|1 - \breve{\mathbb{Q}}^N_{a:b}(1)\right|^p\right]^{1/p} \leq 2\min(C^p_{a:c-1}, C^p_{c:b})\|\bar{\omega}_c\|_\infty\frac{1}{N^{1/2}}. \tag{57}$$

This results in the following inequality

$$\mathbb{E}\left[\left|\breve{\mathbb{Q}}^N_{a:b}(\varphi) - \widetilde{\mathbb{Q}}^N_{a:b}(\varphi)\right|^p\right]^{1/p} \leq 2\min(C^p_{a:c-1}, C^p_{c:b})\|\bar{\omega}_c\|_\infty\frac{\|\varphi\|_\infty}{N^{1/2}}. \tag{58}$$

Putting everything together, we obtain

$$\mathbb{E}\left[\left|\mathbb{Q}_{a:b}(\varphi) - \mathbb{Q}^N_{a:b}(\varphi)\right|^p\right]^{1/p} \leq \left(4\min(C^p_{a:c-1}, C^p_{c:b})\|\bar{\omega}_c\|_\infty + 2^{(p+1)/p)}\right)\frac{\|\varphi\|_\infty}{N^{1/2}}. \tag{59}$$

References

M. Adorisio, A. Pezzotta, C. de Mulatier, C. Micheletti, and A. Celani. Exact and efficient sampling of conditioned walks. *Journal of Statistical Physics*, 170(1):79–100, 2018.

L. Aitchison. Tensor Monte Carlo: Particle methods for the GPU era. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

C. Andrieu, A. Doucet, and E. Punskaya. Sequential Monte Carlo methods for optimal filtering. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 79–95. Springer New York, New York, NY, 2001.

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

I. Arasaratnam and S. Haykin. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009.

Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley, New York, NY, 2001.

B. M. Bell. The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization*, 4(3):626–636, 1994.

B. M. Bell and F. W. Cathey. The iterated Kalman filter update as a Gauss–Newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: a review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

G. E. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11):1526–1538, 1989.

N. Chopin and O. Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer International Publishing, 2020.

N. Chopin and S. S. Singh. On particle Gibbs sampling. *Bernoulli*, 21(3):1855–1883, 2015.

A. Corenflos and S. Särkkä. The coupled rejection sampler. *arXiv preprint arXiv:2201.09585 (version 1)*, 2022.

A. Corenflos, J. Thornton, G. Deligiannidis, and A. Doucet. Differentiable particle filtering via entropy-regularized optimal transport. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2100–2111. PMLR, 2021.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 3rd edition, 2009.

D. Crisan and A. Doucet. Convergence of sequential Monte Carlo methods. *Signal Processing Group, Department of Engineering, University of Cambridge, Technical Report CUEDIF-INFENGrrR38*, 1, 2000.

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

P. Del Moral and A. Doucet. Particle motions in absorbing medium with hard and soft obstacles. *Stochastic Analysis and Applications*, 22(5):1175–1207, 2004.

P. Del Moral and A. Guionnet. On the stability of interacting processes with applications to filtering and genetic algorithms. *Annales de l'Institut Henri Poincaré (B) Probability and Statistics*, 37(2):155–194, 2001.

G. Deligiannidis, A. Doucet, and S. Rubenthaler. Ensemble rejection sampling. *arXiv preprint arXiv:2001.09188 (version 1)*, 2020.

D. Ding and A. Gandy. Tree-based particle smoothing algorithms in a hidden Markov model. *arXiv preprint arXiv:1808.08400 (version 1)*, 2018.

A. Doucet, S. J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

G. Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

Á. F. García-Fernández, L. Svensson, M. R. Morelande, and S. Särkkä. Posterior linearization filter: principles and implementation using sigma points. *IEEE Transactions on Signal Processing*, 63(20):5561–5573, 2015.

Á. F. García-Fernández, L. Svensson, and S. Särkkä. Iterated posterior linearization smoother. *IEEE Transactions on Automatic Control*, 62(4):2056–2063, 2017.

A. Gelb. *Applied Optimal Estimation.* MIT press, 1974.

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, 2004.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

P. Guarniero, A. M. Johansen, and A. Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647, 2017.

S. Hassan, S. Särkkä, and A. F. García-Fernández. Temporal parallelization of inference in hidden Markov models. *IEEE Transactions on Signal Processing*, 69:4875–4887, 2021.

J. Heng, A. N. Bishop, G. Deligiannidis, and A. Doucet. Controlled sequential Monte Carlo. *The Annals of Statistics*, 48(5):2904–2929, 2020.

K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, 2000.

P. E. Jacob, F. Lindsten, and T. B. Schön. Smoothing with couplings of conditional particle filters. *Journal of the American Statistical Association*, 2019.

A. H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, New York, NY, 1970.

S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82 (Series D):35–45, 1960.

J. Kuntz, F. R. Crucinio, and A. M. Johansen. Product-form estimators: exploiting independence to scale up Monte Carlo. *arxiv preprint arXiv:2102.11575 (version 3)*, 2021a.

J. Kuntz, F. R. Crucinio, and A. M. Johansen. The divide-and-conquer sequential Monte Carlo algorithm: theoretical properties and limit theorems. *arXiv preprint arXiv:2110.15782 (version 1)*, 2021b.

R. Lande, S. Engen, and B.-E. Saether. *Stochastic population dynamics in ecology and conservation.* Oxford University Press on Demand, 2003.

T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-encoding sequential Monte Carlo. In *ICLR*, 2018.

A. Lee, C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789, 2010.

A. Lee, S. S. Singh, and M. Vihola. Coupled conditional backward sampling particle filter. *The Annals of Statistics*, 48(5):3066–3089, 2020.

F. Lindsten and T. B. Schön. On the use of backward simulation in the particle Gibbs sampler. In *Proceedings of the 37th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Kyoto, Japan*, 2012.

F. Lindsten, M. I. Jordan, and T. B. Schön. Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research*, 15:2145–2184, 2014.

F. Lindsten, A. M. Johansen, C. A. Naesseth, B. Kirkpatrick, T. B. Schön, J. Aston, and A. Bouchard-Côté. Divide-and-conquer with sequential Monte Carlo. *Journal of Computational Statistics and Graphics*, 26:445–458, 2017.

P. M. Lyster, S. E. Cohn, R. Ménard, L.-P. Chang, S.-J. Lin, and R. G. Olsen. Parallel implementation of a Kalman filter for constituent data assimilation. *Monthly Weather Review*, 125(7):1674–1686, 1997.

C. J. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. W. Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, 2017.

L. Middleton, G. Deligiannidis, A. Doucet, and P. E. Jacob. Unbiased smoothing using particle independent Metropolis-Hastings. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2378–2387. PMLR, 2019.

S. Muchnick. *Advanced Compiler Design and Implementation.* Morgan Kaufmann, 1997.

L. M. Murray, A. Lee, and P. E. Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3):789–805, 2016.

C. A. Naesseth, S. W. Linderman, R. Ranganath, and D. M. Blei. Variational sequential Monte Carlo. In *AISTATS*, 2018.

G. W. Peters, G. R. Hosack, and K. R. Hayes. Ecological non-linear state space model selection via adaptive particle Markov chain Monte Carlo (AdPMCMC). *arXiv preprint arXiv:1005.2238 (version 1)*, 2010.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

K. F. Riley, M. P. Hobson, and S. J. Bence. *Mathematical Methods for Physics and Engineering: A Comprehensive Guide.* Cambridge University Press, 2006.

O. Rosen and A. Medvedev. Efficient parallel implementation of state estimation algorithms on multicore platforms. *IEEE Transactions on Control Systems Technology*, 21(1):107–120, 2013.

S. Särkkä. Unscented Rauch-Tung-Striebel smoother. *IEEE Transactions on Automatic Control*, 53(3):845–849, 2008.

S. Särkkä. *Bayesian Filtering and Smoothing.* Cambridge University Press, 2013.

S. Särkkä and Á. F. García-Fernández. Temporal parallelization of Bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2021.

S. Särkkä and J. Hartikainen. On Gaussian optimal smoothing of non-linear state space models. *IEEE Transactions on Automatic Control*, 55(8):1938–1941, 2010.

G. Sibley, G. S. Sukhatme, and L. H. Matthies. The iterated sigma point Kalman filter with applications to long range stereo. *Robotics: Science and Systems*, 8(1):235–244, 2006.

S. S. Singh, F. Lindsten, and E. Moulines. Blocking strategies and stability of particle Gibbs samplers. *Biometrika*, 104(4):953–969, 2017.

F. Tronarp, Á. F. García-Fernández, and S. Särkkä. Iterative filtering and smoothing in nonlinear and non-Gaussian systems using conditional moments. *IEEE Signal Processing Letters*, 25(3):408–412, 2018.

N. Whiteley. Discussion of 'Particle Markov chain Monte Carlo methods' by Andrieu et al. *J. R. Statist. Soc. B*, 72(3):306–307, 2010.

F. Yaghoobi, A. Corenflos, S. Hassan, and S. Särkkä. Parallel iterated extended and sigma-point Kalman smoothers. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5350–5354, 2021.

Y. Yu and X.-L. Meng. To center or not to center: that is not the question—an ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC efficiency. *Journal of Computational and Graphical Statististics*, 20(3):531–570, 2011.

R. Zhan and J. Wan. Iterated unscented Kalman filter for passive target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):1155–1163, 2007.

Publication V

Adrien Corenflos and Simo Särkkä. Auxiliary MCMC samplers for parallelisable inference in high-dimensional latent dynamical systems. Submitted to *Electronic Journal of Statistics*, July 2023.

Electronic Journal of Statistics
Vol. 0 (0000) 1
ISSN: 1935-7524
DOI: 10.1214/154957804100000000

Auxiliary MCMC samplers for parallelisable inference in high-dimensional latent dynamical systems

Adrien Corenflos and Simo Särkkä[*]

Department of Electrical Engineering and Automation, Aalto University
e-mail: adrien.corenflos@aalto.fi*;* simo.sarkka@aalto.fi

Abstract: We study the problem of designing efficient exact MCMC algorithms for sampling from the full posterior distribution of non-linear non-Gaussian latent dynamical models. Particle Gibbs, also known as conditional sequential Monte Carlo (SMC), constitutes the *de facto* golden standard to do so but suffers from degeneracy problems when the dimension of the latent space increases. On the other hand, the routinely employed globally Gaussian-approximated (e.g., extended Kalman filtering) biased solutions are seldom employed for this same purpose even though they are more robust than their SMC counterparts. In this article, we show how, by introducing auxiliary observation variables in the model, we can both implement efficient exact Kalman-based samplers for large state-space models, as well as dramatically improve the mixing speed of particle Gibbs algorithms when the dimension of the latent space increases. We demonstrate when and how we can parallelise the auxiliary samplers along the time dimension, resulting in algorithms that scale logarithmically with the number of time steps when implemented on graphics processing units (GPUs). Both algorithms are easily tuned and can be extended to accommodate sophisticated approximation techniques. We demonstrate the improved statistical and computational performance of our auxiliary samplers compared to state-of-the-art alternatives.

Keywords and phrases: Feynman–Kac models, state-space models, particle Gibbs, Kalman filtering, parameter estimation.

Contents

[*]Both authors gratefully acknowledge funding from the Academy of Finland, project 321891 (ADAFUME), and project 321900 (PARADIST).

1. Introduction

State-space models [SSMs, see, e.g., 17, 52, 9], otherwise known as hidden Markov models, are a class of dynamic statistical models routinely employed to model phenomenons in bio-medicine, epidemiology, chemistry, or economy. For a given finite horizon $T > 0$, they are fully described by the joint distribution over their latent states and the observations:

$$\mathbb{P}(\mathrm{d}x_{0:T}\,, \mathrm{d}y_{0:T}) \coloneqq \mathbb{P}_0(\mathrm{d}x_0) \left\{ \prod_{t=0}^{T} H_t(\mathrm{d}y_t \mid x_t) \right\} \left\{ \prod_{t=1}^{T} P_t(\mathrm{d}x_t \mid x_{t-1}) \right\}. \tag{1}$$

In this formulation, $\mathbb{P}_0$ represents the initial distribution of the state X_0, while P_t and H_t represent the Markov transition and emission kernels for the states $X_t \in \mathbb{R}^{d_x}$ and observations $Y_t \in \mathbb{R}^{d_y}$, respectively.

Inference in SSMs typically recovers different meanings depending on the context: filtering is concerned with representing, sampling, or computing expectations of the quantity $\mathbb{P}(\mathrm{d}x_t \mid y_{0:t})$, where $y_{0:t} = \{y_i; i = 0, 1, \dots, t\}$; marginal smoothing is concerned with the same problems for the quantity $\mathbb{P}(\mathrm{d}x_t \mid y_{0:T})$, $t \leq T$; and pathwise smoothing is concerned with sampling or computing expectations of the quantity $\mathbb{P}(\mathrm{d}x_{0:T} \mid y_{0:T})$.

In many cases, the "true" generative model, consisting of the initial distribution $\mathbb{P}_0$, the transition kernels P_t, and emission kernels H_t, is unknown, and one needs to estimate it from the observed data. A typical way is to assume parametric forms for $\mathbb{P}_0(\mathrm{d}x_0 \mid \theta)$, $P_t(\mathrm{d}x_t \mid x_{t-1}, \theta)$, and $H_t(\mathrm{d}y_t \mid x_t, \theta)$, as well as a prior distribution $\mathbb{P}(\mathrm{d}\theta)$ for the parameters, resulting in a joint distribution

$$\begin{aligned}&\mathbb{P}(\mathrm{d}x_{0:T}, \mathrm{d}y_{0:T}, \mathrm{d}\theta) \\ &\coloneqq \mathbb{P}_0(\mathrm{d}x_0 \mid \theta) \left\{\prod_{t=0}^{T} H_t(\mathrm{d}y_t \mid x_t, \theta)\right\} \left\{\prod_{t=1}^{T} P_t(\mathrm{d}x_t \mid x_{t-1}, \theta)\right\} \mathbb{P}(\mathrm{d}\theta).\end{aligned} \tag{2}$$

Under these notations, the parameter estimation problem then consists of computing either deterministic or probabilistic estimates of the posterior distribution over the parameters $\mathbb{P}(\mathrm{d}\theta \mid y_{0:T})$. In this work, we will focus on computing probabilistic estimates for the pathwise smoothing distribution $\mathbb{P}(\mathrm{d}x_{0:T} \mid y_{0:T})$ and the joint state-parameter posterior distribution $\mathbb{P}(\mathrm{d}\theta, \mathrm{d}x_{0:T} \mid y_{0:T})$ (which marginally recovers $\mathbb{P}(\mathrm{d}\theta \mid y_{0:T})$).

Throughout the rest of the article, we will assume that all distributions and kernels considered have a density and we consequently will write

$$p(x_{0:T}, y_{0:T}, \theta) \coloneqq p_0(x_0 \mid \theta) \left\{\prod_{t=0}^{T} h_t(y_t \mid x_t, \theta)\right\} \left\{\prod_{t=1}^{T} p_t(x_t \mid x_{t-1}, \theta)\right\} p(\theta). \tag{3}$$

Moreover for notational simplicity and when this is not harmful, the dependency on the parameters θ will be implicit, and the methods will be presented for models with fixed parameters, i.e., we will write $p(x_{0:T}, y_{0:T})$ and similar for the related conditions distributions.

In this article, we consider a slight generalisation of (3), as given by the larger class of models

$$\pi(x_{0:T}) \propto g(x_0, x_1, \ldots, x_T)\, p_0(x_0) \left\{\prod_{t=1}^{T} p_t(x_t \mid x_{t-1})\right\}. \tag{4}$$

It is easy to see that this class comprises, as a special case, the pathwise smoothing distribution $p(x_{0:T} \mid y_{0:T})$ of (2) by setting $g(x_0, x_1, \ldots, x_T) = \prod_{t=0}^{T} h_t(y_t \mid x_t)$. It also recovers the class of Feynman-Kac models [see, e.g., 17]

$$\pi(x_{0:T}) \propto g_0(x_0)\, p_0(x_0) \left\{\prod_{t=1}^{T} g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1})\right\}, \tag{5}$$

for a Markovian potential function $g(x_0, x_1, \ldots, x_T) = g_0(x_0) \prod_{t=1}^{T} g_t(x_t, x_{t-1})$, which is typically the setting in which the so-called particle filtering methods apply [9, Ch. 5].

Our goal is to perform statistically and computationally efficient exact state and parameter inference in models of the form (4) and (5), in particular in the regimes of large state dimensions d_x, and number of time steps T. A particular focus will be given to parallelisation along the time-dimension, which is a natural way to scale up the inference to large T. The most popular two classes of methods for inference in SSMs are the Gaussian approximation-based methods (i.e., Kalman filters and smoothers), and the sequential Monte Carlo (SMC) based methods (i.e., particle filter and smoothers). These methods, their benefits, and their drawbacks are briefly reviewed next in Sections 1.1 and 1.2.

1.1. Gaussian approximated state-space models

Gaussian approximations rely on the fact that when the SSM at hand is linear Gaussian (LGSSM), then the filtering and marginal smoothing distributions are Gaussian as well, and their means and covariances can be computed sequentially and in closed form [see, e.g., 52, 3]. This is leveraged in Gaussian approximations to the filtering and marginal smoothing solutions of general SSMs. Typically, such approximations rely on Taylor linearisation, leading to the classical extended Kalman filtering [see, e.g., 32], or on sigma-point linearisations, first introduced in [34, 60]. In this work, we will focus on a more general posterior linearisation framework encompassing both methods and recently introduced in [23, 57].

The state of the art for these methods consists in iteratively reusing the approximated marginal smoothing distributions to refine the Gaussian approximation of the SSM at hand [4, 23, 57]. Doing so makes it possible to handle SSMs for which the reverse Markov chain representing the smoothing distribution is a slow-mixing process, that is, SSMs which have "sticky" transitions kernels and for which the filtering transition largely differs from the smoothing one. These recursive methods have been shown to be equivalent to certain minimisation programs (such as Gauss–Newton) for some given loss functions and to be (locally) convergent. For a review, we refer the reader to [56] and [52, Ch. 10 and 13-14].

Finally, it has been recently shown [51, 62, 63] that (extended/sigma-points) Kalman filtering and smoothing can be parallelised in time (PIT), resulting in a computational complexity of $\mathcal{O}(\log(T))$ on parallel hardware such as graphics processing units (GPUs), comparing to their classical $\mathcal{O}(T)$ complexity on sequential hardware. This is particularly fruitful in the iterated context, as in [62, 63], where the operation needs to be repeated until eventual convergence of the smoothing solution.

An important drawback of all the Gaussian approximation-based methods is that they (in all but the LGSSM case) result in biased estimates of the true non-Gaussian filtering as well as marginal and pathwise smoothing distributions. It is also present in the normalisation constant estimate (marginal likelihood of the observations) of the model, which makes parameter estimation procedures biased as well. This bias was the motivation for introducing Monte Carlo filtering methods [29] which we review next.

1.2. Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods [see, e.g., 9] are an alternative to Gaussian-approximated posteriors which represent the filtering and smoothing distributions using Monte Carlo samples. They proceed by propagating the trajectory sequentially via a sampling-resampling routine. Notably, SMC methods usually provide a representation of the full pathwise smoothing distribution as a byproduct of its representation of the filtering one. This representation converges when the number of samples tends to infinity [36]. However, in practice, the resulting paths degenerate for time steps $t \ll T$. This has justified the introduction of backward methods to rejuvenate the trajectories far from the endpoint [27], and their resulting convergence improvements have been studied, for example in [19], and under a more general framework, in [14].

Importantly, because particle filtering provides an unbiased likelihood estimate, it can be used to perform asymptotically exact parameter-state estimation in state-space models. A particularly useful class of methods leveraging this property are the particle Markov chain Monte Carlo (pMCMC) methods [2, 1], which are based on constructing MCMC schemes either as a Metropolis–Rosenbluth–Teller–Hastings (MRTH) algorithm [43, 31], or a Gibbs-like sampler [24]. We refer to these as pseudo-marginal and particle Gibbs (pGibbs), respectively.

These two methods sample consistently from the (joint) pathwise smoothing and parameter posterior distributions in general SSMs, but fail when the latent space dimension is large (or equivalently, when the observations are too informative compared to the prior dynamics). Backward sampling methods [61, 40] can be, to some extent, used to mitigate this problem. However, the failure is due to the inherent property that the set of particles available to describe the smoothing distribution comes from the forward filtering pass in the first place [15]. This problem can, to some extent, be mitigated by using observation-informed proposals, sometimes inherited from the approximations of Section 1.1 applied *locally* [see, e.g. 58]. Doing so, however, still fails as the dimension becomes larger.

Recently, [21] and [42, Chap. 4] independently proposed two related particle Gibbs algorithms that alleviate this issue by a generic localisation trick rather than approximation methods. [21] in particular showed that under a proper scaling of their algorithms, the methods bypass the curse of dimensionality present in classical particle MCMC methods.

Finally, it was recently shown in [11] that divide-and-conquer methods can provide consistent PIT solutions for particle smoothing and pGibbs algorithms at the cost of additional variance in the resulting estimates, providing an SMC counterpart to the algorithms of [51, 62, 63].

1.3. Motivation and Contributions

As a summary of the sections above, the Gaussian approximated smoothing solutions, whilst being more robust than SMC methods (and extensions thereof), provide coarse approximations of the full posterior and lack the unbiasedness and convergence properties of SMC. They therefore cannot be used for exact Bayesian inference in general SSMs. Furthermore, while Gaussian approximations are regularly used *locally* within particle filtering, and therefore particle MCMC [see, e.g. 58], they are seldom used to design global MCMC kernels [see, e.g., the introduction of 1, for a discussion on the difficulty of designing MCMC kernels for state-space models]. On the other hand, SMC methods allow for asymptotically exact sampling of posterior SSMs distributions but suffer from a

Algorithm 1: Auxiliary MCMC

```
  Result: An updated trajectory $x_{0:T}^{k+1}$
1 Function AUX-MCMC($x_{0:T}^k$)
2     Sample $u_{0:T}^k \sim \prod_{t=0}^T \mathcal{N}(u_t; x_t^k, \frac{\delta}{2}\Sigma_t)$
3     Sample $x_{0:T}^{k+1} \sim K(\cdot \mid x_{0:T}^k)$ // from a $\pi(x_{0:T} \mid u_{0:T}^k)$-invariant kernel
4     return $x_{0:T}^{k+1}$
```

curse of dimensionality that restricts their use to low-dimensional state spaces. This is true even when *locally* informative proposal distributions are used and is a feature of pGibbs [21, Proposition 2.2] that is inherited from particle filtering in general.

In view of this, the goal of this article is to develop general methods that allow performing statistically and computationally efficient inference in large-dimensional latent dynamical systems. To do so, we will consider two routes, which, at first, may seem unrelated but happen to be two specific instances of the same algorithm. The first one consists in designing an MCMC kernel based on SSM-specific Gaussian approximations and linearisations, while the second one relies on using localisation and linearisation techniques in a modified particle Gibbs algorithm. In both cases, we will pay particular attention to opportunities for parallelising the method on GPUs, specifically along the time dimension.

These two approaches are respectively based on (i) [54] design auxiliary MCMC gradient-based inference in high-dimensional latent Gaussian models, which we review in Section 2.1; (ii) [21] reduce the curse of dimensionality in pGibbs methods by using localisation and exchangeable proposals within the underlying conditional SMC algorithm. At heart, both methods — the former explicitly, the latter implicitly, as is explained in Section 4.2 — consist in augmenting the target distribution π with an auxiliary variable: using our SSM notation, $\pi(\mathrm{d}x_{0:T}, \mathrm{d}u_{0:T}) = \pi(\mathrm{d}x_{0:T}) \prod_{t=0}^T \mathcal{N}(\mathrm{d}u_t; x_t, \delta\Sigma_t)$ which marginally recovers the original distribution $\pi(x_{0:T})$. The inference is then performed in two steps summarised in Algorithm 1 in which the choice of the kernel used in step 3 is, in our specific context, either a custom MRTH kernel [54] or a pGibbs kernel for a modified model [21].

This perspective motivates our following contributions:

1. In Section 2, we show that, in the case of generalised Feynman–Kac models (4) with Gaussian dynamics, the auxiliary proposals of [54] recover the posterior distribution of an auxiliary LGSSM. We leverage this to reduce their time and space complexity to $\mathcal{O}(T)$ rather than $\mathcal{O}(T^3)$. We then extended this to non-Gaussian prior dynamics using local Gaussian approximants.
2. In Section 3, we introduce parallel-in-time samplers for LGSSM full distributions based on a prefix-sum implementation akin to [51], resulting in an overall $\mathcal{O}(\log T)$ MCMC algorithm on parallel hardware.
3. In Section 4, we describe how [21] is an instance of the auxiliary sampler.

This allows us to introduce novel auxiliary particle Gibbs methods, overperforming their state-of-the-art method. We also describe when these can be parallelised efficiently on GPU along the time dimension.

4. In Section 5, we apply the proposed methods to perform inference on a multidimensional stochastic volatility model from the SMC literature, a high-dimensional spatio-temporal model with fat-tailed observations taken from [13], and on a joint state-parameter inference problem for a non-linear stochastic differential equation. In all cases, special attention is paid to understanding their statistical as well as computational trade-offs.

2. Auxiliary Kalman samplers

In this section, we first review the auxiliary samplers of [54] for latent Gaussian models $\pi(x) \propto \exp(f(x))\,\mathcal{N}(x; 0, C)$. We then show how, in the case of latent Gaussian dynamics models, they can be specialised so as to reduce the time and memory complexity to linear in the number of time steps rather than quadratic. We then show how linearisation methods can be used to extend the method to non-linear dynamics that can be approximated by Gaussian ones well enough.

2.1. Auxiliary gradient-based samplers

Auxiliary gradient-based methods were introduced in [54] as a way to construct posterior-informed proposals in MCMC samplers for Gaussian latent models with a density $\pi(x) \propto \exp(f(x))\,\mathcal{N}(x; 0, C)$[1], where $x \in \mathbb{R}^{d_x}$. They were shown to outperform classical pre-conditioned (prior-informed) and gradient-based (likelihood-informed) MCMC samplers, such as the pre-conditioned Crank–Nicholson [12], or manifold MCMC [26] samplers for latent Gaussian models. This impressive performance is both due to its better representation of the covariance of the posterior distribution [54, Section 3.4], as well as its computational advantage compared to classical methods, resulting in an improved effective sample size per unit of time even when the effective sample size itself was lesser [54, Table 2].

Auxiliary gradient-based samplers rely on augmenting the target π with an auxiliary variable u:

$$\pi(x, u) \propto \exp(f(x))\,\mathcal{N}(x; 0, C)\,\mathcal{N}\left(u; x, \frac{\delta}{2} I\right), \tag{6}$$

where $\delta > 0$ is a step size, so that the marginal of $\pi(x, u)$ is $\pi(x)$. Auxiliary samplers then proceed by linearising f around the current state x of the Markov chain to obtain a Gaussian proposal distribution

$$\begin{aligned} q(y \mid x, u) &\propto \exp\big(\nabla f(x)^\top y\big)\,\mathcal{N}(y; 0, C)\,\mathcal{N}\left(u; y, \frac{\delta}{2} I\right) \\ &= \mathcal{N}\left(y; \frac{2}{\delta} A \left(u + \frac{\delta}{2} \nabla f(x)\right), A\right), \end{aligned} \tag{7}$$

[1]As well as, under a trivial change of variables, for models with non-zero prior mean.

where $A = \frac{\delta}{2}(C + \frac{\delta}{2}I)^{-1}C = (C^{-1} + \frac{2}{\delta}I)^{-1}$. Sampling from $\pi(x, u)$ (and therefore from $\pi(x)$ by discarding the intermediate auxiliary steps) is then done via Hastings-within-Gibbs [45]:

1. Sample $u \mid x \sim \mathcal{N}(u; x, \frac{\delta}{2}I)$.
2. Propose $y \sim q(\cdot \mid x, u)$ targeting $\pi(\cdot \mid u) \propto \pi(\cdot, u)$, and accept the move with the corresponding acceptance probability.

A more efficient counterpart of this, targeting $\pi(x)$ directly, can be given by integrating the proposal distribution (7) with respect to $\mathcal{N}(u; x, \frac{\delta}{2}I)$:

$$q(y \mid x) = \mathcal{N}\left(y; \frac{2}{\delta}A\left(x + \frac{\delta}{2}\nabla f(x)\right), \frac{2}{\delta}A^2 + A\right). \tag{8}$$

This marginalised version skips the intermediate sampling step of the auxiliary variable, and is provably better – both empirically and in terms of Peskun ordering [50, 53, 38] – than its auxiliary version, resulting in step sizes δ roughly twice larger [see Tables 1, 2, and 3 in 54] for the same acceptance rate, at virtually no additional computational complexity.

A crucial property of both these instances of the auxiliary sampler is that for all $\delta > 0$, of the matrices A and C share the same eigenspace [54, Section 3.3]. This ensures that after an initial spectral decomposition of C, changing the value of δ can be done at a negligible cost compared to the actual sampling process itself, making the algorithm easy to tune for a given target acceptance rate.

However, when C depends on a parameter θ, changing θ will not keep the eigenspace invariant. This means that when using either of these samplers within a Hastings-within-Gibbs routine targeting a joint model

$$\pi(x, \theta) \propto \exp(f(x))\,\mathcal{N}(x; 0, C_\theta)\,p(\theta), \tag{9}$$

the spectral decomposition of C_θ has to be recomputed every time the value of θ changes. This is computationally prohibitive for large dimensional x. This, however, can be mitigated thanks to the following observation [54]: under a reparametrisation of u, which corresponds to considering the augmented target

$$\pi(x, u) \propto \exp(f(x))\,\mathcal{N}(x; 0, C)\,\mathcal{N}\left(u; x + \frac{\delta}{2}\nabla f(x), \frac{\delta}{2}I\right), \tag{10}$$

rather than (6), the proposal distribution $q(y \mid x, u)$ can be made independent of the current state of the chain x. This allows doing joint updates of x and θ in parametric models, rather than using Gibbs steps to sample x conditionally on θ, and θ conditionally on x, thereby improving the mixing rate of the sampled Markov chain. This improvement, however, does not change the need for updating the spectral decomposition of C_θ and comes at the price of lower statistical efficiency than the non-reparametrised version.

2.2. Auxiliary Kalman samplers

The distribution $\pi(x) \propto \exp(f(x))\mathcal{N}(x; 0, C)$ covers latent Gaussian models in general, and in particular covers models with latent Gaussian dynamics[2]:

$$\pi(x_{0:T}) \propto g(x_0, \dots, x_T)\mathcal{N}(x_0; m_0, P_0)\prod_{t=1}^{T}\mathcal{N}(x_t; F_{t-1}\,x_{t-1} + b_{t-1}, Q_{t-1}). \quad (11)$$

However, directly treating these as latent Gaussian models with the methods of [54] would incur a computational complexity of $\mathcal{O}(T^2 d_x^2)$, with an initial pre-processing step that scales as $\mathcal{O}(T^3 d_x^3)$, and a memory cost of $\mathcal{O}(T^2 d_x^2)$ corresponding to the size of the underlying covariance matrix C. This is true even though the inverse of C is sparse [see, e.g., 3, Chap. 3] due to the need to compute the eigen-decomposition of either C or C^{-1} [54, Supplementary material]. Instead of doing this, it is possible, in the case of a model like (11), to preserve the Markovian structure of the model and formulate the auxiliary sampler as an LGSSM, which can then be used more efficiently.

In order to do so, we emulate [54] and consider the augmented target distribution

$$\pi(x_{0:T}, u_{0:T}) \propto \pi(x_{0:T})\prod_{t=0}^{T}\mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right), \quad (12)$$

where $\delta > 0$ and, for all $t = 0, \dots, T$, Σ_t is some positive definite matrix in $\mathbb{R}^{d_x \times d_x}$. Note that when $\Sigma_t = I$ is the identity matrix for all t, this recovers the proposal (6).

Let us define γ via $\exp(\gamma(x_0, x_1, \dots, x_T)) := g(x_0, x_1, \dots, x_T)$, and linearise it around the previously sampled trajectory $x_{0:T}$, $\gamma(z_{0:T}) \approx \gamma(x_{0:T}) + \langle v_{0:T}, z_{0:T} - x_{0:T}\rangle$, where $v_t = \frac{\partial \gamma}{\partial x_t}(x_{0:T})$ for all t, and $\langle a_{0:T}, b_{0:T}\rangle$ denotes the sum of inner products $\sum_{t=0}^{T}\langle a_t, b_t\rangle$. Under these notations, we can define the auxiliary proposal

$$\begin{aligned} q(z_{0:T} \mid u_{0:T}, x_{0:T}) \propto \mathcal{N}(z_0; m_0, P_0) &\left\{\prod_{t=1}^{T}\mathcal{N}(z_t; F_{t-1}z_{t-1} + b_{t-1}, Q_{t-1})\right\} \\ &\left\{\prod_{t=0}^{T}\mathcal{N}\left(u_t + \frac{\delta}{2}\Sigma_t v_t; z_t, \frac{\delta}{2}\Sigma_t\right)\right\}, \end{aligned} \quad (13)$$

which corresponds to the pathwise smoothing distribution of an LGSSM with unchanged dynamics compared to (11), and observations given by $u_t + \frac{\delta_t}{2}\Sigma_t v_t$ for an observation model $\mathcal{N}\left(\cdot; z_t, \frac{\delta}{2}\Sigma_t\right)$, $t = 0, 1, \dots, T$. Sampling from this distribution, and evaluating its likelihood can be done using Kalman primitives [see, e.g. 52, Ch. 6 and Ch. 12] in $\mathcal{O}(T)$ steps. In fact, this representation is crucial

[2] This was in fact explicitly used in [9, Chap. 15], where the authors successfully apply [54] to a one-dimensional stochastic volatility model with latent Gaussian dynamics. The fact that the sampler corresponded to a linear Gaussian state-space model was, however, not noted by the authors.

to reduce the memory requirements to linear in T as well as the computational complexity from cubic to linear or even logarithmic in T for parallel hardware. We come back to this last point in Section 3.

To summarise, sampling from $\pi(x_{0:T}, u_{0:T})$ is then done via Hastings-within-Gibbs [45]: (i) sample $u^k_{0:T} \mid x^k_{0:T} \sim \prod_{t=0}^T \mathcal{N}(u_t; x^k_t, \frac{\delta}{2} I)$, (ii) propose $x^*_{0:T} \sim q(\cdot \mid x^k_{0:T}, u^k_{0:T})$ targeting $\pi(\cdot \mid u^k_{0:T}) \propto \pi(\cdot, u^k_{0:T})$, and accept the move with the corresponding acceptance probability.

We insist that this proposal is statistically equivalent to the auxiliary method of [54] for a choice of constant $\Sigma_t = I$, but exhibits better computational complexity. When the potentials are separable, as is the case for state-space models. We can easily use second-order approximations. Indeed, when $g(x_{0:T}) = \prod_{t=0}^T g_t(x_t)$, or equivalently, when $\gamma(x_{0:T}) = \sum_{t=0}^T \gamma_t(x_t)$, we can write

$$\gamma(z_{0:T}) \approx \gamma(x_{0:T}) + \langle v_{0:T}, z_{0:T} - x_{0:T} \rangle + \frac{1}{2} \sum_{t=0}^T (z_t - x_t)^\top \Lambda_t (z_t - x_t), \tag{14}$$

where Λ_t is the Hessian matrix of γ_t evaluated at x_t. By rearranging the terms, we can derive the resulting proposal distribution as

$$\begin{aligned} q(z_{0:T} \mid u_{0:T}, x_{0:T}) \propto \mathcal{N}(z_0; m_0, P_0) &\left\{ \prod_{t=1}^T \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1}) \right\} \\ &\left\{ \prod_{t=0}^T \mathcal{N}\left(\omega_t; z_t, \Omega_t\right) \right\}, \end{aligned} \tag{15}$$

with $\Omega_t = \left(\frac{2}{\delta}\Sigma_t^{-1} - \Lambda_t\right)^{-1}$ and $\omega_t = \Omega_t \left(\frac{2}{\delta}\Sigma_t^{-1} u_t + v_t - \Lambda_t x_t\right)$. This proposal is well defined as an LGSSM as soon as δ is small enough.

Finally, when the dynamics are not Gaussian, it is often possible to transform the model at hand into an equivalent representation of $q_{0:T}$ with Gaussian dynamics by setting

$$\begin{aligned} &p_0(x_0) \leftarrow \mathcal{N}(x_0; m_0, P_0), \quad p_t(x_t \mid x_{t-1}) \leftarrow \mathcal{N}(x_t; F_{t-1} x_{t-1} + b_{t-1}, Q_{t-1}) \\ &g(x_{0:T}) \leftarrow g(x_{0:T}) \frac{p_0(x_0)}{\mathcal{N}(x_0; m_0, P_0)} \prod_{t=1}^T \frac{p_t(x_t \mid x_{t-1})}{\mathcal{N}(x_t; F_{t-1} x_{t-1} + b_{t-1}, Q_{t-1})}. \end{aligned} \tag{16}$$

While this is sometimes a natural thing to do, it can also happen that there is no natural way to make such a Gaussian appear in the model. This justifies the need for introducing a new class of auxiliary samplers.

2.3. New auxiliary samplers for models with tractable conditional moments

In Section 2.2, we have made an explicit link between the auxiliary samplers of [54] and Kalman filtering when the latent model has Gaussian dynamics. This linearity of the latent model corresponds to the assumption of linear Gaussian dynamics in the case of state-space models. This is a rather strong modelling assumption that is not easily verified, or enforced, in practice. In this section, we present an approach which uses local approximations of the dynamics model by conditional Gaussian transitions using statistical linear regression.

In order to present the method in its most general form, we consider the case of state-space models, where the potential function $g(x_{0:T})$ corresponds to a product of observation models $\prod_{t=0}^{T} h_t(y_t \mid x_t)$ so that we can also form the Gaussian approximation to the potential itself. Following [57], we suppose that the first two conditional moments

$$\begin{aligned} m^X(x_{t-1}) &\coloneqq \mathbb{E}[X_t \mid X_{t-1} = x_{t-1}], & m^Y(x_t) &\coloneqq \mathbb{E}[Y_t \mid X_t = x_t], \\ V^X(x_{t-1}) &\coloneqq \mathbb{V}[X_t \mid X_{t-1} = x_{t-1}], & V^Y(x_t) &\coloneqq \mathbb{V}[Y_t \mid X_t = x_t], \end{aligned} \tag{17}$$

of, respectively, the transitions and observation models appearing in (1) can easily be either computed in closed form or approximated well enough. Similarly, we suppose that the two first moments m_0 and P_0 of p_0 are known at least approximately.

Similarly as in Section 2.2, we start by considering an augmented target distribution

$$\begin{aligned} p(x_{0:T}, y_{0:T}, u_{0:T}) \coloneqq p(x_0) &\left\{ \prod_{t=0}^{T} h_t(y_t \mid x_t)\, \mathcal{N}(u_t; x_t, \frac{\delta}{2}\Sigma_t) \right\} \\ &\left\{ \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}) \right\}, \end{aligned} \tag{18}$$

where $\delta > 0$, and for all t, Σ_t is a positive definite matrix.

In order to form a proposal distribution $q(x_{0:T} \mid u_{0:T}, y_{0:T})$ for $p(x_{0:T} \mid y_{0:T}, u_{0:T})$, we linearise the state-space model (1) around the trajectory at hand. Let $x_{0:T} \in \mathbb{R}^{T \times d_x}$ and $u_{0:T} \in \mathbb{R}^{T \times d_x}$ be the current states of the auxiliary Markov chain, and let $\Gamma_{0:T}$ be a set of reference covariance matrices in $\mathbb{R}^{T \times d_x \times d_x}$, by which we mean that $\Gamma_t \in \mathbb{R}^{d_x \times d_x}$ needs to be positive definite for all t. We can apply the generalised statistical linear regression (GSLR) framework of [57] for the reference random variables $\zeta_t \sim \mathcal{N}(x_t, \Gamma_t)$, $t = 0, \dots, T$ to derive Gaussian approximations of the transition and observation models as follows:

$$\begin{aligned} p_t(z_t \mid z_{t-1}) &\approx \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1}), \\ h_t(y_t \mid z_t) &\approx \mathcal{N}(y_t; H_t z_t + c_t, R_t), \end{aligned} \tag{19}$$

with,

$$\begin{aligned} F_{t-1} &= C^X_{t-1} \Gamma^{-1}_{t-1}, & H_t &= C^Y_t \Gamma^{-1}_t, \\ b_{t-1} &= \mu^X_{t-1} - F_{t-1} x_{t-1}, & c_t &= \mu^Y_t - H_t x_t, \\ Q_{t-1} &= S^X_{t-1} - F_{t-1} \Gamma^{-1}_{t-1} F^\top_{t-1}, & R_t &= S^Y_t - H_t \Gamma^{-1}_t H^\top_t, \end{aligned} \tag{20}$$

and where, for the sake of readability, we do not notationally emphasise the dependency on x and Γ. These Gaussian approximations are known to minimise a forward KL divergence with respect to the transition and observation models for the Gaussian variational family. The coefficients appearing in (20) are in turn given by the general formulae

$$\begin{aligned} C^X_{t-1} &= \mathbb{C}\left[m^X(\zeta_{t-1}), \zeta_{t-1}\right], & C^Y_t &= \mathbb{C}\left[m^Y(\zeta_t), \zeta_t\right], \\ \mu^X_{t-1} &= \mathbb{E}\left[m^X(\zeta_{t-1})\right], & \mu^Y_t &= \mathbb{E}\left[m^Y(\zeta_t)\right], \\ S^X_{t-1} &= \mathbb{E}\left[V^X(\zeta_{t-1})\right], & S^Y_t &= \mathbb{E}\left[V^Y(\zeta_t)\right]. \end{aligned} \tag{21}$$

Clearly, the quantities in (21) are not typically available in closed-form, and we instead need to resort to further approximations. Such approximations are given by, for example, Taylor series expansions or sigma-point methods, such as Gauss–Hermite or unscented methods [see, e.g., 52, Ch. 8].

Example 2.1. The first-order Taylor approximation to C^X_{t-1} can be obtained by $m^X(\zeta_{t-1}) \approx m^X(x_{t-1}) + \nabla m^X(x_{t-1})(\zeta_{t-1} - x_{t-1})$, so that we get $C^X_{t-1} \approx \nabla m^X(x_{t-1})\Gamma_{t-1}$, and finally $F_{t-1} \approx \nabla m^X(x_{t-1})$. When the model at hand has additive Gaussian noise: $X_t = f(x_{t-1}) + \epsilon_{t-1}$, this recovers the well-known extended Kalman filter linearisation. Similarly, all the other linearisation parameters will be independent of the choice of Γ_t, $t = 0, \dots, T$. This property however does not hold when using second-order Taylor or sigma-points approximations for the integrals appearing in (21), and the choice of Γ_t will then impact the performance of the algorithm [see, e.g., 52, Ch. 8-10, for the role of the reference covariance in classical Gaussian-approximated filtering and smoothing].

These linear approximations, together with the known (or approximated) first two moments m_0 and P_0 of $\mathbb{P}_0$, can then be used to form a proposal distribution defined as an auxiliary LGSSM smoothing distribution with density

$$\begin{aligned} q(z_{0:T} \mid u_{0:T}, x_{0:T}, y_{0:T}) \propto \mathcal{N}(z_0; m_0, P_0) & \\ \left\{\prod_{t=0}^{T} \mathcal{N}(y_t; H_t z_t + c_t, R_t)\, \mathcal{N}\left(u_t; z_t, \frac{\delta}{2}\Sigma_t\right)\right\}& \\ \left\{\prod_{t=1}^{T} \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1})\right\}&. \end{aligned} \tag{22}$$

This proposal distribution is then included as part of a Metropolis–Rosenbluth–Teller–Hastings (MRTH) acceptance-rejection step. The resulting sampler then corresponds to Algorithm 2. Evaluating the augmented density (12) appearing in the acceptance ratio of the MRTH algorithm, line 9, is easily done. Therefore, to effectively implement the steps above we only need to understand how to sample from the smoothing distribution of the LGSSM at hand, and compute the corresponding smoothing density $\frac{q(x^*_{0:T}, y_{0:T}, u_{0:T} \mid x_{0:T})}{q(y_{0:T}, u_{0:T} \mid x_{0:T})}$. This is readily achieved by applying sequential Kalman filtering equations to compute the filtering densities as well as the marginal likelihood of the true and auxiliary observations [see, e.g., 52, Chap. 6], and then sampling from the pathwise smoothing distribution

```
Algorithm 2: General Auxiliary Kalman sampler
   Result: An updated trajectory $x_{0:T}$
1  Function AuxKalmanSampler($x_{0:T}$)
       // Generate the auxiliary observations
2      for $t = 0, 1, \dots, T$ sample $u_t \mid x_t \sim \mathcal{N}(\cdot; x_t, \frac{\delta}{2}\Sigma_t)$
       // Proposal part
3      for $t = 0 \dots, T$ do
4          if $t > 0$ then
5              Form an approximation $\mathcal{N}(z_t; F^*_{t-1} z_{t-1} + b^*_{t-1}, Q^*_{t-1}) \approx p_t(z_t \mid z_{t-1})$
                around $x_{t-1}$
6          Form an approximation $\mathcal{N}(y_t; H^*_t z_t + c^*_t, R^*_t) \approx p_t(y_t \mid z_t)$, around $x_t$
7      Sample $x^*_{0:T} \sim q(\cdot \mid y_{0:T}, u_{0:T}, x_{0:T})$ and compute $L^* = \frac{q(x^*_{0:T}, y_{0:T}, u_{0:T} \mid x_{0:T})}{q(y_{0:T}, u_{0:T} \mid x_{0:T})}$
       // MRTH step
8      Form the reversed proposal $q^*(x_{0:T} \mid y_{0:T}, u_{0:T}, x^*_{0:T})$ following steps 5 and 6
        around $x^*_{0:T}$ and compute $L = \frac{q^*(x_{0:T}, y_{0:T}, u_{0:T} \mid x^*_{0:T})}{q^*(y_{0:T}, u_{0:T} \mid x^*_{0:T})}$
9      With probability $\min\left(1, \frac{p(x^*_{0:T}, y_{0:T}, u_{0:T}) L}{p(x_{0:T}, y_{0:T}, u_{0:T}) L^*}\right)$, set $x_{0:T} = x^*_{0:T}$
10     return $x_{0:T}$
```

backwards [16]. We are therefore able to achieve an overall computational complexity of $\mathcal{O}(T \times d_x^3)$. This is to be compared with the $\mathcal{O}(T^2 \times d_x^2)$ complexity of using the latent Gaussian samplers directly [54, Section 3.3] as well as with the $\mathcal{O}(T^3 \times d_x^3)$ cost of forming their samplers in the first place. In particular, when the model at hand depends on a parameter θ updating it will not increase the cost of the $x_{0:T}$ sampling step, as opposed to the algorithms of [54]. This property is crucial for deriving efficient samplers for the joint distribution of the parameters and latent states.

We end this section by noting that, while we assumed that the model at hand was a state-space model for consistency and ease of exposition, the method developed in this section can also be applied to the generalised Feynman–Kac model (4). To do so, it suffices to apply the first-order linearisation of Section 2.2 to the likelihood term, and, independently, the GSLR approach of this section to the latent dynamics model. Similarly, the linearisation method used for the observation and transition need not be the same one, and the choice thereof is fully left to the user; for instance, if the likelihood is separable, we can use a Laplace approximation for the individual terms g_t, but still an extended linearisation for the transitions.

3. Parallel-in-time pathwise sampling for LGSSMs

While sampling from the pathwise smoothing distribution of an LGSSM is rather easy, it naturally has a computational complexity of $\mathcal{O}(T)$ in the number of time steps. However, this can be improved by parallelisation. In this section, we present two different methods to sample from an LGSSM in logarithmic $\mathcal{O}(\log(T))$ time. The methods are based on similar ideas as the parallel smoothing methods presented in [51]. One of the methods uses a computational primitive called prefix-sum or associative scan [7], which generalises cumulative sums to other (associative) operators than addition, while the second one relies on a divide-and-conquer mechanism. It is worth noting that we expect the first to perform better due to its lower memory requirements, but the second one can be used more easily in distributed settings. We note that both methods are *statistically equivalent*, i.e., they both sample from the same distribution, albeit in a different manner.

Putting aside the notations of the rest of the article, in this section we consider an LGSSM given by its joint distribution $q(x_{0:T}, y_{0:T})$ over the states and observations such that

$$\begin{aligned} &X_0 \sim \mathcal{N}(m_0, P_0), \quad X_t = F_{t-1}X_{t-1} + b_{t-1} + e_{t-1}, \quad t > 0 \\ &Y_t = H_t X_t + c_t + r_t, \quad t \geq 0, \end{aligned} \tag{23}$$

with $e_t \sim \mathcal{N}(0, Q_t)$ and $r_t \sim \mathcal{N}(0, R_t)$ for all $t \geq 0$.

3.1. Prefix-sum sampling for LGSSMs

We now describe how to sample from $q(\cdot \mid y_{0:T})$ using similar methods as in [51, 62, 63]. Given that both the target of the log-likelihood (as a sum of T independent terms) and the marginal log-likelihood of the LGSSM approximation [51] can be computed in $\mathcal{O}(\log(T))$ on parallel hardware, this is the only missing piece for implementing a parallel-in-time version of our auxiliary Kalman samplers. While several different formulations [see, e.g., 20, 22] may be employed to do so, we here focus on the forward filtering backward sampling (FFBS) [22] approach. We can compute the filtering distributions $q(x_t \mid y_{0:t}) = \mathcal{N}(x_t; m_t, P_t)$ for (23) in parallel using the methods of [51]. Furthermore, we know [22, Proposition 1] that

$$\begin{aligned} q(x_T \mid y_{0:T}) &= \mathcal{N}(x_T; m_T, P_T) \\ q(x_t \mid x_{t+1}, y_{0:t}) &= \mathcal{N}\left(x_t; m_t + G_t x_{t+1} - F_t m_t - b_t, \Sigma_t\right), \quad t < T, \end{aligned} \tag{24}$$

where $G_t = P_t F_t^\top \left(F_t P_t F_t^\top + Q_t\right)^{-1}$ and $\Sigma_t = P_t - G_t(F_t P_t F_t^\top + Q_t)G_t^\top$ for all $t < T$.

We can furthermore rearrange the terms to express $\hat{X}_t \sim q(x_t \mid \hat{X}_{t+1}, y_{0:t})$ recursively as $\hat{X}_t = G_t \hat{X}_{t+1} + \nu_t$, where all the ν_t's are independent, and $\nu_t \sim \mathcal{N}(m_t - G_t(F_t m_t + b_t), \Sigma_t)$ for all $t < T$. We also let $G_T = 0$, so that we can then define $\nu_T \sim \mathcal{N}(m_T, P_T)$. Because the means and covariances of the ν_t's only depend on the LGSSM coefficients and the filtering means and covariances at time t, they can be sampled fully in parallel. To sample from $q(x_{0:T} \mid y_{0:T})$ we then need to apply the recursion to the pre-sampled sequence $U_t \sim \nu_t$, $t = 0, \dots, T$. However, the recursive dependency in (24) is not directly parallelisable, and we instead need to rephrase it in terms of an associative operator, which will allow us to use prefix-sum primitives [7]. Thankfully, this is readily done by considering the $\circ$ operator defined as follows

$$(G_{ij}, U_{ij}) = (G_i, U_i) \circ (G_j, U_j), \quad \text{where } G_{ij} = G_i G_j, \text{ and } U_{ij} = G_i U_j + U_i. \tag{25}$$

Proposition 3.1. *The backward prefix-sum of operator $\circ$ applied to the sequence (G_t, U_t), $t = 0, \dots, T$, recovers the pathwise smoothing distribution $q(x_{0:T} \mid y_{0:T})$, that is, if $(\tilde{G}_t, \tilde{U}_t) = (G_t, U_t) \circ \dots \circ (G_T, U_T)$, then $(\tilde{U}_0, \dots, \tilde{U}_T)$ is distributed according to $q(x_{0:T} \mid y_{0:T})$.*

Proof. The operator $\circ$ defined in (25) is clearly associative. We prove that its result corresponds to sampling from the pathwise smoothing distribution by reversed induction: suppose that $(\tilde{U}_t, \dots, \tilde{U}_T)$ is distributed according to $q(x_{t:T} \mid y_{0:T})$, then $\tilde{U}_{t-1} = G_{t-1}\tilde{U}_t + U_{t-1}$, which is distributed according to $q(x_{t-1} \mid \tilde{U}_t, y_{0:t-1})$ as discussed before, so that $(\tilde{U}_{t-1}, \dots, \tilde{U}_T)$ is distributed according to $q(x_{t-1:T} \mid y_{0:T})$. The initial case follows from the definition of U_T. □

To summarise, in order to perform prefix-sum sampling of LGSSMs, it suffices to use the parallel-in-time Kalman filtering method of [51] to compute the filtering means and covariances m_t, P_t, $t = 0, \dots, T$, then form all the elements G_t and sample U_t fully in parallel, and finally, apply the prefix-sum primitive [7] to $(G_t, U_t)_{t=0}^T$ with the associative operator $\circ$.

3.2. Divide-and-conquer sampling for LGSSMs

We now present a divide-and-conquer alternative to Section 3.1 for PIT sampling from the pathwise smoothing distribution of LGSSMs. The method is based on recursively finding tractable Gaussian expressions for the "bridging" $q(x_l \mid y_{0:T}, x_k, x_m)$, $0 \leq k < l < m \leq T$ of the smoothing distribution. This will allow us to derive a tree-based divide-and-conquer sampling mechanism for the pathwise smoothing distribution $q(x_{0:T} \mid y_{0:T})$.

Suppose we are given the LGSSM (23), then given three indices $0 \leq k < l < m \leq T$. We have

$$q(x_l \mid y_{0:T}, x_k, x_m) = \frac{q(x_k, x_l \mid y_{0:T}, x_m)}{q(x_k \mid y_{0:T}, x_m)} \tag{26}$$

with, furthermore,

$$q(x_k, x_l \mid y_{0:T}, x_m) = q(x_k \mid y_{0:T}, x_l) q(x_l \mid y_{0:T}, x_m) \tag{27}$$

thanks the to Markovian structure of the model. Now let $q(x_k \mid y_{0:T}, x_l)$ and $q(x_l \mid y_{0:T}, x_m)$ be given by

$$\begin{aligned} q(x_k \mid y_{0:T}, x_l) &= \mathcal{N}(x_k; E_{k:l} x_l + g_{k:l}, L_{k:l}) \\ q(x_l \mid y_{0:T}, x_m) &= \mathcal{N}(x_k; E_{l:m} x_m + g_{l:m}, L_{l:m}) \end{aligned} \tag{28}$$

for some parameters $E_{k:l}$, $g_{k:l}$, $L_{k:l}$, $E_{l:m}$, $g_{l:m}$, and $L_{l:m}$ that we will define below. Then we can write

$$q(x_k, x_l \mid y_{0:T}, x_m) = \mathcal{N}\left(\begin{pmatrix} x_l \\ x_k \end{pmatrix}; \xi_{k:l}, \Xi_{k:l}\right), \tag{29}$$

for

$$\xi_{k:l} := \begin{pmatrix} E_{l:m} x_m + g_{l:m} \\ E_{k:l} E_{l:m} x_m + E_{k:l} g_{l:m} + g_{k:l} \end{pmatrix} \tag{30}$$

and

$$\Xi_{k:l} := \begin{pmatrix} L_{l:m} & L_{l:m} E_{k:l}^\top \\ E_{k:l} L_{l:m} & E_{k:l} L_{l:m} E_{k:l}^\top + L_{k:l} \end{pmatrix}. \tag{31}$$

This gives both the marginal distribution of x_k

$$\begin{aligned} q(x_k \mid y_{0:T}, x_m) &= \mathcal{N}(x_k; E_{k:l} E_{l:m} x_m + E_{k:l} g_{l:m} + g_{k:l}, E_{k:l} L_{l:m} E_{k:l}^\top + L_{k:l}) \\ &= \mathcal{N}(x_k; E_{k:m} x_m + g_{k:m}, L_{k:m}), \end{aligned} \tag{32}$$

where

$$E_{k:m} = E_{k:l} E_{l:m}, \quad g_{k:m} = E_{k:l} g_{l:m} + g_{k:l}, \quad L_{k:m} = E_{k:l} L_{l:m} E_{k:l}^\top + L_{k:l}, \tag{33}$$

and (after simplification for (33)) the conditional distribution of x_l

$$q(x_l \mid y_{0:T}, x_k, x_m) = \mathcal{N}(x_l; G_{k:l:m} x_k + \Gamma_{k:l:m} x_m + w_{k:l:m}, V_{k:l:m}), \tag{34}$$

for

$$\begin{aligned} G_{k:l:m} &= L_{l:m} E_{k:l}^\top L_{k:m}^{-1}, & w_{k:l:m} &= g_{l:m} - G_{k:l:m} g_{k:m}, \\ \Gamma_{k:l:m} &= E_{l:m} - G_{k:l:m} E_{k:m}, & V_{k:l:m} &= L_{l:m} - G_{k:l:m} L_{k:m} G_{k:l:m}^\top. \end{aligned} \tag{35}$$

This construction provides a recursive tree structure for sampling from $q(x_{0:T} \mid y_{0:T})$ which can be initialised by

$$q(x_t \mid y_{0:T}, x_{t+1}) = \mathcal{N}(x_t; E_{t:t+1} x_{t+1} + g_{t:t+1}, L_{t:t+1}), \tag{36}$$

with

$$\begin{aligned} E_{t:t+1} &= P_t F_t^\top (F_t P_t F_t^\top + Q_t)^{-1}, \\ g_{t:t+1} &= m_t - E_{t:t+1}(F_t m_t + b_t), \\ L_{t:t+1} &= P_t - E_{t:t+1} F_t P_t, \end{aligned} \tag{37}$$

and $q(x_T \mid y_{0:T}) = \mathcal{N}(x_T; m_T, P_T)$. Finally, noting that

$$q(x_0 \mid y_{0:T}, x_T) = \mathcal{N}(x_0; E_{0:T} m_T + g_{0:T}, L_{0:T}), \tag{38}$$

we can combine these identities to form a divide-and-conquer algorithm.

To summarise, in order to perform divide-and-conquer sampling of LGSSMs, it suffices, as in Section 3.1, to use the parallel-in-time Kalman filtering method of [51] to compute the filtering means and covariances m_t, P_t, $t = 0, \dots, T$. After this, we can recursively compute the tree of elements $E_{k:m}, g_{k:m}, L_{k:m}$, together with the auxiliary variables $G_{k:l:m}, w_{k:l:m}, \Gamma_{k:l:m}, V_{k:l:m}$, starting from $E_{t:t+1}, g_{t:t+1}, L_{t:t+1}$, for $t = 0, 1, \dots, T-1$, then $E_{t-1:t+1}, g_{t-1:t+1}, L_{t-1:t+1}$, for $t = 1, 3, 5, \dots, 2\lfloor (T-1)/2 \rfloor + 1$, etc. Once this has been done, we can then sample from $q(x_T \mid y_{0:T})$, then from $q(x_0 \mid y_{0:T}, x_T)$, then $x_{\lfloor T/2 \rfloor}$ conditionally on x_0 and x_T, then, in parallel $x_{\lfloor T/4 \rfloor}$ and $x_{\lfloor 3T/4 \rfloor}$, conditionally on the rest, and continue until all have been sampled.

4. Auxiliary particle Gibbs samplers

We have so far been concerned with MCMC algorithms using Kalman primitives to sample from an LGSSM proposal distribution designed as a local approximation of the target model at hand. This method, while expected to work particularly well when the prior is almost Gaussian and the potential relatively non-informative, presents at least two limitations: (i) it accepts or rejects a full trajectory at once so that an unfortunate choice for a single time step would result in rejecting the full proposal, thereby hindering the progress of the Markov chain; (ii) it requires that the full model be *de facto* differentiable or has fully tractable moments, preventing its use in, for example, bounded state-spaces.

We first quickly recall the basic particle Gibbs algorithm, after which we show how it can be used to sample from an auxiliary target resembling (12). A general method is then presented, after which we show how additional information may be used to improve the sampler.

4.1. SMC and particle Gibbs algorithms

Particle Gibbs algorithms are Gibbs-like MCMC samplers that target the posterior distribution of Feynman–Kac models [1, 41, 40]. In their simplest form, they consist in running a particle filter algorithm conditioned on the current state of the MCMC chain “surviving” the resampling step. This kernel, called conditional SMC (cSMC), can be proven to be ergodic for the pathwise smoothing distribution of the systems under the weak hypothesis that the potential functions are bounded above [37, and references within]. In Algorithm 3, we reproduce the original version [1] of a cSMC kernel with $N \geq 2$ particles, targeting the posterior distribution of a generic Feynman–Kac model $\pi(x_{0:T}) \propto g_0(x_0)\, p_0(x_0) \left\{ \prod_{t=1}^{T} g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1}) \right\}$.

Other versions of this algorithm exist, in particular, when it is possible to evaluate p_t pointwise, we can rejuvenate the selection of the genealogy, allowing for lower degeneracy in the early time steps. The most notable two such methods are the backward and ancestor sampling methods [61, 40, respectively]. Another method, useful in our context, is that of [11, Section 3], which implements a

Algorithm 3: Conditional SMC

```
Result: An updated trajectory $x_{0:T}$
1 Function cSMC($x_{0:T}$, $N$)
      // Forward propagation
2     for $n = 1, 2, \dots, N-1$ do
3         Sample $X_0^n \sim p_0$ and set $w_0^n = g_0(X_0^n)$
4     Set $X_0^N = x_0$, $w_0^N = g_0(x_0)$
5     for $t = 1, \dots, T$ do
6         for $n = 1, \dots, N-1$ do
7             Sample $A_t^n$ with $\mathbb{P}(A_t^n = k) \propto w_{t-1}^k$
8             Sample $X_t^n \sim p_t(\cdot \mid X_{t-1}^{A_t^n})$ and set $w_t^n = g_t(X_t^n \mid X_{t-1}^{A_t^n})$
9         Set $X_t^N = x_t$, $w_t^N = g_t(x_t \mid x_{t-1})$
      // Genealogy selection
10    Sample $B_T$ with $\mathbb{P}(B_T^n = k) \propto w_T^k$ and set $x_T = X_T^{B_T}$
11    for $t = T-1, \dots, 0$ do
12        Set $B_t = A_{t+1}^{B_{t+1}}$, $x_t = X_t^{B_t}$
13    return $x_{0:T}$
```

parallel-in-time conditional SMC, particularly amenable to when the dynamics model is separable, that is, when $p_t(x_t \mid x_{t-1}) = p_t(x_t)$ does not depend on x_{t-1} as is the case for some of the samplers in this article.

4.2. Particle Gibbs for Feynman–Kac models with auxiliary observations

Section 2 offers a class of new samplers for latent dynamical systems with tractable moments. It is however not the case that all practical problems verify this assumption, or that the potential function is always differentiable. In this section, we instead consider the case of Feynman–Kac models (5) with tractable densities. For this class of models, the auxiliary target corresponds to a model with an augmented potential function

$$\begin{aligned}\pi(x_{0:T}, u_{0:T}) \propto g_0(x_0)\, p_0(x_0) &\left\{\prod_{t=1}^{T} g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1})\right\} \\ &\left\{\prod_{t=0}^{T} \mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right)\right\}.\end{aligned} \tag{39}$$

In order to sample from $\pi(x_{0:T}, u_{0:T})$, it is, therefore, enough to implement an abstract algorithm given by Algorithm 4.

Clearly, in Algorithm 4, if $(x_{0:T}^k, u_{0:T}^k)$ are distributed according to π, then $(u_{0:T}^{k+1}, x_{0:T}^k)$ are too after line 2, so that $x_{0:T}^k$ is distributed according to $\pi(\cdot \mid u_{0:T}^{k+1})$, and therefore $(x_{0:T}^{k+1}, u_{0:T}^{k+1})$ are still distributed according to π after line 3. Otherwise said, this algorithm can be seen as a "true" particle Gibbs algorithm [1] for the choice of an improper prior $\pi(u_{0:T}) = 1$ for the auxiliary variables.

Algorithm 4: Auxiliary cSMC

```
Result: An updated trajectory $x_{0:T}^{k+1}$
1 Function AUX-cSMC($x_{0:T}^k$)
2     Sample $u_{0:T}^k \sim \prod_{t=0}^T \mathcal{N}(u_t; x_t^k, \frac{\delta}{2}\Sigma_t)$
3     Sample $x_{0:T}^{k+1} \sim K(\cdot \mid x_{0:T}^k)$ // from a $\pi(x_{0:T} \mid u_{0:T}^k)$-invariant cSMC kernel
4     return $x_{0:T}^{k+1}$
```

At first sight, this may seem like a very bad idea, and it appears like we have made the problem more difficult than it was originally, and this is probably the reason why (to the best of our knowledge) this has not been *explicitly* proposed before. Indeed, instead of considering the potential function $g_t(x_t, x_{t-1})$, we are now considering the potential function $g_t(x_t, x_{t-1}) \mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right)$ at each time step t. This new potential function becomes very informative as δ gets smaller, which is known to induce high variance weights in particle filtering and smoothing algorithms [see, e.g. 9, Section 10.3.1]. However, rather than seeing $\mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right)$ as describing an auxiliary observation, we can leverage the symmetry of Gaussian distributions to look at it as the generative model $\mathcal{N}\left(x_t; u_t, \frac{\delta}{2}\Sigma_t\right)$ instead. We can consider the model

$$\pi(x_{0:T}, u_{0:T}) \propto \tilde{p}_0(x_0 \mid u_0) \left\{\prod_{t=1}^T \tilde{p}_t(x_t \mid x_{t-1}, u_t)\right\} \tilde{g}_0(x_0) \left\{\prod_{t=1}^T \tilde{g}_t(x_t, x_{t-1})\right\}, \tag{40}$$

for the modified dynamics $\tilde{p}_t(x_t \mid x_{t-1}, u_t) = \mathcal{N}\left(x_t; u_t, \frac{\delta}{2}\Sigma_t\right)$ and potential functions $\tilde{g}_t = g_t \cdot p_t$, $t = 0, 1, \ldots, T$ and similarly for $\tilde{p}_0$ and $\tilde{g}_0$.

This change of perspective immediately makes the problem much simpler, as we are now given a model with an informative and separable prior for which we can implement Step 3 of Algorithm 4 via Algorithm 3. Moreover, because the auxiliary prior model is separable across time, the method of [11] applies directly[3], and a parallel-in-time particle Gibbs can be implemented to reduce the computational complexity to $\mathcal{O}(\log T)$ on parallel hardware. We also note that, contrarily to [11, Section 5.2], in this specific case, doing so would not necessarily come at a loss of statistical efficiency compared to sequential conditional SMC counterparts. This is due to the fact that the sequential algorithms would also rely on sampling from independent proposals.

In hindsight, it is easy to see that this method is *exactly* the same one as the one proposed in [21, Algorithm 3 and extensions] who instead phrase it as a form of conditional SMC with exchangeable proposals. Informally, rather than using a proposal $\tilde{p}_t(x_t \mid x_{t-1})$, they use a proposal $\tilde{p}_t(x_t^1, \ldots, x_t^N)$ which induces an exchangeable dependency across particles, that is, $\tilde{p}_t(x_t^1, \ldots, x_t^N) = \tilde{p}_t(x_t^{\sigma(1)}, \ldots, x_t^{\sigma(N)})$ for any permutation σ. As done in [21] – and first introduced in the context of classical MCMC in [55] –, in the case of Gaussians, taking a

[3]While in [11] it was derived for likelihood terms $g_t(x_t)$ rather than $g_t(x_t, x_{t-1})$ this was a notational simplification, and all the results derived within in fact hold for bivariate potentials.

conditional sample $p(\dots, x_t^{k-1}, x_t^k, \dots \mid x_t^k)$ can for instance be achieved by first sampling a "centering" variable $u_t \sim \mathcal{N}(x_t^k, \frac{\delta_t}{2} I)$ and then the remainder of the variables from $\prod_{i \neq k} \mathcal{N}(x_t^i; u_t, \frac{\delta_t}{2} I)$. This directly corresponds to Algorithm 4 for the modified Feynman–Kac model (40).

Proposition 4.1. *The method of [21] implements Algorithm 4 with proposal distributions $\mathcal{N}\left(\cdot; u_t, \frac{\delta_t}{2}\Sigma_t\right)$ for different choices of kernels: embedded HMM [47], conditional SMC [1], conditional SMC with forced move [10], and conditional SMC with backward sampling [61].*

They show that for a given choice of a standard conditional SMC – with and without backward sampling – Algorithm 4 avoids the curse of dimensionality, that is, its mixing time increases linearly with respect to d_x rather than exponentially. This new perspective on their method is rich in consequences: the entirety of the literature on particle Gibbs can be applied to step 3 of Algorithm 4, and we can expect that the curse of dimensionality can be controlled in this case too.

4.3. Adapted proposals in particle Gibbs with auxiliary observations

In the previous section, we have described an algorithm that recovers [21, Algorithm 3 and extensions]. However, explicitly introducing the auxiliary variable allows us to decouple the state of the Markov chain and the generative model so that we can incorporate statistical information in the auxiliary particle Gibbs sampler beside simple locality. Formally, we can implement "locally-adapted" particle filters for $\pi(x_{0:T} \mid u_{0:T})$ that improve the statistical properties of [21]. While this can be applied to many models, we demonstrate how this can be done for differentiable models and for those that have (approximately) conditional Gaussian transitions and arbitrary potential functions.

4.3.1. Differentiable models

When the potential functions g_t are differentiable, it is possible to incorporate first or second-order information from the potential. Indeed, we have

$$\exp(\gamma(x_{0:T})) \approx \exp\left(\gamma(u_{0:T}) + \sum_{t=0}^{T} \frac{\partial \gamma(u_{0:T})}{\partial u_t} \cdot (x_t - u_t)\right) =: \prod_{t=0}^{T} \hat{g}_t(x_t \mid u_{0:T}). \tag{41}$$

Now, as in Section 2, we can form the proposal distributions

$$\begin{aligned} \tilde{p}_t(x_t \mid x_{t-1}, u_{0:T}) &\propto \mathcal{N}(x_t; u_t, \frac{\delta}{2}\Sigma_t)\, \hat{g}_t(x_t \mid u_{0:T}) \\ &\propto \mathcal{N}\left(x_t; u_t + \frac{\delta}{2}\Sigma_t \frac{\partial \gamma(u_{0:T})}{\partial u_t}, \frac{\delta}{2}\Sigma_t\right) \end{aligned} \tag{42}$$

and similarly for $\tilde{p}_0$.

Similarly to Section 2.2, when the potential function is separable, i.e., when we have $\gamma(x_{0:T}) = \sum_{t=0}^{T} \gamma_t(x_t)$, it is also possible to use second-order linearisation whilst not relinquishing the Feynman–Kac structure required to implement Algorithm 3. And, finally, when p_t is also differentiable, we can also include information from it in the sampler by considering $\exp(\gamma) = \prod_{t=0} p_t \, g_t$ rather than simply using g_t.

We can then plug these choices for $\tilde{p}$ and $\tilde{g}$ inside (40) to then recover a gradient-informed equivalent representation of $\pi(x_{0:T} \mid u_{0:T})$ that will still be local, as [21], but will have proposal distributions adapted to the model at hand. Interestingly, these new proposal distributions are fully separable in time, so that they can immediately be used in the parallel-in-time particle Gibbs algorithm of [11].

4.3.2. Approximately Gaussian transitions

Consider now the case when the prior process is conditionally Gaussian (or more generally, as in Section 2.2, when the prior conditional means and covariances are tractable). We can easily design a model [this is called a guided proposal in 9, Section 10.3.2] locally adapted to the auxiliary observation:

$$\begin{aligned} \tilde{p}_t(x_t \mid x_{t-1}, u_t) &\propto \mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right) \mathcal{N}\left(x_t; m^X_{t-1}(x_{t-1}), C^X_{t-1}(x_{t-1})\right) \\ &\propto \mathcal{N}\left(x_t; \mu_t, \Lambda_t\right), \end{aligned} \tag{43}$$

for

$$\begin{aligned} \mu_t &= m^X_{t-1}(x_{t-1}) + K_{t-1}[u_t - m^X_{t-1}(x_{t-1})] \\ \Lambda_t &= C^X_{t-1}(x_{t-1}) - K_{t-1} C^X_{t-1}(x_{t-1}), \end{aligned} \tag{44}$$

where $K_{t-1} = C^X_{t-1}(x_{t-1}) \left[C^X_{t-1}(x_{t-1}) + \frac{\delta}{2}\Sigma_t\right]^{-1}$. A similar form is available for p_0. Using this new proposal, and making the dependency on u_t implicit for notational simplicity, an equivalent Feynman–Kac model will then take the form

$$\pi(x_{0:T} \mid u_{0:T}) \propto \tilde{p}_0(x_0) \left\{ \prod_{t=1}^{T} \tilde{p}_t(x_t \mid x_{t-1}) \right\} \tilde{g}_0(x_0) \left\{ \prod_{t=1}^{T} \tilde{g}_t(x_t, x_{t-1}) \right\}, \tag{45}$$

where $\tilde{p}$ is given by (43), and

$$\tilde{g}_t(x_t, x_{t-1}) = \frac{g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1})}{\tilde{p}_t(x_t \mid x_{t-1})} \mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right). \tag{46}$$

Using such a generative model, contrary to the independent auxiliary proposal cases, is not parallelisable in time, and will scale as $\mathcal{O}(T)$, even on parallel hardware. On the other hand, when the potential is weakly informative compared to the dynamics, we can expect them to have better statistical properties, in the sense that they are likely to have lower auto-correlation than their independent proposal counterparts. We also note that the construction proposed in (43) and (45) extends to other methods developed to leverage approximate Gaussian conjugacy relationships in state-space models, for instance, they are directly compatible to Laplace approximations of the potential [see, e.g. 9, Section 10.5.3] or Rao–Blackwellisation [46].

4.3.3. Hybrid proposal models

It is worth highlighting that the two approaches presented before are not mutually exclusive. Indeed, we can combine an approximately Gaussian transition model together with a first or second-order linearisation of the potential function to obtain hybrid adapted proposals that may work better than their individual components taken in isolation.

With the notations above, this would, for example, correspond to

$$\begin{aligned}\tilde{p}_t(x_t \mid x_{t-1}, u_{0:T}) \propto \mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right) \mathcal{N}\left(x_t; m^X_{t-1}(x_{t-1}), C^X_{t-1}(x_{t-1})\right)& \\ \hat{g}_t(x_t \mid u_{0:T})& \\ \propto \mathcal{N}\left(u_t + \frac{\delta}{2}\Sigma_t \frac{\partial \gamma(u_{0:T})}{\partial x_t}; x_t, \frac{\delta}{2}\Sigma_t\right)& \\ \mathcal{N}\left(x_t; m^X_{t-1}(x_{t-1}), C^X_{t-1}(x_{t-1})\right),&\end{aligned} \tag{47}$$

if the linearisation point of γ was taken to be $u_{0:T}$. This can then be simplified explicitly as in (43) to obtain gradient-informed, guided proposals.

Other linearisation/combination choices are also possible, and the willing statistician is free to fully leverage the flexibility brought by introducing the auxiliary observations $u_{0:T}$. Understanding which is the best choice will typically be application specific, although we expect the methods presented in this section to provide a competitive test-bed for more advanced methods.

4.4. Extension to pseudo-marginal methods

While the particle Gibbs approach to sampling from (39) is perhaps the most natural, it is also possible to instead consider a pseudo-marginal approach [2] as given by the particle marginal Metropolis-–Hastings (PMMH) sampler of [1]. Consider a proposal distribution $q(\mathrm{d}u'_{0:T} \mid u_{0:T})$, for example, $\prod_0^T \mathcal{N}(u'_t; u_t, \frac{\delta}{2}\Sigma_t)$. Similarly to PMMH, because sequential Monte Carlo provides an unbiased estimate $\hat{\mathcal{Z}}_N(u_{0:T})$ of the normalising constant for $\pi(x_{0:T} \mid u_{0:T})$, we can marginally target $\pi(x_{0:T})$ using a PMMH methodology. We succinctly summarise this extension in Algorithm 5.

This method is related to the method of [18]. They show that, by correlating the noise introduced by the particle filter, the pseudo-marginal algorithm can be made to scale better with time series of increasing lengths T. This is because

Algorithm 5: Auxiliary pseudo-marginal sampler

```
  Result: An updated trajectory $x_{0:T}^{k+1}$
1 Function AUX-PM($x_{0:T}^k$, $u_{0:T}^k$, $\hat{\mathcal{Z}}_N^k$)
2     Sample $u'_{0:T} \sim q(\cdot \mid u_{0:T}^k)$
3     Sample $x'_{0:T}$ and $\hat{\mathcal{Z}}'_N$ using a particle filter targeting $\pi(x_{0:T} \mid u'_{0:T})$
4     Set $x_{0:T}^{k+1}$ to $x'_{0:T}$ with probability $\frac{\hat{\mathcal{Z}}'_N q(u_{0:T}^k \mid u'_{0:T})}{\hat{\mathcal{Z}}_N^k q(u'_{0:T} \mid u_{0:T})}$, otherwise, set it to $x_{0:T}^k$
5     return $x_{0:T}^{k+1}$
```

it results in correlated likelihood ratios $\frac{\hat{\mathcal{Z}}'_N}{\hat{\mathcal{Z}}_N^k}$ which exhibit lower variance than they would have otherwise.

By using a proposal distribution adapted to the auxiliary target at hand, in a similar spirit as for the auxiliary particle Gibbs sampler of Algorithm 4, we can hope to also benefit from a reduced variance of the likelihood estimates ratio in Algorithm 5. This, however, is not because the two estimates are correlated, but rather because they will both exhibit lower variance individually than their non-augmented counterparts. Contrary to [18], this method necessitates the evaluation of the full (unnormalised) density of the Feynman–Kac model at hand, and will likely not perform well for a very large T. On the other hand, and in contrast to the correlated pseudo-marginal method [18, see the comments in Theorem 3 and Section 5.3], Algorithm 5 is likely to perform well in higher dimensions, due to the localisation of the proposals. Both approaches are furthermore not incompatible and could be used together. The benefit of doing so compared to simply using a particle Gibbs sampler, which (under backwards sampling) is stable for an increasing number of observations too [37], is however not clear, and we leave the study of this question open for future work.

5. Experimental evaluation

In this section, we aim to empirically evaluate the statistical and computational behaviours of our proposed methods. To this end, we consider three sets of examples. In all cases we compare to state-of-the-art methods, that is either the original method of [21] or [44].

- The first model is a multivariate stochastic volatility model known to be challenging for Gaussian approximations and used as a benchmark in, for example, [30, 21]. This model has latent Gaussian dynamics, and an observation model which both happen to be differentiable with respect to the latent state, so that all the methods of Section 2.2 and Section 4 apply. We consider the same parametrisation as in [21], which makes the system lack ergodicity and the standard particle Gibbs samplers not converge.
- The second one is a spatio-temporal model with independent latent Gaussian dynamics and is used in [13] as a benchmark for high dimensional filtering. This model is akin to a type of dynamic random effect model in

the sense that the latent states only interact at the level of the observations. This model is used to illustrate how latent structure can be used to design computationally efficient Kalman samplers that beat cSMC ones when the runtime is taken into account.

- The last model performs joint parameter and state estimation for a discretely observed stochastic differential equation. This model was used in [44] to assess the performance of their forward-guiding backwards-filtering method. We demonstrate here how to use auxiliary samplers for the same purpose and show the competitiveness of our approach.

Throughout this section, when using an auxiliary cSMC sampler, be it the sequential or the parallel-in-time formulation, we use $N = 25$ particles and a target acceptance rate of 25% across all time steps. This is more conservative than the recommendation of [21], corresponding to $1 - (1 + N)^{-1/3} \approx 66\%$. The difference stems from the fact that it may happen that the methods do not reach the relatively high acceptance rate implied by the more optimistic target for all time steps, even with very small δ values. As a consequence, the sampler is "stuck" by only proposing very correlated trajectories in some places. We believe that this is mostly due to the largely longer time series considered here as well as to the use of multinomial resampling which prevents achieving the optimal acceptance rate of $N/(N+1)$ when $\delta \ll 1$. Softening this constraint resulted in empirically better mixing. Furthermore, for all the samplers, and following [54, 21], we consider $\delta \Sigma_t = \delta_t I$, with δ_t being constant across time steps for the Kalman samplers. We then calibrate δ_t to achieve the desired acceptance rate (globally for Kalman samplers or per time step for the cSMC samplers) and the actual acceptance rate is reported below. Finally, we note that all the posterior distributions recovered from all the proposed methods were coherent so we only report mixing statistics throughout.

The implementation details for all the experiments are as follows: whenever we say that a method was run on a CPU, we have used an AMD® Ryzen Threadripper 3960X with 24 cores, and whenever the method has been run on a GPU, we used a Nvidia® GeForce RTX 3090 GPU with 24GB memory. All experiments were implemented in Python [59] using the JAX library [8] which natively supports CPU and GPU backends as well as automatic differentiation that we use to compute the gradients required. The code to reproduce the experiments listed below can be found at the following address: `https://github.com/AdrienCorenflos/aux-ssm-samplers`.

5.1. Multivariate stochastic volatility model

We consider the same multivariate stochastic volatility example as in [21, Section E.3.]. This model is classically used as a benchmark for high dimensional SMC-related methods [see also 30]. It is given by homogeneous auto-regressive Gaussian latent dynamics $p_t(x_t \mid x_{t-1}) = \mathcal{N}(x_t; F\, x_{t-1}, Q)$ and a potential defined as a multidimensional observation model

$$g(x_{0:T}) = \prod_{t=0}^{T} h(y_t \mid x_t), \text{ where } \quad h(y_t \mid x_t) = \prod_{d=1}^{d_x} \mathcal{N}(y_t(d); 0, \exp(x_t(d))). \tag{48}$$

As per [21], we take $F = \phi I_d$, $Q_{ij} = \tau(\delta(i = j) + \delta(i \neq j)\rho)$ for $\phi = 90\%$, $\tau = 2$, and $\rho = 25\%$. Similarly, the initial distribution $p_0(x_0)$ is also taken to be the stationary distribution of the latent Gaussian dynamics and we take $d_x = 30$. However, we increase the number of time steps to $T = 250$, rather than 50 and we take the number of particles for all the auxiliary cSMC algorithms to be $N = 25$.

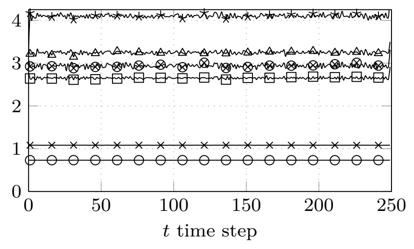
The different methods we compare here are the following: (i) auxiliary Kalman sampler with first order linearisation (13) (both on CPU and GPU), (ii) with second order linearisation (15) (both on CPU and GPU), (iii) auxiliary cSMC sampler with backward sampling for the proposals $\mathcal{N}(\cdot; u_t, \frac{\delta_t}{2} I)$ corresponding to [21] (on CPU), (iv) auxiliary cSMC sampler with parallel-in-time [11] sampling for the proposals $\mathcal{N}(\cdot; u_t, \frac{\delta_t}{2} I)$ (on GPU), (v) auxiliary cSMC sampler with backward sampling for the gradient-informed proposals (42) (on CPU), (vi) auxiliary cSMC sampler with parallel-in-time sampling for the gradient-informed proposals (42) (on GPU), and (vii) the guided auxiliary cSMC sampler with backward sampling for both the proposals (43) and (43) with the gradient information of (42) (on CPU).

In order to compare the samplers in this example, we generate 10 different datasets. For every dataset, we run each sampler for 2 500 adaptation steps. After this, we run 10 000 more iterations to compute the empirical root expected squared jump distance [RESJD, 49] for each sampler, defined as, for each time step t, the empirical value of

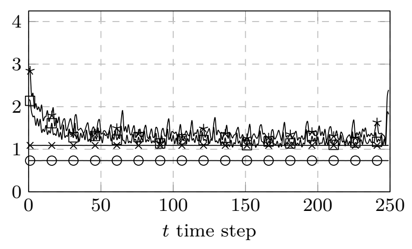
$$\sqrt{\frac{1}{L} \sum_{l=1}^{L-1} \sum_{i,j=1}^{d} \left[X_{t+1}^{A+l+1}(i,j) - X_{t+1}^{A+l}(i,j)\right]^2}. \tag{49}$$

All samplers, in both the sequential and parallel case, were targeting 50% acceptance rate across all time steps and the effective acceptance rate ranged between 47 and 52% for all samplers and time steps. The averaged (across the 10 experiments) RESJD is reported in Figure 1a for the sequential versions of the algorithm, and in Figure 1b for the parallel counterparts (noting that there is, as expected, no statistical difference between the sequential and parallel implementations of the Kalman samplers). As highlighted by Figure 1a, the gradient-informed auxiliary cSMC statistically dominates the alternatives for all time steps on both CPU and GPU (although this is less obvious on the GPU).

This picture, however, is modified when looking at the RESJD per second – heuristically corresponding to the average cumulative distance travelled by the sampler in a unit of time – rather than per iteration in Figures 2a and 2b. In this case, on the CPU, the method of [21] dominates the other ones. This



(a) CPU: uxiliary first order Kalman sampler −○−, second order Kalman sampler −×−, cSMC sampler −□−, gradient-informed cSMC −★− sampler, the guided cSMC −⊗− sampler, and the gradient-informed guided cSMC −△− sampler.

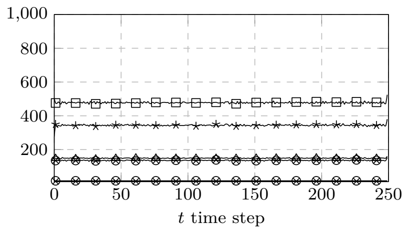


(b) GPU: Auxiliary first order Kalman sampler −○−, second order Kalman sampler −×−, cSMC sampler −□−, and gradient-informed cSMC −★− sampler. The latter two are hard to distinguish, but the gradient-informed cSMC is generally above the non-informed.

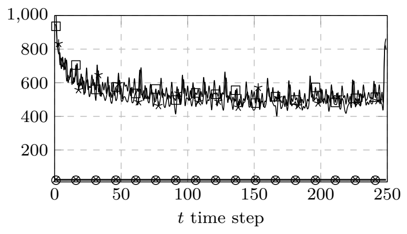
Fig 1: Average (across 10 different experiments) root expected squared jump distance per iteration for all the sampler considered on the stochastic volatility model of Section 5.1.

is because it offers reasonable statistical efficiency ($\sim$ 70% the RESJD of the most efficient sampler tested here) with a rather small time-complexity overall (no gradient calculation and no matrix inversion like in the Kalman samplers is needed here). On the other hand, the Kalman samplers are here completely dominated by all Monte Carlo alternatives. The GPU picture is more mixed, and both the gradient-informed and uninformed proposals seem to provide the same overall efficiency in this case but still completely dominate the Kalman alternatives here too.

This underwhelming performance of the Kalman sampler was in fact to be expected given the need to solve 250 matrix systems of dimension 30 per iteration (albeit some are done in parallel on GPU). In fact, this had another deleterious effect: the parallel versions of the auxiliary Kalman sampler suffered from numerical divergence in this experiment when using single precision floats (32-bits representation). This problem, due to the numerical instability of the covariance matrices calculations is well known in the literature [see, e.g. 6] and prompted the development of a square-root version of the parallel Kalman filtering and smoothing algorithms in [63]. Here, we instead simply used double precision floats instead of the square-root method as this sufficed to fix the numerical instability. This numerical instability is an important drawback of Kalman methods in general and is particularly salient on parallel hardware which is often optimised to run on lower precision arithmetic [33]. The issue did not arise for the sequential version of the algorithms, and we, therefore, stuck to single float precision arithmetic for these. In the next section, we show how latent structure can be leveraged to bypass the dimensionality problem.



(a) CPU: auxiliary first order Kalman sampler —○—, second order Kalman sampler —×—, cSMC sampler —□—, gradient-informed cSMC —★— sampler, the guided cSMC —⊗— sampler, and the gradient-informed guided cSMC —△— sampler.



(b) GPU: auxiliary first order Kalman sampler —○—, second order Kalman sampler —×—, cSMC sampler —□—, and gradient-informed cSMC —★— sampler. The latter two are hard to distinguish, with no clear difference in terms of performance.

Fig 2: Average (across 10 different experiments) root expected squared jump distance per second for all the sampler considered on the stochastic volatility model.

5.2. *Spatio-temporal model*

Finally, we consider the spatio-temporal model of [13, Section 4.2] which was recently introduced as a benchmark for high-dimensional state inference in nonlinear systems. It consists of independent latent dynamics for a state $X_t(i,j)$ located on a two-dimensional lattice $\{1,\dots,d\} \times \{1,\dots,d\} \ni (i,j)$, for $d = 8$, with an observation model that does not factorise over the nodes of the lattice, thereby creating non-trivial posterior structure between the states. We are given a $8^2 = 64$ dimensional model

$$\begin{aligned} X_t(i,j) &= X_{t-1}(i,j) + U_t(i,j), \quad i,j = 1,\dots,d, \\ Y_t(i,j) &= X_t(i,j) + V_t(i,j), \quad i,j = 1,\dots,d, \end{aligned} \tag{50}$$

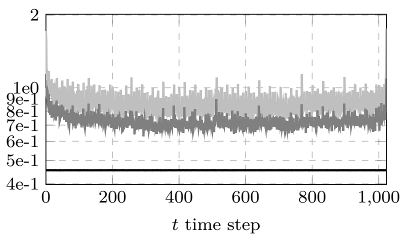
where, for all t, i, j, the $U_t(i,j)$ are i.i.d. according to $\mathcal{N}(0, \sigma_X^2)$, and for all t, the V_t's are i.i.d. according to a multivariate t-distribution with ν degrees of freedom centred on 0. The precision matrix of the V_t's is given by $\Sigma^{-1} = \tau^{D[(i,j),(i',j')]}$ if $D[(i,j),(i',j')] \le r_y$ and 0 otherwise, where $D[(i,j),(i',j')]$ is a graph distance, and $\tau < 0$ a given parameter.

In [13], the parameters are chosen to be $\sigma_X = 1$, $\nu = 10$, $\tau = -1/4$, $r_y = 1$, and $D[(i,j),i',j')] = |i-i'| + |j-j'|$, so that an observation is mostly corrupted by its direct neighbours. We keep all the parameters unchanged, with the exception that, in order to make the problem more difficult, we take $\nu = 1$ so that the observation model does not have first or second moments, and to showcase the parallelisation in time, we also consider a substantially higher number of time steps $T = 1\,024$ [vs. $T = 10$ in 13]. Overall, the total dimension of the target model is therefore of the order of 65 000. The first-order auxiliary Kalman sampler is particularly suited to this type of model, even if the underlying state dimension is large. This is due to the fact that the prior factorises across all dimensions, so that the auxiliary LGSSM proposal (22) factorises too, even if the target $\pi(x_{0:T})$ does not. As a consequence, we are left with sampling from $d \times d$ independent one-dimensional LGSSMs rather than a $d \times d$ dimensional one. This means that, instead of needing to compute conditional Gaussian distributions of dimension $d \times d$, and therefore needing to solve systems of size $d \times d$, we only need to solve one-dimensional systems, that is, divide by scalars. This property extends to some extent to auxiliary cSMC samplers where the proposal is chosen to factorise across dimensions too. This means that the cost will be dominated by the computation of the log-likelihood of the multivariate t-distribution at each time step and (for specialised implementations) the complexity of the auxiliary cSMC will then be a direct multiple of the complexity of the auxiliary Kalman sampler. This property is not verified in the case of "full" prior dynamics as we will see in Section 5.1.

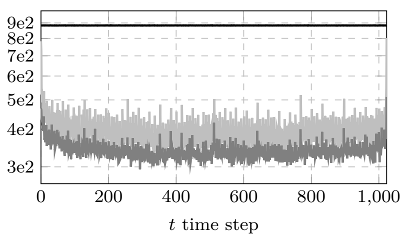
The experiment design is as follows: we simulate 20 datasets from (50). For each of these, we set the initial trajectory of the MCMC chain to be the result of a single trajectory formed from the backward sampling [28] of a bootstrap filter algorithm with 1 000 particles (this gives bad smoothing statistics but is a good starting point for an MCMC chain) and run $A = 5\,000$ adaptation steps, after which the statistics of the chain are collected over $L = 20\,000$ iterations. For this experiment, all the sequential versions of the auxiliary Kalman and cSMC samplers were dramatically slower than the parallel-in-time alternatives: they took in the order of a second per iteration, both on CPU and GPU, compared to the PIT versions that took in the order of a millisecond per iteration on GPU. As a consequence, we do not report their results here. Instead, we focus on (i) the parallel-in-time version of [21] given by using [11] on step 3 of Algorithm 4, (ii) the parallel-in-time version of the gradient auxiliary proposal (42) of Section 4.3, which we refer to as gradient-informed, and finally (iii) the auxiliary Kalman sampler (13) of Section 2.2, with first-order linearisation only, noting that the second order would remove the benefits of having a separable prior.

As per [54], we target a 50% acceptance rate for the auxiliary Kalman sampler. The final average acceptance rates were a little lower, with the auxiliary Kalman sampler accepting 34% of the trajectories. This is most likely due to our calibration algorithm being too optimistic, but did not seem to impact the final results beyond reason and therefore did not, in our opinion, warrant further investigation.

The RESJD is shown, averaged over all experiments, in Figure 3a, while the time-scaled RESJD, namely RESJD divided by the number of seconds taken to run one step of the sampler is shown, averaged over all experiments, in Figure 3b. The gradient-enhanced PIT auxiliary cSMC has a better RESJD than the basic PIT auxiliary cSMC which in turn has a better RESJD than the auxiliary Kalman sampler. The ordering of these methods however changes if one takes into account the additional complexity incurred by SMC, and after rescaling by the time taken by iteration, the auxiliary Kalman sampler dominates the gradient-enhanced PIT auxiliary cSMC which still dominates its basic



(a) Root expected squared jump distance for the auxiliary Kalman sampler ▬, the auxiliary cSMC sampler ▬, and the auxiliary cSMC sampler with gradient-informed proposals ▬. Kalman ▬ shows as a roughly horizontal line at the bottom.



(b) Root expected squared jump distance per second for the auxiliary Kalman sampler ▬, the auxiliary cSMC sampler ▬, and the auxiliary cSMC sampler with gradient-informed proposals ▬.

Fig 3: Average (across 20 different experiments) root expected squared jump distance per iteration and second for all the samplers considered on the spatio-temporal model (50).

counterpart.

In practice, the auxiliary Kalman, conditional SMC, and gradient-enhanced conditional SMC samplers took respectively in average 0.52, 2.1, and 2.2 milliseconds per iteration. While some idiosyncrasies may be present, we believe that this performance gap could be further improved by careful consideration of the structure of the model in the Kalman sampler — we have not undertaken this here in order to preserve the general applicability of our implementation.

5.3. *Parameter estimation in a continuous-discrete diffusion smoothing problem*

In this section, we consider the same experiment as in [44, Section 6.1], which consists of a joint sampling of the state of a discretely observed chaotic Lorenz stochastic differential equation, and of the parameter defining its drift. The SDE is given, conditionally on a parameter $\theta = (\theta_1, \theta_2, \theta_3)$ as a three-dimensional SDE $\mathrm{d}x = \beta_\theta(x)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$, where W is a three-dimensional standard Wiener process and

$$\beta_\theta(x) = \begin{pmatrix} \theta_1(x_2 - x_1) \\ \theta_2 x_1 - x_2 - x_1 x_3 \\ x_1 x_2 - \theta_3 x_3 \end{pmatrix}. \tag{51}$$

The state is then observed at regular intervals (every $t_0 = 0.01, t_1 = 0.02, t_2 = 0.03, \dots, t_K = 2$) through its second and third component only, giving an observation model $Y_k \sim \mathcal{N}(Hx(t_k), 5I_2)$, for $H = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. In order to provide comparable results to [44], we use the code they provided to generate the same dataset and pick the same parametrisation of the model, including the same prior for the parameters. The Markov chain is then initialised according to the prior dynamics conditionally on the same initial parameter values as in [44]. As per their experiment, we sample from the joint distribution $\pi(x(t'_0), \dots, x(t'_L), \theta \mid y_{0:K})$, where $t'_0 = 0, t'_1 = 2e-4, \dots, t'_L = 2$ is a finer grid, making for a total sampling space dimension of $3 + 30\,000$. To do so, we too use the conjugacy relationship of θ given the full path for x, implementing a Hastings-within-Gibbs routine which samples θ conditionally on $x(t'_0), \dots, x(t'_L)$ using its closed-form Gaussian posterior [44, Proposition 4.5], and then the auxiliary Kalman sampler to sample $x(t'_0), \dots, x(t'_L)$ conditionally on θ.

In our case, because the observation model is linear, we use the following proposal in the Kalman sampler: first, given the current trajectory $(x^k(t'_l))_{l=0}^L$ and parameter θ^k state of the MCMC chain we linearise $\mathbb{E}[x(t'_l) \mid x(t'_{l-1})] = \beta_{\theta^k}(x(t'_{l-1}))(t'_l - t'_{l-1})$ around $x^k(t'_{l-1})$ using the method of Section 2.2 with extended linearisation, obtaining approximations

$$p(x(t'_l) \mid x(t'_{l-1})) \approx \mathcal{N}(x(t'_l); F_{l-1} x(t'_{l-1}) + b_{l-1}, Q_{l-1}) \tag{52}$$

For $l = 0, \dots, L$ we then sample $u_l \sim \mathcal{N}(x(t'_l), \frac{\delta}{2} I_3)$, and then form the proposal

$$\begin{aligned} & q\left((z(t'_l))_{l=0}^L \mid u_{0:L}, (x^k(t'_l))_{l=0}^L, y_{0:L}\right) \\ & \propto \mathcal{N}\left(z(t'_0); m_0, P_0\right) \left\{ \prod_{l=1}^L \mathcal{N}\left(z(t'_l); F_{l-1} z(t'_{l-1}) + b_{l-1}, Q_{l-1}\right) \right\} \\ & \quad \left\{ \prod_{k=0}^K \mathcal{N}\left(y_k; H z(t_k), 5 I_2\right) \right\} \left\{ \prod_{l=0}^L \mathcal{N}\left(u_l; z(t'_l), \frac{\delta}{2} I_3\right) \right\} \end{aligned} \tag{53}$$

targeting the augmented model

$$\begin{aligned} & \pi\left((z(t'_l))_{l=0}^L \mid u_{0:L}, y_{0:K}\right) \\ & \propto \mathcal{N}\left(z(t'_0); m_0, P_0\right) \left\{ \prod_{l=1}^L p\left(z(t'_l) \mid z(t'_{l-1})\right) \right\} \\ & \quad \left\{ \prod_{k=0}^K \mathcal{N}\left(y_k; H z(t_k), 5 I_2\right) \right\} \left\{ \prod_{l=0}^L \mathcal{N}\left(u_l; z(t'_l), \frac{\delta}{2} I_3\right) \right\}. \end{aligned} \tag{54}$$

We run 2 500 adaptation steps, during which we modify δ to target an average acceptance rate of 23.4% (as per [44]). Interestingly, our actual acceptance rate after adaptation was closer to 70%, and the resulting δ was virtually infinite. This means that the proposal distribution is almost reversible with respect to the target distribution. This high acceptance rate did not negatively impact the convergence of our algorithm. In fact, our resulting effective sampling size was larger than the best one reported by [44] for both the parameters and the smoothing marginals (while the posterior distributions were similar). We report this in Table 1.

Table 1
Effective sample size (ESS) for the auxiliary sampler, computed using chains of length 10^5*. The results for [44] are reported for ease of comparison.*

	$X_{1,1.5}$	$X_{2,1.5}$	$X_{3,1.5}$	θ_1	θ_2	θ_3
This paper	31254.0	35469.9	36584.7	11850.0	22960.5	12240.5
[44]	10480.3	22890.5	24070.2	4592.4	15379.5	10917.7

In practice, our sampler took 3 149 seconds (52 minutes) to run on the GPU, and 9 424 seconds (2h30mn) on the CPU. [44], on the other hand, report much faster run times (roughly 3-4 minutes). While this difference may seem massive, it can be imputed in totality to the difference in software for this experiment. Indeed, because they too rely on Gaussian filtering, the theoretical serial complexity of the two methods (when run on CPU) are exactly the same. While they use the programming language Julia [5], we use the JAX library [8] written in the Python language. Our choice comes with the benefit of direct GPU support but also presents the inconvenience of not supporting varying-size arrays. Consequently, rather than running Kalman filtering on the proposal LGSSM (53) optimally by alternatively considering independent observations of size 3 (u_l) and 2 (y_k), we have to consider stacked observations (u_l, y'_l) of dimension 5 and treat the y'_l as being missing when t'_l is not part of the t'_k. This technical limitation would be removed by considering instead a specialised implementation in a framework allowing for such optimisations.

6. Discussion

In this article, we have presented a principled approach to doing MCMC-based inference in general tractable Feynman–Kac models. At the core, the method corresponds to augmenting the model by introducing an artificial observation model, and then proceeding to sample from the augmented model using a two-step approach: first sample the observations conditionally on a trajectory, and second, sample from a MCMC kernel keeping the trajectory conditional distribution invariant.

To summarise, we have described two versions of this class of samplers. The first one, which we coined auxiliary Kalman sampler can be seen as an extension/specialisation of [54] to models with latent dynamics, and is particularly useful when the latent model is quasi-Gaussian and of relatively small dimension. We believe that this class of samplers opens the door to using the Gaussian approximations developed in the signal processing community for exact inference in state-space models. The second class, which considers using conditional SMC to sample the trajectory conditional to the auxiliary observations, can be seen as a generalisation of [21] which allows for more flexibility (and therefore performance) in the design of proposal distributions. Importantly, we have shown that both methods introduced could be parallelised across time steps on hardware such as GPUs, while retaining good statistical properties: the sequential and parallel versions of the auxiliary Kalman sampler are fully statistically equivalent, while the particle Gibbs ones are not, but the parallel-in-time auxiliary particle Gibbs does not suffer from worse mixing properties, in particular when run time is taken into account.

At least two classes of latent Markovian models elude our auxiliary Kalman samplers:

1. Models with multi-modal posteriors, which are hard for MCMC methods in general due to the "local" perspective they take. This can, however, be handled by combining the method with meta-algorithms, such as parallel tempering [25].
2. Models with very non-Gaussian latent dynamics or observations, such as those exhibiting multiplicative noise or presenting boundary constraints akin to discontinuities.

Another, softer, issue is concerned with the computational complexity of the method with respect to the dimension of the latent space d_X. Indeed, because Kalman filtering and backward sampling relies on recursive Gaussian conditioning, it requires computing matrices inverses of size $d_X \times d_X$ (or more precisely, solving systems of the same size). In models where no specific structure alleviates these computations, they can quickly become computationally overwhelming as the dimension of the latent space increases.

Replacing the LGSSM proposal of Section 2.2 by a local conditional SMC update as per Section 4 allowed us to trade the single expensive computation for a quick computation across several particles. This also partially alleviates the other issues with Kalman approximations. However, the usual issues with cSMC remain: several trajectories need to be simulated, and the fully adapted auxiliary cSMCs of Section 4.3 cannot be parallelised-in-time. They however do not solve the problem of intractable densities, or multimodality.

The reformulation of [21] as a conditional SMC within a Gibbs sampler is a particularly promising avenue as it invites the direct application of the many cSMC practical and theoretical technologies developed over the past decade. Our experiments showed that leveraging this representation to design better auxiliary proposal distributions already largely improved the statistical properties of the algorithm at a very low additional computational cost. Interestingly, the main benefit came from introducing gradients in the proposal, rather than incorporating the dynamics themselves. This is likely because the potentials have been designed to be particularly informative compared to the dynamics (in order to increase the variance of the underlying particle filter). We believe that this can still be improved upon many-fold in a number of settings. A natural first step would be to combine these with methods developed to tackle degeneracy in particle Gibbs [e.g. 39] or very long time series [35].

A final remark is concerned with the implementation of the prefix-sum algorithm [7] in the JAX library [8]. At the time of writing this article, the JAX implementation can be considered high-level, by which we mean that the algorithm is implemented in Python [59] rather than natively using the CUDA [48] GPU backend. This is in contrast to other control flow primitives such as loops and "if-else" branching and a native implementation of the algorithm, fully GPU-focused would improve the time-performance of the Kalman samplers.

Individual Contributions

The original idea, methodology, implementation, and redaction of the first version of this article are due to Adrien Corenflos. Simo Särkkä contributed the divide-and-conquer sampling algorithm and reviewed the final version of the manuscript.

Acknowledgments

The first author would like to warmly thank Nicolas Chopin for pointing out the link between the first method presented in this article and [54]. Some of the cSMC ideas presented in this article also stemmed from discussions and presentations that took place at the "Computational methods for unifying multiple statistical analyses" (Fusion) workshop organised by Rémi Bardenet, Kerrie Mengersen, Pierre Pudlo, and Christian Robert in Centre International de Rencontres Mathématiques (CIRM) in October 2022.

References

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

[2] Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics* **37** 697 – 725.

[3] Barfoot, T. D. (2017). *State estimation for robotics.* Cambridge University Press.

[4] Bell, B. M. (1994). The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization* **4** 626–636.

[5] Bezanson, J., Edelman, A., Karpinski, S. and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review* **59** 65–98.

[6] Bierman, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation.* Academic Press.

[7] Blelloch, G. E. (1989). Scans as primitive parallel operations. *IEEE Transactions on Computers* **38** 1526–1538.

[8] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D. and Wanderman-Milne, S. (2018). JAX: composable transformations of Python+NumPy programs. `http://github.com/google/jax`.

[9] Chopin, N. and Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo.* Springer.

[10] Chopin, N. and Singh, S. S. (2015). On particle Gibbs sampling. *Bernoulli* **21** 1855–1883.

[11] Corenflos, A., Chopin, N. and Särkkä, S. (2022). De-Sequentialized Monte Carlo: a parallel-in-time particle smoother. *Journal of Machine Learning Research* **23** 1–39.

[12] Cotter, S. L., Roberts, G. O., Stuart, A. M. and White, D. (2013). MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster. *Statistical Science* **28** 424 – 446.

[13] Crucinio, F. R. and Johansen, A. M. (2022). A divide and conquer sequential Monte Carlo approach to high dimensional filtering. *arXiv preprint arXiv:2211.14201*.

[14] Dau, H.-D. and Chopin, N. (2022). On the complexity of backward smoothing algorithms. *arXiv preprint arXiv:2207.00976*.

[15] Daum, F. and Huang, J. (2003). Curse of dimensionality and particle filters. In *2003 IEEE aerospace conference proceedings (Cat. No. 03TH8652)* **4** 4_1979–4_1993. IEEE.

[16] de Jong, P. and Shephard, N. (1995). The Simulation Smoother for Time Series Models. *Biometrika* **82** 339–350.

[17] Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer New York, New York, NY.

[18] Deligiannidis, G., Doucet, A. and Pitt, M. K. (2018). The correlated pseudomarginal method. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **80** 839-870.

[19] Douc, R., Garivier, A., Moulines, E. and Olsson, J. (2011). Sequential Monte Carlo smoothing for general state space hidden Markov models. *The Annals of Applied Probability* **21** 2109–2145.

[20] Doucet, A. (2010). A Note on Effcient Conditional Simulation of Gaussian Distributions Technical Report, University of British Columbia.

[21] Finke, A. and Thiery, A. H. (to appear, 2023). Conditional sequential Monte Carlo in high dimensions. *Annals of Statistics*.

[22] Frühwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *Journal of Time Series Analysis* **15** 183–202.

[23] García-Fernández, Á. F., Svensson, L. and Särkkä, S. (2017). Iterated posterior linearization smoother. *IEEE Transactions on Automatic Control* **62** 2056–2063.

[24] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence* **PAMI-6** 721–741.

[25] Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *Interface Proceedings*.

[26] Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73** 123-214.

[27] Godsill, S. J., Doucet, A. and West, M. (2004). Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association* **99** 156–168.

[28] Godsill, S. J., Doucet, A. and West, M. (2004). Monte Carlo Smooth-

ing for Nonlinear Time Series. *Journal of the American Statistical Association* **99** 156-168.

[29] Gordon, N. J., Salmond, D. J. and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE proceedings F (radar and signal processing)* **140** 107–113. IET.

[30] Guarniero, P., Johansen, A. M. and Lee, A. (2017). The Iterated Auxiliary Particle Filter. *Journal of the American Statistical Association* **112** 1636-1647.

[31] Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* **57(1)** 97–109.

[32] Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory.* Academic Press.

[33] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A. et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international Symposium on Computer Architecture* 1–12.

[34] Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* **92** 401–422.

[35] Karppinen, S., Singh, S. S. and Vihola, M. (2022). Conditional particle filters with bridge backward sampling. *arXiv preprint arXiv:2205.13898.*

[36] Kitagawa, G. (1996). Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics* **5** 1-25.

[37] Lee, A., Singh, S. S. and Vihola, M. (2020). Coupled conditional backward sampling particle filter. *The Annals of Statistics* **48** 3066–3089.

[38] Leisen, F. and Mira, A. (2008). An extension of Peskun and Tierney orderings to continuous time Markov chains. *Statistica Sinica* 1641–1651.

[39] Lindsten, F., Bunch, P., Singh, S. S. and Sch on, T. B. (2015). Particle ancestor sampling for near-degenerate or intractable state transition models. *arXiv preprint arXiv:1505.06356.*

[40] Lindsten, F., Jordan, M. I. and Schön, T. B. (2014). Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research* **15** 2145–2184.

[41] Lindsten, F., Schön, T. and Jordan, M. (2012). Ancestor sampling for particle Gibbs. *Advances in Neural Information Processing Systems* **25**.

[42] Malory, S. J. (2021). *Bayesian Inference for Stochastic Processes.* Lancaster University (United Kingdom).

[43] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* **21** 1087–1092.

[44] Mider, M., Schauer, M. and Van der Meulen, F. (2021). Continuous-discrete smoothing of diffusions. *Electronic Journal of Statistics* **15** 4295–4342.

[45] Müller, P. (1993). Alternatives to the Gibbs sampling scheme Technical Report, Institute of Statistics and Decision Sciences, Duke Univ.

[46] Murphy, K. and Russell, S. (2001). *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks* In *Sequential Monte Carlo Methods in Practice* 499–515. Springer New York, New York, NY.

[47] Neal, R. M. (2003). Markov Chain Sampling for Non-linear State Space Models Using Embedded Hidden Markov Models.

[48] NVIDIA, Vingelmann, P. and Fitzek, F. H. P. (2022). CUDA, release: 11.8.x.

[49] Pasarica, C. and Gelman, A. (2010). Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica* 343–364.

[50] Peskun, P. H. (1973). Optimum Monte-Carlo sampling using Markov chains. *Biometrika* **60** 607–612.

[51] Särkkä, S. and García-Fernández, Á. F. (2021). Temporal parallelization of Bayesian smoothers. *IEEE Transactions on Automatic Control* **66** 299–306.

[52] Särkkä, S. and Lennart, S. (2023). *Bayesian filtering and smoothing.* Cambridge University Press.

[53] Tierney, L. (1998). A note on Metropolis-Hastings kernels for general state spaces. *Annals of applied probability* 1–9.

[54] Titsias, M. K. and Papaspiliopoulos, O. (2018). Auxiliary gradient-based sampling algorithms. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **80** 749-767.

[55] Tjelmeland, H. (2004). Using all Metropolis–Hastings proposals to estimate mean values Technical Report, NTNU.

[56] Tronarp, F. (2020). Iterative and Geometric Methods for State Estimation in Non-linear Models, PhD thesis, Aalto University.

[57] Tronarp, F., García-Fernández, Á. F. and Särkkä, S. (2018). Iterative Filtering and Smoothing in Nonlinear and Non-Gaussian Systems Using Conditional Moments. *IEEE Signal Processing Letters* **25** 408-412.

[58] Van Der Merwe, R., Doucet, A., De Freitas, N. and Wan, E. (2000). The unscented particle filter. *Advances in neural information processing systems* **13**.

[59] Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual.* CreateSpace, Scotts Valley, CA.

[60] Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium* 153–158. IEEE.

[61] Whiteley, N. (2010). Discussion on particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B* **72** 306–307.

[62] Yaghoobi, F., Corenflos, A., Hassan, S. and Särkkä, S. (2021). Parallel iterated extended and sigma-point Kalman smoothers. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 5350–5354. IEEE.

[63] Yaghoobi, F., Corenflos, A., Hassan, S. and Särkkä, S. (2022). Parallel square-root statistical linear regression for inference in nonlinear

state-space models. *arXiv preprint arXiv:2207.00426*.

Publication VI

Adrien Corenflos and Hany Abdulsamad. Variational Gaussian filtering via Wasserstein gradient flows. In *Proceedings of the 31st European Signal Processing Conference (EUSIPCO)*, Helsinki, Finland, Pages 1838–1842, September 2023.

Variational Gaussian filtering via Wasserstein gradient flows

Adrien Corenflos*, Hany Abdulsamad†
Department of Electrical Engineering and Automation
Aalto University, Finland
Email: *adrien.corenflos@aalto.fi, †hany.abulsamad@aalto.fi

***Abstract*—We present a novel approach to approximate Gaussian and mixture-of-Gaussians filtering. Our method relies on a variational approximation via a gradient-flow representation. The gradient flow is derived from a Kullback–Leibler discrepancy minimization on the space of probability distributions equipped with the Wasserstein metric. We outline the general method and show its competitiveness in posterior representation and parameter estimation on two state-space models for which Gaussian approximations typically fail: systems with multiplicative noise and multi-modal state distributions.**

***Index Terms*—Kalman filtering, variational inference, state-space models, Wasserstein gradient flow.**

I. Introduction

State-space models (or hidden Markov models) are a class of models widely used to represent latent dynamics that are partially or indirectly observed. They typically arise in ecological, economical, and tracking applications [for an introduction, see, e.g. 1]. Formally, state-space models are given by a set of dynamics and noisy observations, often depending on a set of parameters θ

$$\begin{aligned} X_0 &\sim p_0(\cdot \,|\, \theta), \\ X_{k+1} &\sim p_k(\cdot \,|\, X_k, \theta), \\ Y_k &\sim h_k(\cdot \,|\, X_k, \theta). \end{aligned} \tag{1}$$

While the problem of inference in such models is generally intractable, computing the filtering distribution $p(x_k \,|\, y_{0:k})$ can typically be done exactly if the state-space is finite (x_k can only take a finite number of values) or when all the (conditional) densities in (1) are Gaussian using the celebrated Kalman filter [2]. When this is not the case, relying on approximations becomes necessary. Two important types of approximations are approximate Gaussian filters [see, e.g. 3], and Monte Carlo filters [see, e.g. 4].

Although the standard filtering problem is important, one may also be interested in system identification, which, in the parametric context, refers to learning θ from a sequence of observations $\{y_0, \dots, y_K\}$.

In this article, we pay particular attention to two classes of models, typically neglected in approximate Gaussian filtering. The first class is that of models with multiplicative noise, for which stochastic volatility models are an illustrative example, often used in economics to model financial returns [5]. These are usually given as an auto-regressive latent state x_k, and observations y_k following

$$\begin{aligned} Y_k &= \exp(X_k/2)\, \eta_k, \\ X_{k+1} &= \mu + \alpha(X_k - \mu) + \sigma\, \epsilon_k, \end{aligned} \tag{2}$$

where the noise processes are correlated

$$\begin{pmatrix} \epsilon_k \\ \eta_k \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right). \tag{3}$$

The second class we consider are systems for which the state (filtering) posterior is a multi-modal distribution. A simple example of this form can be given by constructing a latent state x_k with random walk dynamics while the observations y_k are a function of the absolute value of x_k:

$$\begin{aligned} X_{k+1} &= X_k + \epsilon_k, \qquad \epsilon_k \sim \mathrm{N}(0, 1), \\ Y_k &= |X_k| + \eta_k, \qquad \eta_k \sim \mathrm{N}(0, 1). \end{aligned} \tag{4}$$

If, for example, $X_0 \sim \mathrm{N}(0, \delta^2)$, it is straightforward to see that the state filtering distribution will be bi-modal and fully symmetric with respect to the x-axis.

A. Contributions

Existing approximate Gaussian filtering methods suffer from several drawbacks. The linearization methods of [6, 7] for example require computing conditional expectations $m_k(x) = \mathbb{E}\left[Y_k \,|\, X_k = x\right]$. In the case of the stochastic volatility model (2), this quantity will unequivocally be null (at least for $\rho = 0$, see Section III-A for details). Consequently, applying these methods to (2) will lead to a filtering (or a smoothing) solution independent of the gathered observations, which is problematic.

Additionally, classical linearization methods do not extend directly to multi-modal distributions, and they do not, to the best of our knowledge, enable handling mixtures of Gaussians. In view of this, our contributions are the following:

1) We rephrase the filtering problem as an iterative distribution fitting problem.
2) We apply the variational inference method from [8] to propagate Gaussian approximations between time steps and formulate it as a fixed point iteration for efficient gradient calculation.
3) We use our method for parameter estimation in stochastic volatility models and filtering of multi-model target distributions.

Adrien Corenflos is funded by the Academy of Finland project 321891 (ADAFUME). Hany Abdulsamad is funded by the Finnish Center for Artificial Intelligence (FCAI).

II. Methodology

A. Variational Inference via Wasserstein Gradient-flows

Let $\pi(x) \propto \exp(-V(x))$ be an arbitrary target distribution known up to a normalizing constant. Given a variational family of distributions, $q_\phi(x)$, $\phi \in \Phi$, and a measure of discrepancy $L(\phi) = D(\pi, q_\phi)$ between π and q_ϕ, it is natural to try and approximate π with q_ϕ by minimizing L.

A typical discrepancy is given by the Kullback–Leibler (KL) divergence [9]

$$L(\phi) = \mathrm{KL}(q_\phi \,|\, \pi) := \int q_\phi(x) \log \frac{q_\phi(x)}{\pi(x)} \,\mathrm{d}x. \tag{5}$$

This divergence presents a number of attractive properties for statistical inference: (i) it is positive, (ii) it only requires evaluating $V(x)$ and does not necessitate knowing the normalizing constant of π, and (iii), it is exact in the sense that $\mathrm{KL}(q_\phi \,|\, \pi) = 0$ if and only if $q_\phi = \pi$. For more details, we refer the reader to [10].

For example, if $q_\phi(x) = \mathrm{N}(x \,|\, \mu, \Sigma)$ is in the family of well-defined Gaussians, parameterized by their mean and covariance, then the inference problem can be cast as a minimization of $L(\cdot)$ with respect to (μ, Σ). Furthermore, when ignoring the positivity constraints on Σ, it is possible to define a gradient flow on $\phi = (\mu, \Sigma)$, akin to a gradient descent on $L(\cdot)$ in continuous time

$$\frac{\mathrm{d}\phi_t}{\mathrm{d}t} = -\nabla_\phi L(\phi_t), \tag{6}$$

which, under an assumption of convexity, will converge to the minimizer of the objective $L(\cdot)$ [11].

While this procedure is correct in essence, it targets the problem indirectly by first assuming an arbitrary parameterization of the model which may or may not respect convexity. A more direct approach is to fit q_ϕ to π in terms of a minimization problem over the space of probability distributions, where we want to minimize $L(q) = D(\pi, q)$. In this case, it is possible to define an analog to the gradient flow (6) by equipping the space of probability distributions with the Wasserstein distance [12, Chap. 6]. Under this metric, we can define a trajectory of probability distributions $q_t(x) \in \mathcal{P}(\mathbb{R}^d)$ via a partial differential equation

$$\frac{\partial q_t(x)}{\partial t} = \nabla \cdot \left[q_t(x) \nabla \log \frac{q_t(x)}{\pi(x)} \right], \tag{7}$$

where $\nabla\cdot$ is the divergence operator, expressed in Euclidean coordinates.

Interestingly, by restricting q_t to represent a Gaussian distribution, it was shown in [8], following [13], that (7) can be reformulated into coupled ordinary differential equations (ODEs) on the mean μ_t and covariance Σ_t of q_t

$$\begin{aligned} \frac{\mathrm{d}\mu_t}{\mathrm{d}t} &= -\mathbb{E}\left[\nabla V(Z_t)\right] \\ \frac{\mathrm{d}\Sigma_t}{\mathrm{d}t} &= 2\, I - \mathbb{E}\left[\nabla V(Z_t) \otimes (Z_t - \mu_t)\right] \\ &\quad - \mathbb{E}\left[(Z_t - \mu_t) \otimes \nabla V(Z_t)\right], \end{aligned} \tag{8}$$

where I is the identity matrix with dimensions $d \times d$ and $Z_t \sim \mathrm{N}(\mu_t, \Sigma_t)$ is Gaussian. Now, provided that we can compute or approximate the expectations arising in (8), we can find a minimizer $q^*(x) = \mathrm{N}(x \,|\, m, P)$ of $L(\cdot)$ by integrating these coupled ODEs until convergence.

B. Filtering as Variational Inference

The filtering problem involves inferring the posterior distribution $p(x_k \,|\, y_{0:k})$ for each time step k. To do so, it is often possible to rely on the familiar *innovation-prediction* decomposition [3, Chap. 4]

$$\begin{aligned} p(x_k \,|\, y_{0:k}) &\propto p(y_k \,|\, x_k) p(x_k \,|\, y_{0:k-1}), \\ p(x_k \,|\, y_{0:k-1}) &= \int p(x_k \,|\, x_{k-1}) p(x_{k-1} \,|\, y_{0:k-1}) \,\mathrm{d}x_{k-1}. \end{aligned} \tag{9}$$

For simplicity, we will focus on state-space models with affine Gaussian transition models

$$p(x_k \,|\, x_{k-1}) = \mathrm{N}(x_k \,|\, A_{k-1}\, x_{k-1} + b_{k-1}, Q_{k-1}), \tag{10}$$

covering those previously presented in (2) and (4).

In that case, assuming a Gaussian approximation is available for the previous time step $k-1$,

$$p(x_{k-1} \,|\, y_{0:k-1}) \approx \mathrm{N}(x_{k-1} \,|\, m_{k-1}, P_{k-1}), \tag{11}$$

then the *prediction* step of the filter leads to another Gaussian distribution $p(x_k \,|\, y_{0:k-1}) \approx \mathrm{N}(x_k \,|\, \bar{m}_k, \bar{P}_k)$ with mean and covariance

$$\begin{aligned} \bar{m}_k &= A_{k-1}\, m_{k-1} + b_{k-1}, \\ \bar{P}_k &= A_{k-1}\, P_{k-1}\, A_{k-1}^\top + Q_{k-1}. \end{aligned} \tag{12}$$

Given this predictive distribution, the *innovation* step computes the approximate filtering distribution

$$p(x_k \,|\, y_{0:k}) \propto \exp(-V(x_k)), \tag{13}$$

where the potential function $V(x_k)$ is given as

$$V(x_k) = -\log p(y_k \,|\, x_k) - \log \mathrm{N}(x_k \,|\, \bar{m}_k, \bar{P}_k). \tag{14}$$

Consequently, in order to find the parameters of a variational Gaussian $\mathrm{N}(x_k \,|\, m_k, P_k)$ that approximates the posterior $p(x_k \,|\, y_{0:k})$, it suffices to follow the recipe from Section II-A and integrate (8) up to stationarity, starting from the predictive parameters $(\bar{m}_k, \bar{P}_k)$, or any other approximation of (m_k, P_k).

When the transition $p(x_k \,|\, x_{k-1})$ is not Gaussian, it is possible to use a similar variational approach to propagate the Gaussian approximation $p(x_{k-1} \,|\, y_{0:k-1}) \approx \mathrm{N}(x_{k-1} \,|\, m_{k-1}, P_{k-1})$ through the non-Gaussian dynamics and approximate the marginal $p(x_k \,|\, y_{0:k-1})$ with another Gaussian $\mathrm{N}(x_k \,|\, \bar{m}_k, \bar{P}_k)$. This method was used in [13] to propagate the Gaussian approximation through dynamics defined by a stochastic differential equation. Combining this with our approach is, therefore, *de facto* possible. However, due to the limited scope, we leave this aspect for future work and only consider Gaussian dynamics in the remainder of this article.

Finally, because the likelihood of the observations is given by $p(y_{0:k}) = p(y_{0:k-1}) \int p(y_k \,|\, x_k) p(x_k \,|\, y_{0:k-1}) \,\mathrm{d}x_k$, it is easy to derive an approximation of the marginal log-likelihood of the model by recursion. That is because the quantity $\ell_k = \int p(y_k \,|\, x_k) p(x_k \,|\, y_{0:k-1}) \,\mathrm{d}x_k$ is evaluated as part of (8), and can therefore be used to provide an online estimate of the log-likelihood increments, which, in turn, can be used, for example, in a system identification scenario. We return to this point in Sections II-D and III-A.

C. *The Multi-modal Filtering Case*

To generalize the variational technique from the previous section, let us suppose that the filtering distribution at time $k-1$ is instead given by a mixture density

$$p(x_{k-1} \mid y_{0:k-1}) = \sum_{i=1}^{N_i} w^{(i)} \, \mathrm{N}(x_{k-1} \mid m_{k-1}^{(i)}, P_{k-1}^{(i)}), \tag{15}$$

with $w^{(i)} = 1/N_i, \forall i \in [1, N_i]$. In this case, when the transition dynamics are Gaussian, it is easy to show that

$$p(x_k \mid y_{0:k-1}) = \sum_{i=1}^{N_i} w^{(i)} \, \mathrm{N}(x_k \mid \bar{m}_k^{(i)}, \bar{P}_k^{(i)}), \tag{16}$$

where $\forall i \in [1, N_i]$

$$\begin{aligned} \bar{m}_k^{(i)} &= A_{k-1} \, m_{k-1}^{(i)} + b_{k-1}, \\ \bar{P}_k^{(i)} &= A_{k-1} \, P_{k-1}^{(i)} \, A_{k-1}^\top + Q_{k-1}. \end{aligned} \tag{17}$$

These updates correspond to a tractable *prediction* step. As a consequence, we only need to understand how to perform the *innovation* step to arrive at the posterior $p(x_k \mid y_{0:k})$.

Conveniently, [8] also shows that the duality between the gradient flow (7) and the coupled ODEs (8) extends to the case of the finite variational mixture of Gaussians

$$q_t(x) = \sum_{i=1}^{N_i} \omega^{(i)} q_t^{(i)}(x) = \sum_{i=1}^{N_i} \omega^{(i)} \, \mathrm{N}(x \mid \mu_t^{(i)}, \Sigma_t^{(i)}), \tag{18}$$

with $\omega^{(i)} = 1/N_i, \forall i \in [1, N_i]$. In this case, rather than a pair of ODEs, we obtain a system of such ODEs

$$\begin{aligned} \frac{\mathrm{d}\mu_t^{(i)}}{\mathrm{d}t} &= -\mathbb{E}\left[\nabla \log \frac{q_t}{\pi}(Z_t^{(i)})\right] \\ \frac{\mathrm{d}\Sigma_t^{(i)}}{\mathrm{d}t} &= -\,\mathbb{E}\left[\nabla^2 \log \frac{q_t}{\pi}(Z_t^{(i)})\right] \Sigma_t^{(i)} \\ &\quad - \Sigma_t^{(i)} \mathbb{E}\left[\nabla^2 \log \frac{q_t}{\pi}(Z_t^{(i)})\right], \end{aligned} \tag{19}$$

where, for all i, $Z_t^{(i)} \sim \mathrm{N}(\mu_t^{(i)}, \Sigma_t^{(i)})$ is Gaussian, and ∇^2 denotes the Hessian operator.

This result means that, given a Gaussian mixture approximation of the predictive $p(x_k \mid y_{1:k-1})$, we can obtain a Gaussian mixture approximation of $p(x_k \mid y_{0:k})$ by integrating (19) and following a similar approach to Section II-B.

D. *Numerical Considerations and Implementation*

In practice, the integrals arising in (8) and (19) are not available in closed form. Therefore, we need to resort to numerical integration. This can be done by using any form of deterministic or stochastic Gaussian quadrature, for example, Monte Carlo [see, e.g., 14] or sigma-points [3, see, e.g.,] methods. The two approaches have their pros and cons: Monte Carlo will give the correct solution on average, while deterministic quadrature is bound to be biased. Nonetheless, deterministic rules are sometimes more practical when no stochasticity should be allowed in the system. Whichever is chosen, we will obtain an approximation

$$\frac{\mathrm{d}\mu_t}{\mathrm{d}t} \approx F_m(\mu_t, \Sigma_t), \quad \frac{\mathrm{d}\Sigma_t}{\mathrm{d}t} \approx F_P(\mu_t, \Sigma_t), \tag{20}$$

of (8) for the choice of $V(x_k)$ given by (14). Moreover, the same approximation scheme can be used to compute log-likelihood increments $\log \mathbb{E}\left[p(y_k \mid x_k)\right]$ under the predictive $\mathrm{N}(x_k \mid \bar{m}_k, \bar{P}_k)$. In Section III, we use Gauss–Hermite [see, e.g., 3, Chap. 5] quadrature integration rules with order five.

We now assume that we have chosen an integration method I for which the stationary solution of (20) is a fixed point: $(m_k, P_k) = I(m_k, P_k)$. This can, for example, be an Euler integration step with a small step size. This fixed-point perspective allows us to leverage the implicit function theorem and bypass the loop when computing gradients for system identification. For the sake of brevity, we omit the details here and refer to [15] for the method and to our code[1] for a Python implementation.

Therefore, the final algorithm for variational uni-modal filtering from Section II-B is given by Algorithm 1. Due to space constraints, we do not reproduce the algorithm for the multi-modal case from Section II-C: the procedure follows the same steps, albeit for a larger system of ODEs.

Algorithm 1 Uni-modal Wasserstein Gradient-flow Filter

```
 1: input: Measurements $y_{0:K}$ and prior $(m_0, P_0)$
 2: output: Filtering posterior distributions $(m_{1:K}, P_{1:K})$ and
    marginal log-likelihood $\ell = \log p(y_{0:K})$
 3: Set $\ell \leftarrow 0$
 4: for $k \leftarrow 1$ to $K$ do
 5:     Set $\ell_k \leftarrow \log \mathbb{E}_{x \sim \mathrm{N}(\bar{m}_k, \bar{P}_k)} p(y_k \mid x_k)$
 6:     Set $\ell \leftarrow \ell + \ell_k$                         ▷ Log-likelihood
 7:     $(m_k, P_k) \leftarrow (\bar{m}_k, \bar{P}_k)$
 8:     while Not converged do                          ▷ Innovation
 9:         $(m_k, P_k) \leftarrow I(m_k, P_k)$
10:     end while
11:     Set $\bar{m}_{k+1} \leftarrow A_k \, m_k + b_k$                ▷ Prediction
12:     Set $\bar{P}_{k+1} \leftarrow A_k \, P_k \, A_k^\top + Q_k$
13: end for
14: return $(m_{1:K}, P_{1:K})$, $\ell$
```

III. Empirical Evaluation

As discussed in Section I, introducing our method is motivated by the problems posed by multiplicative noise and multi-modality in approximate Gaussian filtering. Consequently, we aim to demonstrate its effectiveness on these two problems. We compare the variational Wasserstein filter (VWF) to a bootstrap particle filter (PF, [16]) using 500 particles and the *continuous* resampling scheme from [17]. This resampling allows using the particle filter in a standard parameter estimation

[1]Implementation of the fixed-point iteration is available under https://github.com/hanyas/wasserstein-flow-filter/blob/master/wasserstein_filter/utils.py

scenario. The code to reproduce these experiments can be found at https://github.com/hanyas/wasserstein-flow-filter.

A. *Stochastic Volatility with Leverage*

First, we consider the stochastic volatility model as given by (2) and (3). Because the noises ϵ_k and η_k are correlated, we need to construct the joint distribution over X_k and its generating noise ϵ_k and form the augmented two-dimensional state $\zeta_k = \begin{bmatrix} X_k & \epsilon_k \end{bmatrix}^\top$. The dynamics of ζ_k are then given by

$$\begin{pmatrix} X_{k+1} \\ \epsilon_{k+1} \end{pmatrix} = \begin{pmatrix} \alpha & \sigma \\ 0 & 0 \end{pmatrix} \begin{pmatrix} X_k \\ \epsilon_k \end{pmatrix} + \begin{pmatrix} \mu(1-\alpha) \\ 0 \end{pmatrix} + q_k \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad (21)$$

where q_k is a standard Gaussian random variable. With these dynamics, we have the following observation model

$$Y_k = \exp(X_k/2)(\rho\epsilon_k + \sqrt{1-\rho^2}\, r_k), \quad (22)$$

with r_k being a standard Gaussian random variable. Contrary to the original form of the model, the noise processes are now de-correlated so that we can apply our method to the 2-dimensional system defined by ζ_k.

To perform an empirical comparison, we simulate the model and record three trajectories with $K = 1\,000$, $K = 1\,500$, and $K = 2\,000$ observations. We use parameters $\alpha_* = 0.975$, $\mu_* = 0.5$, $\sigma_*^2 = 0.02$, and $\rho_* = -0.8$. These values correspond to those in [17], typical of a standard stock market.

To illustrate the problem of using methods based on linearizing the conditional observation mean $\mathbb{E}[Y_k \mid X_k, \epsilon_k]$, we evaluate the marginal log-likelihood of the data approximated by an extended Kalman filter (EKF) targeting (21) with parameters $(\alpha_*, \mu_*, \sigma_*)$ while varying levels of correlation ρ and number of observations K. The resulting curves for the particle filter of [17], our method, and the extended Kalman filter are shown in Figure 1. As the (model) correlation ρ between the two noise-generating processes decreases, the predictive value of the observations becomes negligible. As a consequence, the marginal likelihood, as approximated by the EKF, will struggle to capture the right level of correlation. This means that calibrating the model using an extended (or similar) Kalman filter will result in an inconsistent estimate, at least for the parameter ρ.

To further confirm this hypothesis, we follow [17] and perform joint maximum likelihood parameter estimation given $K = 1\,000$ observations. We compare our method to theirs and to an extended Kalman filter. The likelihood is optimized in all cases using the gradient of the log-likelihood, delivered by automatic reverse differentiation. In the particular case of the variational Wasserstein filter, differentiation is made efficient by our fixed point formulation of the variational calibration.

For a statistical comparison, we repeat this experiment over ten trials associated with different random seeds and, therefore, ten distinct sets of observations. We report the mean plus or minus one standard deviation of the parameter estimates for each algorithm. However, because the particle filter is inherently a stochastic method, even for a fixed set of observations, we perform 25 *sub-trials* per set of observations and use the median as a parameter estimate per trial. The

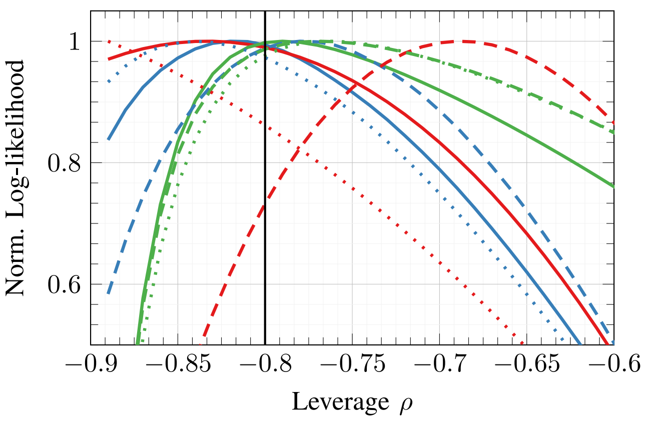


Fig. 1. Comparing the (normalized) marginal log-likelihood as a function of the leverage parameter ρ in a stochastic volatility model. We plot the estimates as returned by an extended Kalman filter (red), a bootstrap particle filter (blue), and a variational Wasserstein filter (green). The dotted, dashed, and solid lines correspond to different numbers of observations $K = 1\,000$, $K = 1\,500$, and $K = 2\,000$. The true value is $\rho = -0.8$ (vertical line). The variational Wasserstein and particle filters deliver consistent approximations independent of the data size, while the extended Kalman filter does not.

TABLE I
STATISTICS FOR PARAMETER ESTIMATION OF THE STOCHASTIC VOLATILITY MODEL. RESULTS ARE AVERAGED OVER TEN TRIALS.

	μ	α	σ	ρ
True	0.50	0.975	0.14	-0.80
PF [17]	0.56 (±0.08)	0.972 (±0.007)	0.15 (±0.02)	−0.85 (±0.06)
VWF	0.56 (±0.07)	0.972 (±0.009)	0.15 (±0.02)	−0.80 (±0.04)
EKF	0.69 (±0.33)	0.780 (±0.590)	0.21 (±0.25)	−0.58 (±0.54)

results are given in Table I and confirm our intuition. The extended Kalman filter provides inconsistent solutions, while our method delivers estimates comparable to the estimates of the particle filter.

B. *Multi-Modal Example*

We now turn to the multi-modal motivating example (4). We simulate $K = 500$ observations from the model, where we have taken the variance at origin to be $\delta^2 = 1$. We then perform variational filtering using the mixture version of Algorithm 1 with $N_i = 2$ mixture components.

Our goal is a correct representation of the filtering posterior. Thus we will not assess the performance in terms of root mean square error, which is inappropriate for such problems. Instead, we qualitatively report the resulting filtering distributions in Figure 2. There is visually hardly any difference between the particle filtering estimate and our method.

One caveat to this positive result is that the bi-modal variational Wasserstein filter tends to collapse when the two modes are too close to each other. It is still unclear whether this is a feature of the general method or the ODE solver, and further investigations are warranted.

IV. Conclusion

In this article, we have presented a novel approach for approximate Gaussian filtering, which avoids the reliance on

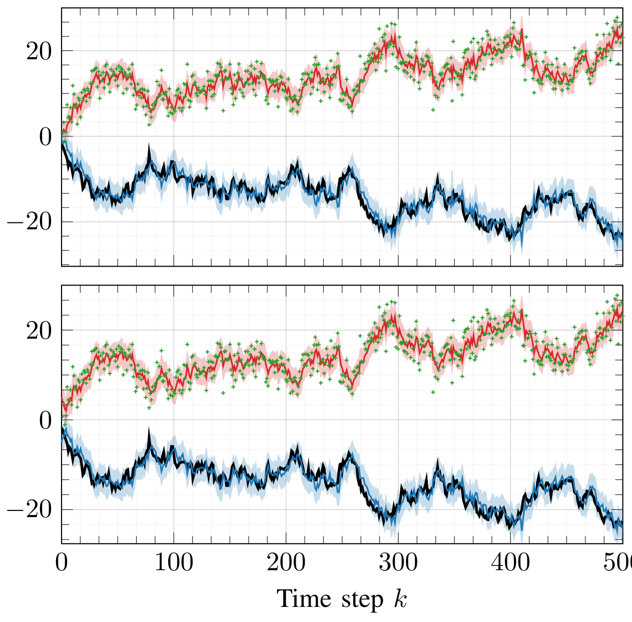


Fig. 2. Performing filtering on a multi-modal dynamical system. We compare the filtering result of a particle filter (top) with that of a variational Wasserstein filter equipped with a mixture-of-Gaussians posterior representation (bottom). Both filters capture the bi-modal posterior distribution (red and blue) inferred from the observations (green) induced by the true states (black).

enabling assumptions [6], which are usually not amenable to multiplicative noise and not extendable to multi-modal distributions. Several questions remain open:

a) Assumptions: We have assumed, for simplicity, that the transition dynamics are governed by affine Gaussian densities. Overcoming this limitation is an interesting research question. In fact, in [13], the coupled ODEs (8) were originally introduced to propagate the Gaussian through a nonlinear stochastic differential equation.

b) Numerics: The numerical scheme chosen here is, primarily for ease of exposition, different from that of [8], which uses an iterative method called JKO [11] after [18], instead of integrating the differential equation (8) directly. It is unclear which approach is the best fit for filtering applications.

c) Mixture weights: The weights of the mixture in the ODE (20) are not allowed to vary. This modeling restriction needs to be relaxed. However, that may lead to identifiability issues (for example, a single Gaussian can be represented with a mixture of two Gaussians with different weights in infinite ways). Furthermore, introducing time-varying weights should be done carefully.

V. Individual Contributions

The original idea and redaction of the article are due to Adrien Corenflos. Both authors contributed to the design of the methodology. The implementation and experiments are due to Hany Abulsamad.

References

[1] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Courier Corporation, 2007.

[2] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, 1960.

[3] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[4] N. Chopin and O. Papaspiliopoulos, *An Introduction to Sequential Monte Carlo*. Springer, 2020.

[5] S. L. Heston, “A closed-form solution for options with stochastic volatility with applications to bond and currency options,” *The Review of Financial Studies*, 1993.

[6] A. F. Garcìa-Fernàndez, L. Svensson, M. R. Morelande, and S. Särkkä, “Posterior linearization filter: Principles and implementation using sigma points,” *IEEE Transactions on Signal Processing*, 2015.

[7] F. Tronarp, Á. F. García-Fernández, and S. Särkkä, “Iterative filtering and smoothing in nonlinear and non-Gaussian systems using conditional moments,” *IEEE Signal Processing Letters*, 2018.

[8] M. Lambert, S. Chewi, F. Bach, S. Bonnabel, and P. Rigollet, “Variational inference via Wasserstein gradient flows,” *arXiv preprint arXiv:2205.15902*, 2022.

[9] J. M. Joyce, “Kullback–Leibler divergence,” in *International Encyclopedia of Statistical Science*. Springer, 2011.

[10] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.

[11] F. Santambrogio, “Euclidean, metric, and Wasserstein gradient flows: An overview,” *Bulletin of Mathematical Sciences*, 2017.

[12] C. Villani, *Optimal Transport: Old and New*. Springer Berlin Heidelberg, 2009.

[13] S. Särkkä, “On unscented Kalman filtering for state estimation of continuous-time nonlinear systems,” *IEEE Transactions on Automatic Control*, 2007.

[14] G. Pagès, *Numerical Probability*. Springer, 2018.

[15] B. Christianson, “Reverse accumulation and attractive fixed points,” *Optimization Methods & Software*, 1994.

[16] N. J. Gordon, D. J. Salmond, and A. F. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” in *IEEE Proceedings of Radar and Signal Processing*, 1993.

[17] S. Malik and M. K. Pitt, “Particle filters for continuous likelihood evaluation and maximisation,” *Journal of Econometrics*, 2011.

[18] R. Jordan, D. Kinderlehrer, and F. Otto, “The variational formulation of the Fokker–Planck equation,” *SIAM Journal on Mathematical Analysis*, 1998.

Publication VII

Adrien Corenflos and Axel Finke. Particle-MALA and Particle-mGRAD: Gradient-based MCMC methods for high-dimensional state-space models. Submitted to *Journal of Machine Learning Research*, January 2024.

Journal of Machine Learning Research 23 (2022) 1-67 Submitted 1/21; Revised 5/22; Published 9/22

Particle-MALA and Particle-mGRAD: Gradient-based MCMC methods for high-dimensional state-space models

Adrien Corenflos ADRIEN.CORENFLOS@AALTO.FI
Department of Electrical Engineering and Automation
Aalto University, Finland

Axel Finke A.FINKE@LBORO.AC.UK
Department of Mathematical Sciences
Loughborough University, UK

Editor: My editor

Abstract

State-of-the-art methods for Bayesian inference in state-space models are (a) *conditional sequential Monte Carlo (CSMC)* algorithms; (b) sophisticated 'classical' gradient-based Markov chain Monte Carlo (MCMC) algorithms like *Metropolis-adjusted Langevin algorithm (MALA)*, the *preconditioned Crank–Nicolson–Langevin (PCNL)*, or the *marginal gradient (mGRAD)* algorithm recently introduced in Titsias and Papaspiliopoulos (2018). The former propose N particles at each time step to exploit the model's 'decorrelation-over-time' property and scale favourably with the time horizon, T, but break down when the dimension of the latent states, D, increases. The latter leverage gradient-/prior-informed local proposals to scale favourably with D but exhibit sub-optimal scalability with T due to a lack of model-structure exploitation. We introduce methods combining both approaches. The first, *particle MALA (Particle-MALA)*, spreads N particles around the current state using gradient information, extending MALA to $T > 1$ time steps and $N > 1$ proposals. The second, *particle marginal gradient (Particle-mGRAD)*, additionally incorporates (conditionally) Gaussian prior dynamics into the proposal, extending mGRAD to $T, N > 1$. Particle-mGRAD provably resolves the 'tuning problem' of choosing between CSMC (superior for informative dynamics) and Particle-MALA (superior for uninformative dynamics). We similarly extend other 'classical' MCMC approaches like *auxiliary MALA*, *auxiliary gradient (aGRAD)*, and *PCNL*. In experiments, our methods substantially improve upon CSMC and 'classical' MCMC approaches.

Keywords: Sequential Monte Carlo, Particle MCMC, MALA, Markovian models

1 Introduction

1.1 Feynman–Kac models

The aim of this work is to construct efficient *Markov chain Monte Carlo (MCMC)* updates for sampling from a continuous *joint smoothing* distribution $\pi_T(\mathbf{x}_{1:T})$ on $\mathcal{X}^T$, where $\mathcal{X} := \mathbb{R}^D$

and where for any $t \leq T$, we have defined the following distributions (termed *filters*):

$$\pi_t(\mathbf{x}_{1:t}) \propto \prod_{s=1}^{t} Q_s(\mathbf{x}_{s-1:s}). \tag{1}$$

Here, $Q_t(\mathbf{x}_{t-1:t}) > 0$ is differentiable and can be evaluated point-wise. Throughout this work, we use the convention that quantities with 'time' subscripts $t \leq 0$ or $t > T$ should be ignored, so that, e.g., $Q_1(x_{0:1}) \equiv Q_1(\mathbf{x}_1)$ and $Q_{T+1}(\mathbf{x}_{T:T+1}) \equiv 1$. We will frequently work with some decomposition

$$Q_t(\mathbf{x}_{t-1:t}) = M_t(\mathbf{x}_t|\mathbf{x}_{t-1})G_t(\mathbf{x}_{t-1:t}),$$

such that

- $M_t(\,\cdot\,|\mathbf{x}_{t-1})$ is a density (w.r.t. a suitable version of the Lebesgue measure) and also defines a Markov transition kernel called *mutation kernel*;
- $G_t(\mathbf{x}_{t-1:t}) > 0$ is called *potential function.*

We assume that these densities and potential functions are differentiable and that they (as well as their gradients) can be evaluated point-wise. Motivated by the following example, we will sometimes refer to $M_{1:T}(\mathbf{x}_{1:T}) \coloneqq \prod_{t=1}^{T} M_t(\mathbf{x}_t|\mathbf{x}_{t-1})$ as the *prior dynamics* of the *latent states* $\mathbf{x}_{1:T}$ and $G_{1:T}(\mathbf{x}_{1:T}) \coloneqq \prod_{t=1}^{T} G_t(\mathbf{x}_{t-1:t})$ as the *likelihood.*

Example 1 (state-space model) *One important special case of Feynman–Kac models are* state-space models. *A state-space model is a bivariate Markov chain* $(\mathbf{x}_t, \mathbf{y}_t)_{t\geq 1}$ *on* $\mathcal{X} \times \mathcal{Y}$*, where* $\mathcal{X} \coloneqq \mathbb{R}^D$ *and* $\mathcal{Y} \coloneqq \mathbb{R}^{D'}$*, with initial density* $p(\mathbf{x}_1, \mathbf{y}_1) = f_1(\mathbf{x}_1)g_1(\mathbf{y}_1|\mathbf{x}_1)$ *and transition densities* $p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{x}_{t-1}) = f_t(\mathbf{x}_t|\mathbf{x}_{t-1})g_t(\mathbf{y}_t|\mathbf{x}_t)$ *(w.r.t. a suitable version of the Lebesgue measure). State-space models assume that only the measurements* $\mathbf{y}_{1:T}$ *can be observed whilst the Markov chain* $(\mathbf{x}_t)_{t\geq 1}$ *(often representing the evolution of the phenomenon of interest) is latent. The joint smoothing distribution then encodes our knowledge of the latent states* $\mathbf{x}_{1:T}$ *given the available data* $\mathbf{y}_{1:T}$*:*

$$\pi_T(\mathbf{x}_{1:T}) \coloneqq p(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \propto \prod_{t=1}^{T} f_t(\mathbf{x}_t|\mathbf{x}_{t-1})g_t(\mathbf{y}_t|\mathbf{x}_t).$$

One possible way of casting such a state-space model as a Feynman–Kac model (there are others) is then to take $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = f_t(\mathbf{x}_t|\mathbf{x}_{t-1})$ *and* $G_t(\mathbf{x}_{t-1:t}) = g_t(\mathbf{y}_t|\mathbf{x}_t)$*. In this case,* $M_{1:T}(\mathbf{x}_{1:T}) = p(\mathbf{x}_{1:T})$*,* $G_{1:T}(\mathbf{x}_{1:T}) = p(\mathbf{y}_{1:T}|\mathbf{x}_{1:T})$*, and* $\pi_t(\mathbf{x}_{1:t}) = p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$*, for* $t \leq T$*.*

1.2 Sampling the latent states

Performing inference about the latent states $\mathbf{x}_{1:T}$ requires calculating expectations of the form $\mathbb{E}_{\mathbf{x}_{1:T}\sim\pi_T}[\varphi(\mathbf{x}_{1:T})]$, for some integrable test function $\varphi\colon \mathcal{X}^T \to \mathbb{R}$. Unfortunately, such expectations do not admit closed-form expressions in most realistic problems and must be approximated by some Monte Carlo estimate $\frac{1}{I}\sum_{i=1}^{I}\varphi(\mathbf{x}_{1:T}[i])$ using samples $(\mathbf{x}_{1:T}[i])_{i=1}^{I}$ (approximately) distributed according to π_T. These often come from some MCMC algorithm targeting π_T.

'Classical' MCMC methods. Unfortunately, simple MCMC approaches like the *independent Metropolis–Hastings (IMH)* algorithm (Hastings, 1970) perform poorly if the problem size: $D \times T$, is large due the difficulty of constructing efficient *global* (a.k.a. *independent*) proposal distributions in high dimensions. To circumvent this difficulty, MCMC algorithms with *local* moves like the *random-walk Metropolis (RWM)* algorithm (Metropolis et al., 1953), propose a new state of the Markov chain near the current state. By decreasing the proposal scale at a suitable rate with the problem size, the RWM algorithm can circumvent this curse of dimension (Roberts et al., 1997). Further improved performance can be achieved by exploiting

- *gradient information,* i.e. by including gradients of the log-likelihood or log-target density into the proposal as in the *Metropolis-adjusted Langevin algorithm (MALA)* from Besag (1994) and in the *auxiliary MALA (aMALA)* from Titsias and Papaspiliopoulos (2018); and *additionally*
- *prior information,* i.e. by explicitly incorporating the prior dependence structure into the proposal as in the *preconditioned Crank–Nicolson–Langevin (PCNL)* and related algorithms (see, e.g., Cotter et al., 2013, and references therein) or in the *marginal gradient (mGRAD)* and *auxiliary gradient (aGRAD)*[1] algorithms from Titsias (2011); Titsias and Papaspiliopoulos (2018).

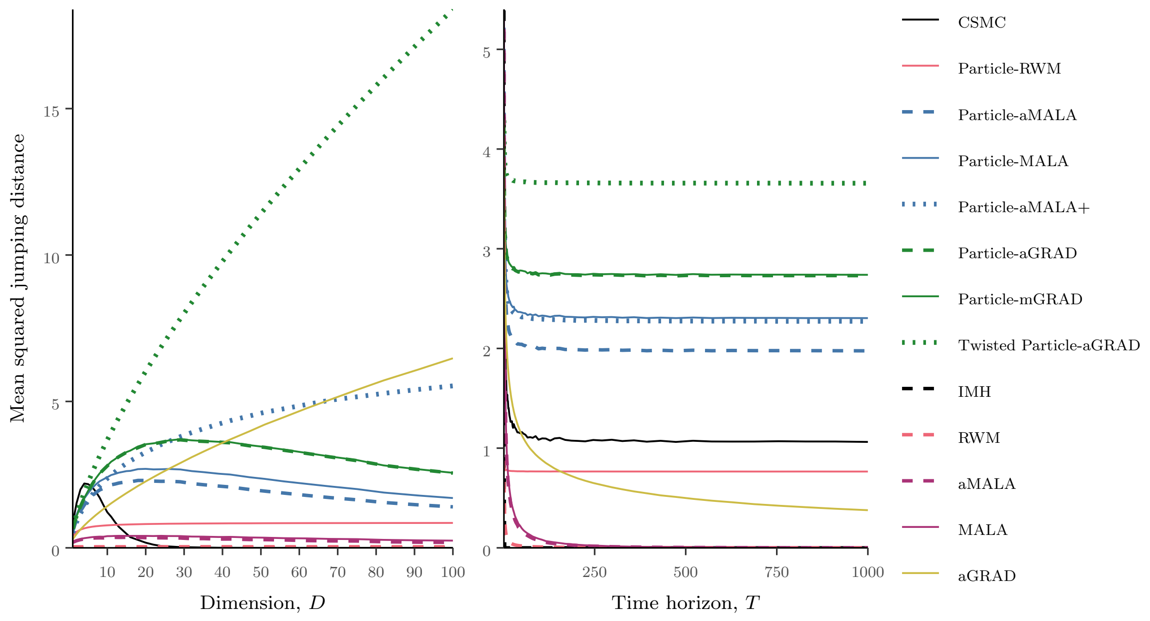
Figure 1a illustrates that 'classical' MCMC algorithms can scale favourably with D.

However, 'classical' MCMC algorithms are agnostic to the 'decorrelation-over-time' structure of the target distribution $\pi_T(\mathbf{x}_{1:T})$, i.e., to the fact that, for suitably regular models, the correlation of x_s and $\mathbf{x}_t$ under $\pi(\mathbf{x}_{1:T})$ decays with $|t-s|$. For example, for the simple RWM algorithm and MALA, the *step size* $\delta > 0$ (i.e., proposal variance) would need to decrease at a suitable rate with T ($\delta \in \mathrm{O}((DT)^{-1})$ and $\delta \in \mathrm{O}((DT)^{-1/3})$, respectively) even if the model was completely independent across time steps (Roberts and Rosenthal, 2001). Therefore, it stands to reason that the scaling of 'classical' MCMC methods like MALA, PCNL or mGRAD/aGRAD with the time horizon T could be improved by empowering them to exploit this model structure.

CSMC methods. Another popular π_T-invariant MCMC-kernel, P_{CSMC}, is induced by running the *CSMC* algorithm proposed in the seminal work Andrieu et al. (2010); Whiteley (2010). Given the current state $\mathbf{x}_{1:T} \in \mathcal{X}^T$ of the Markov chain (then called the *reference path*) this algorithm generates $\tilde{\mathbf{x}}_{1:T} \sim P_{\text{CSMC}}(\,\cdot\,|\mathbf{x}_{1:T})$ as follows, where we write $[n] \coloneqq \{1, \dots, n\}$ and $[n]_0 \coloneqq [n] \cup \{0\}$:

1. For $t = 1, \dots, T$, sample some index k_t from a uniform distribution on $[N]_0$; set $\mathbf{x}_t^{k_t} \coloneqq \mathbf{x}_t$ and sample the remaining *particles* $\mathbf{x}_t^{-k_t} \coloneqq (\mathbf{x}_t^0, \dots, \mathbf{x}_t^{k_t-1}, \mathbf{x}_t^{k_t+1}, \dots, \mathbf{x}_t^N)$ conditionally independently such that for $n \neq k_t$,
$$\mathbf{x}_t^n \sim M_t(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n}), \tag{2}$$
for *ancestor indices* $a_{t-1}^n \in [N]_0$ whose rôle is explained later.

1. Throughout this work, 'aGRAD' refers more specifically to the 'aGrad-z' algorithm from Titsias and Papaspiliopoulos (2018).



(a) Empirical scaling with D for fixed time horizon $T = 25$.

(b) Empirical scaling with T for fixed state dimension $D = 10$.

Figure 1: Toy linear-Gaussian state-space model with $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{x}_{t-1}, \lambda\mathbf{I})$ and $G_t(\mathbf{x}_{t-1:t}) = \mathrm{N}(\mathbf{y}_t; \mathbf{x}_t, \mathbf{I})$, where $\mathbf{I}$ is the $(D \times D)$-identity matrix and $\lambda = 1$. For a fair comparison, all methods use $N + 1 = 32$ particles. The step sizes are: $(TD)^{-1}$ for (multi-proposal) RWM, D^{-1} for Particle-RWM, $(TD)^{-1/3}$ for (multi-proposal) aMALA/MALA/aGRAD, and $D^{-1/3}$ for the remaining (i.e., new) methods. Panel a illustrates that as D increases, some 'classical' MCMC algorithms (RWM, MALA, aMALA and aGRAD) are stable but the CSMC algorithm breaks down. Conversely, Panel b illustrates that as T increases, the CSMC algorithm is stable in T but all 'classical' MCMC algorithms (IMH, RWM, MALA, aMALA and aGRAD) break down.

2. Return $\tilde{\mathbf{x}}_{1:T} := (\mathbf{x}_1^{l_1}, \dots, \mathbf{x}_t^{l_T})$, for indices $l_1, \dots, l_T \in [N]_0$ sampled from an appropriate distribution.

Informally, the CSMC algorithm can be interpreted as employing T separate accept–reject steps (one at each time point) which allows it to exploit the 'decorrelation-over-time' property of $\pi_T(\mathbf{x}_{1:T})$ (akin to a 'classical' MCMC algorithm with blocking in the 'time' direction as noted by Singh et al. 2017). For suitably regular problems, the CSMC algorithm therefore scales more favourably with T than 'classical' MCMC approaches as illustrated in Figure 1b.

Unfortunately, as shown in Finke and Thiery (2023), the CSMC algorithm suffers from a curse of dimension in the state dimension D: as D increases, it becomes increasingly likely that $\tilde{\mathbf{x}}_{1:T}$ coincides exactly with $\mathbf{x}_{1:T}$, i.e., the induced MCMC chain gets stuck. This is unsurprising because the CSMC algorithm generalises the IMH algorithm to $T > 1$ time steps and $N > 1$ proposals. Indeed, note that (2) is again an independent (i.e. global) proposal in the sense that it does not depend on the time-t component of the current state of the Markov chain, $\mathbf{x}_t$; and such proposals are known to scale poorly with dimension (due to the difficulty of finding efficient global proposals in high dimensions). The only potential remedy: increasing N exponentially with D, is prohibitively costly.

Existing combinations of 'classical' MCMC and CSMC. To circumvent this problem, Finke and Thiery (2023) introduced the *Particle-RWM*[2] algorithm which scatters the particles locally around the reference path (see also Shestopaloff and Neal, 2018; Malory, 2021, for related approaches). That is, conditional on the reference path $\mathbf{x}_{1:T}$, the remaining particles $\mathbf{x}_t^{-k_t}$ are proposed from a joint distribution under which

$$\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{x}_t, \delta_t \mathbf{I}),$$

for $n \neq k_t$, where $\mathbf{I}$ is the $(D \times D)$-identity matrix. As noted in Tjelmeland (2004), sampling from this joint proposal distribution can be achieved by first sampling an auxiliary variable $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t, \frac{\delta_t}{2}\mathbf{I})$ and then $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{u}_t, \frac{\delta_t}{2}\mathbf{I})$, for $n \neq k_t$. Finke and Thiery (2023) also showed that scaling the step size as $\delta_t \in \mathrm{O}(D^{-1})$ (independently of T) guarantees stability in high dimensions. This is again unsurprising because the Particle-RWM algorithm generalises the RWM algorithm with Gaussian proposals (and proposal variance δ_1) to $T > 1$ time steps and $N > 1$ proposals. Recently, Corenflos and Särkkä (2023) showed that the Particle-RWM algorithm can be viewed as a Gibbs-sampling step for the auxiliary variables $\mathbf{u}_t$ followed by a CSMC update which targets a modified Feynman–Kac model which depends on $\mathbf{u}_{1:T}$, allowing for greater flexibility in the choice proposals. Including related 'pseudo observations' $\mathbf{u}_t$ into CSMC updates had previously been suggested by Murray et al. (2013); Fearnhead and Meligkotsidou (2016); Karppinen and Vihola (2021) but primarily aimed at overcoming the problem that the CSMC algorithm mixes poorly if the initial distribution $M_1(\mathbf{x}_1)$ is diffuse (and potentially also for improving mixing in the presence of 'static' model parameters).

1.3 Contributions

Recall that in the 'classical' MCMC setting, improved performance can often be achieved by enhancing the proposal distribution using gradient or prior information. Thus, in this work, we introduce a methodology which combines the strength of CSMC methods (i.e., exploitation of the 'decorrelation-over-time' property of the target distribution) with the strengths of sophisticated 'classical' MCMC approaches (i.e., gradient-enhanced local proposals).

In the remainder of this section, we detail the contributions of this paper (Table 1 summarises our proposed methodology).

In Section 3, we introduce the following CSMC type methods which propose particles locally around the reference path guided by gradient information:

- **Particle-aMALA.** In Section 3.1, we extend the Particle-RWM algorithm to incorporate gradient information into the proposals. That is, conditional on the reference path $\mathbf{x}_{1:T}$, the remaining particles $\mathbf{x}_t^{-k_t}$ are proposed from a joint distribution under which

$$\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{x}_t + \tfrac{\delta_t}{2}\nabla_{\mathbf{x}_t} \log \pi_t(\mathbf{x}_{1:T}), \delta_t \mathbf{I}), \tag{3}$$

2. Referred to as 'random-walk CSMC' therein.

for $n \neq k_t$. Sampling from this joint proposal can be achieved by first sampling an auxiliary variable $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa \frac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log \pi_t(\mathbf{x}_{1:t}), \frac{\delta_t}{2}\mathbf{I})$ and then $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{u}_t, \frac{\delta_t}{2}\mathbf{I})$, for $n \neq k_t$. We call this method *Particle-aMALA* because the auxiliary variables $\mathbf{u}_t$ are explicitly included in the space, i.e. they appear in the particle weights, and because the algorithm generalises a version of *auxiliary MALA (aMALA)* from Titsias and Papaspiliopoulos (2018) to $T > 1$ time steps and $N > 1$ proposals.

- **Particle-MALA.** In Section 3.2, we improve Particle-aMALA by marginalising out the auxiliary variables $\mathbf{u}_t$. We call the resulting method *Particle-MALA* because it generalises *MALA* (Besag, 1994) to $T > 1$ time steps and $N > 1$ proposals.

- **Particle-aMALA+.** In Section 3.3, we improve Particle-aMALA differently by replacing the 'filter' gradient $\nabla_{\mathbf{x}_t} \log \pi_t(\mathbf{x}_{1:t})$ in (3) with the 'smoothing' gradient $\nabla_{\mathbf{x}_t} \log \pi_T(\mathbf{x}_{1:T})$ which is beneficial when future observations are informative about the current state. We call the resulting method *Particle-aMALA+*.

In Section 4, we consider the special case that the Feynman–Kac model has conditionally Gaussian mutation kernels: $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$. In this setting, we introduce the following methods which propose particles locally around the reference path guided by both gradient information and prior information:

- **Particle-aGRAD.** In Section 4.1, we propose an algorithm which, conditional on the reference path $\mathbf{x}_{1:T}$, proposes the remaining particles $\mathbf{x}_t^{-k_t}$ from a joint distribution under which

$$\mathbf{x}_t^n \sim \mathrm{N}\big((\mathbf{I} - \mathbf{A}_t(\mathbf{x}_{t-1}^{a_{t-1}^n}))\mathbf{m}_t(\mathbf{x}_{t-1}^{a_{t-1}^n}) + \mathbf{A}_t(\mathbf{x}_{t-1}^{a_{t-1}^n})[\mathbf{x}_t + \tfrac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log G_t(\mathbf{x}_{t-1:t})], \mathbf{B}_t(\mathbf{x}_{t-1}^{a_{t-1}^n})\big), \tag{4}$$

  for $n \neq k_t$, where $\mathbf{A}_t(\mathbf{x}) \coloneqq (\mathbf{C}_t(\mathbf{x}) + \frac{\delta_t}{2}\mathbf{I})^{-1}\mathbf{C}_t(\mathbf{x})$ and $\mathbf{B}_t(\mathbf{x}) \coloneqq \frac{\delta_t}{2}\mathbf{A}_t(\mathbf{x})^2 + \mathbf{A}_t(\mathbf{x})$. Sampling from this joint proposal can be achieved by first sampling an auxiliary variable $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \frac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{I})$ and then $\mathbf{x}_t^n \sim M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_t)$, for $n \neq k_t$, where $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ is the fully-adapted auxiliary particle-filter proposal for the state-space model with Gaussian transitions $\mathbf{x}_t|\mathbf{x}_{t-1} \sim M_t(\mathbf{x}_t|\mathbf{x}_{t-1})$ and pseudo observations $\mathbf{u}_t|\mathbf{x}_t \sim \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t; \frac{\delta_t}{2}\mathbf{I})$. We call this the *Particle-aGRAD* algorithm because the auxiliary variables $\mathbf{u}_t$ again appear in the particle weights, and because it generalises the powerful *aGRAD* algorithm from Titsias and Papaspiliopoulos (2018) to $T > 1$ time steps and $N > 1$ proposals.

- **Particle-mGRAD.** In Section 4.2, under the assumption that $\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t$ and in analogy to Section 3.2, we improve Particle-aGRAD by marginalising out the auxiliary variables $\mathbf{u}_t$. We call the resulting method *Particle-mGRAD* because it generalises the powerful *mGRAD* algorithm from Titsias and Papaspiliopoulos (2018) to $T > 1$ time steps and $N > 1$ proposals.

- **Particle-aGRAD+.** In Section 4.3, in analogy to Section 3.3, we improve Particle-aGRAD by replacing the 'filter-potential' gradients $\nabla_{\mathbf{x}_t} \log G_t(\mathbf{x}_{t-1:t})$ in (4) with 'smoothing-potential' gradients $\nabla_{\mathbf{x}_t} \log G_{1:T}(\mathbf{x}_{1:T})$ which may be beneficial if $G_t(\mathbf{x}_{t-1:t})$ varies significantly in $\mathbf{x}_{t-1}$. We call this method *Particle-aGRAD+*.

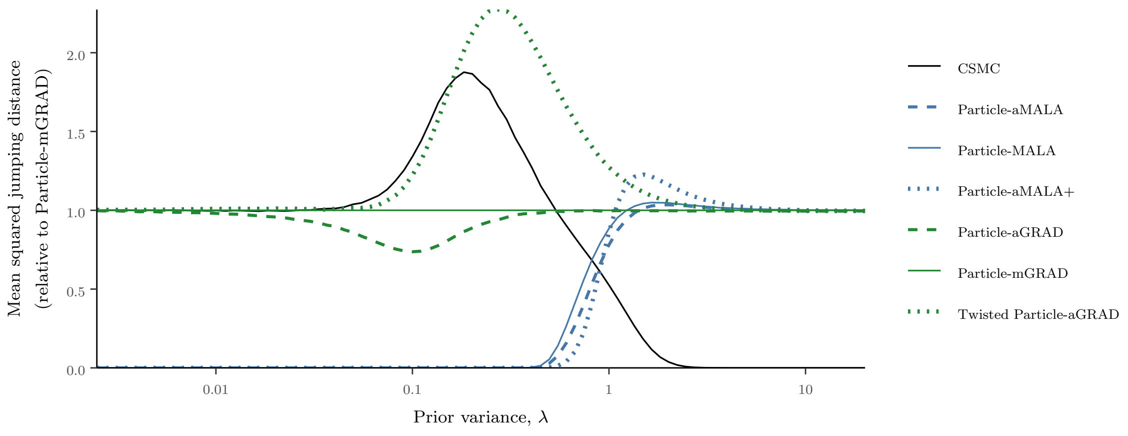


Figure 2: Empirical illustration of the 'interpolation' from Propositions 18 and 19 in the toy linear-Gaussian state-space model from Figure 1 (with $D = T = 10$).

- **Twisted Particle-aGRAD(+).** In Section 4.4, under the assumption that $\mathbf{m}_t(\mathbf{x}_{t-1}) = \mathbf{F}_t\mathbf{x}_{t-1}+\mathbf{b}_t$ and $\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t$, we improve Particle-aGRAD and Particle-aGRAD+ by instead using all future auxiliary variables $\mathbf{u}_{t:T}$ to propose $\mathbf{x}_t^n \sim M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_{t:T})$, for $n \neq k_t$, where $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{t:T}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t:T})$ is the fully *twisted* particle filter proposal for the state-space model with Gaussian transitions and pseudo observations $\mathbf{u}_t$ mentioned above. We call the resulting methods *twisted Particle-aGRAD* and *twisted Particle-aGRAD+*.

In Section 4.6, we prove that Particle-aGRAD and Particle-mGRAD (and their smoothing-gradient/twisted variants) solve the 'tuning' problem of having to choose between:

1. the CSMC algorithm (which proposes particles solely based on the prior dynamics);
2. the Particle-aMALA, Particle-MALA or Particle-aMALA+ (which propose particles solely locally around the reference path).

This choice is not always clear: on the one hand, Choice 2 can exhibit superior performance in high dimensions. On the other hand, if the prior dynamics are highly informative then Choice 1 can outperform Choice 2. Specifically, we prove that the following results hold in stationarity and under the simplifying assumption that the model factorises over time, i.e., if G_t, $\mathbf{m}_t$, $\mathbf{C}_t$ (and hence $\mathbf{A}_t$ and $\mathbf{B}_t$ in (4)) do not depend on the state at time $t - 1$:

- **Proposition 18.** Particle-aGRAD and Particle-mGRAD reduce to the CSMC algorithm as prior dynamics become *more* informative. Informally, we then have $\mathbf{A}_t \approx \mathbf{0}$ and $\mathbf{B}_t \approx \mathbf{C}_t$ so that (4) reduces to (2).
- **Proposition 19.** Particle-aGRAD and Particle-mGRAD reduce to Particle-aMALA and Particle-MALA, respectively, as prior dynamics become *less* informative. Informally, we then have $\mathbf{A}_t \approx \mathbf{I}$ and $\mathbf{B}_t \approx \delta_t\mathbf{I}$ so that (4) reduces to (3).

Propositions 18 and 19 are illustrated in Figure 2 (for a model which does not satisfy the above-mentioned 'factorisation-over-time' assumption). As a by-product, these propositions show that the aGRAD/mGRAD algorithms from Titsias and Papaspiliopoulos (2018) can be viewed as automatically interpolating between the IMH algorithm and aMALA/MALA, depending on the 'informativeness' of the prior. To our knowledge, this has not been pointed out in the literature. As another by-product, the methodology presented in this section also addresses the 'tuning problem' of having to choose whether to sample the initial latent state $\mathbf{x}_1$ within the CSMC scheme (which is preferable if the prior on the initial state is informative) or to treat it as a 'static' parameter to be sampled separately (which is preferable if this prior is diffuse, see Murray et al., 2013; Fearnhead and Meligkotsidou, 2016; Karppinen and Vihola, 2021).

In Section 5, we demonstrate the performance of our methodology on a high-dimensional multivariate stochastic volatility model, often used as a benchmark in the particle filtering literature. The different methods proposed in this article dramatically improve on existing CSMC and related methods and also on 'classical' MCMC methods in terms of effective sample size for different levels of prior informativeness.

All proofs (e.g., of the fact that the proposed methods leave $\pi_T(\mathbf{x}_{1:T})$ invariant) are deferred to the appendix. Additionally, in Appendix A, we introduce *Particle-PCNL* methods which generalise the *preconditioned Crank–Nicolson–Langevin (PCNL)* algorithm from Cotter et al. (2013) to $T > 1$ time steps and $N > 1$ proposals. The methods proposed in this work and their special cases if $N = T = 1$ are summarised in Table 1. Note that for $T = 1$ but $N > 1$, our work implies novel *multi-proposal* versions of 'classical' MCMC kernels like MALA, aMALA, mGRAD, aGRAD and PCNL. These may be of independent interest because they can exploit parallel computing architectures for inference in non-dynamic models.

Importantly, and in keeping with existing CSMC methodology, the computational cost of all our proposed algorithms is linear in both T and N, in time and memory.

Finally, the Python code for reproducing our experiments is publicly available at `https://github.com/AdrienCorenflos/particle_mala`. It was written as a library and can be extended to accommodate other models than the ones considered here.

2 Existing methodology

2.1 CSMC (particle extension of IMH)

2.1.1 Algorithm

Assume that we can generate independent and identically distributed (IID) samples from the mutation kernels $M_t(\mathbf{x}_t|\mathbf{x}_{t-1})$. A method for constructing a π_T-invariant MCMC kernel $P_{\mathrm{CSMC}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ is then given by the *CSMC* algorithm from Andrieu et al. (2010) which proposes N particles at each time step to build up an efficient proposal. Algorithm 1 summarises the scheme, where 'w.p.' is short for 'with probabilitity'. We also recursively define the nth surviving particle lineage at time t as

$$\mathbf{x}_{1:t}^{(n)} := (\mathbf{x}_{1:t-1}^{(a_{t-1}^n)}, \mathbf{x}_t^n).$$

In particular, therefore, $\mathbf{x}_{t-1:t}^{(n)} = (\mathbf{x}_{t-1}^{a_{t-1}^n}, \mathbf{x}_t^n)$.

Table 1: The methods mentioned in this work (new methods are in *italic*).

Method	**Section**	**Special case if $N = T = 1$**
CSMC[†]	2.1	IMH
Particle-RWM	2.2	RWM
Particle-aMALA	3.1	aMALA
Particle-MALA	3.2	MALA
Particle-aMALA+	3.3	aMALA
Particle-aGRAD	4.1	aGRAD
Particle-mGRAD	4.2	mGRAD
Particle-aGRAD+	4.3	aGRAD
Twisted Particle-aGRAD(+)	4.4	aGRAD
Particle-PCNL & more[‡]	Appendix A	PCNL

[†] In our taxonomy, CSMC could be called 'Particle-IMH'. However, the latter already refers to an entirely different algorithm in Andrieu et al. (2010).
[‡] We again also describe auxiliary-variable, smoothing-gradient ('+') and twisted versions.

Algorithm 1 (CSMC) *Given* $\mathbf{x}_{1:T} \in \mathcal{X}^T$*:*

```
1. for $t = 1, \dots, T$,
    (a) sample $k_t$ from a uniform distribution on $[N]_0$ and set $\mathbf{x}_t^{k_t} := \mathbf{x}_t$,
    (b) if $t > 1$, set $a_{t-1}^{k_t} := k_{t-1}$ and sample $a_{t-1}^n = i$ w.p. $W_{t-1}^i$, for $n \in [N]_0 \setminus \{k_t\}$,
    (c) sample $\mathbf{x}_t^n \sim M_t(\cdot \,|\mathbf{x}_{t-1}^{a_{t-1}^n})$ for $n \in [N]_0 \setminus \{k_t\}$,
    (d) for $n \in [N]_0$, set $w_t^n \propto G_t(\mathbf{x}_{t-1:t}^{(n)})$.
    (e) for $n \in [N]_0$, set $W_t^n := w_t^n / \sum_{m=0}^N w_t^m$;
2. sample $i \in [N]_0 \setminus \{k_T\}$ w.p. $\frac{W_T^i}{1 - W_T^{k_T}}$; set $l_T := i$ w.p. $1 \wedge \frac{1 - W_T^{k_T}}{1 - W_T^i}$; otherwise, set $l_T := k_T$;
3. for $t = T-1, \dots, 1$, sample $l_t = i \in [N]_0$ w.p. $\frac{W_t^i Q_{t+1}(\mathbf{x}_t^i, \mathbf{x}_{t+1}^{l_{t+1}})}{\sum_{n=0}^N W_t^n Q_{t+1}(\mathbf{x}_t^n, \mathbf{x}_{t+1}^{l_{t+1}})}$;
4. return $\tilde{\mathbf{x}}_{1:T} := (\mathbf{x}_1^{l_1}, \dots, \mathbf{x}_t^{l_T})$.
```

Algorithm 1 includes two extensions to the original presentation of the CSMC algorithm in Andrieu et al. (2010):

- Step 2 uses the so-called *forced-move* extension for CSMC algorithms which was proposed in Chopin and Singh (2013) (see also Liu, 1996). The algorithm would still be valid if we instead sampled $l_T = i \in [N]_0$ with probability W_T^i.
- Step 3 is the *backward-sampling* extension from Whiteley (2010). The algorithm would still be valid if we instead set $l_t = a_t^{l_{t+1}}$ (but typically much less efficient, especially if T is large).

Importantly, sampling $\tilde{\mathbf{x}}_{1:T}$ given $\mathbf{x}_{1:T}$ as described in Algorithm 1 induces a Markov kernel $P_{\mathrm{CSMC}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ which leaves π_T invariant. For sufficiently ergodic models, this MCMC kernel can yield highly efficient updates of the sequence of latent states, even if the time horizon T is large (Lee et al., 2020; Karjalainen et al., 2023).

2.1.2 Relationship with 'classical' MCMC algorithms

Interestingly, the CSMC algorithm generalises the classical *IMH* algorithm (Hastings, 1970) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where we suppress the 'time' subscript $t = 1$ everywhere to simplify the notation. Given that the current state of the Markov chain is $\mathbf{x} = \mathbf{x}^0$ (we can assume that $k = 0$ without loss of generality), Step 1c of Algorithm 1 proposes $\mathbf{x}^1 \sim M$. The remaining steps return $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\mathrm{IMH}}(\mathbf{x}^0, \mathbf{x}^1)$, where

$$\alpha_{\mathrm{IMH}}(\mathbf{x}^0, \mathbf{x}^1) \coloneqq \frac{1 - W^0}{1 - W^1} = \frac{G(\mathbf{x}^1)}{G(\mathbf{x}^0)} = \frac{\pi(\mathbf{x}^1)M(\mathbf{x}^0)}{\pi(\mathbf{x}^0)M(\mathbf{x}^1)}.$$

Otherwise, the old state $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^0 = \mathbf{x}$ is returned as the new state.

2.1.3 Breakdown in high dimensions

Unfortunately, as shown in Finke and Thiery (2023), Algorithm 1 suffers from a curse of dimension if D is large (unless the number of proposed particles, N, grows exponentially in D but that is prohibitive). This is not surprising since the IMH algorithm is known to break down in high dimensions (due to the difficulty of finding an efficient global proposal distribution M in high dimensions).

2.2 Particle-RWM

2.2.1 Algorithm

To circumvent the curse of dimension, Finke and Thiery (2023) (see also Shestopaloff and Neal, 2018; Malory, 2021, for related methods) developed the *particle random-walk Metropolis (Particle-RWM)* algorithm which scatters the proposed particles locally around the reference path using Gaussian perturbations as outlined in Algorithm 2.

Algorithm 2 (Particle-RWM) *Implement Algorithm 1 but replace the particle proposal (Step 1c) and the weight calculation (Step 1d) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t, \frac{\delta_t}{2}\mathbf{I})$, *and* $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{u}_t, \frac{\delta_t}{2}\mathbf{I})$, *for* $n \in [N]_0 \setminus \{k_t\}$,

1d. for $n \in [N]_0$, *set* $w_t^n \propto Q_t(\mathbf{x}_{t-1:t}^{(n)})$.

Notably, Step 1c marginally samples $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{x}_t, \delta_t \mathbf{I})$, for $n \neq k_t$.

2.2.2 Interpretation as a CSMC update on an extended space

Corenflos and Särkkä (2023) showed that Algorithm 2 can be derived by including the auxiliary variables $\mathbf{u}_t$ into the space and thus considering the extended distribution

$$\pi_T'(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \coloneqq \pi_T(\mathbf{x}_{1:T}) \prod_{t=1}^{T} \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \tfrac{\delta_t}{2}\mathbf{I}),$$

which admits $\pi_T(\mathbf{x}_{1:T})$ as a marginal and which can be targeted by alternating the following two steps. Given $\mathbf{x}_{1:T} \in \mathcal{X}^T$,

1. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t, \frac{\delta_t}{2}\mathbf{I})$, for $t = 1, \dots, T$;
2. run the CSMC algorithm (Algorithm 1) but with $M_t(\mathbf{x}_t|\mathbf{x}_{t-1})$, $G_t(\mathbf{x}_{t-1:t})$, and $Q_t(\mathbf{x}_{t-1:t})$ replaced by $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t) \coloneqq \mathrm{N}(\mathbf{x}_t; \mathbf{u}_t, \frac{\delta_t}{2}\mathbf{I})$, $G_t'(\mathbf{x}_{t-1:t}) \coloneqq Q_t(\mathbf{x}_{t-1:t})$ and $Q_t'(\mathbf{x}_{t-1:t}; \mathbf{u}_t) \coloneqq M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t) G_t'(\mathbf{x}_{t-1:t})$.

In particular, this shows that sampling $\tilde{\mathbf{x}}_{1:T}$ given $\mathbf{x}_{1:T}$ via Algorithm 2 induces a Markov kernel $P_{\text{Particle-RWM}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ which leaves π_T invariant.

2.2.3 Relationship with 'classical' MCMC algorithms

The Particle-RWM algorithm generalises the classical (Gaussian) *RWM* algorithm of Metropolis et al. (1953) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where we again suppress the 'time' subscript $t = 1$ everywhere to simplify the notation. Given that the current state of the Markov chain is $\mathbf{x} = \mathbf{x}^0$ (we can again assume that $k = 0$ without loss of generality), Step 1c of Algorithm 2 proposes $\mathbf{x}^1 \sim \mathrm{N}(\mathbf{x}^0, \delta \mathbf{I})$. The remaining steps return $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\text{RWM}}(\mathbf{x}^0, \mathbf{x}^1)$, where

$$\alpha_{\text{RWM}}(\mathbf{x}^0, \mathbf{x}^1) \coloneqq \frac{1 - W^0}{1 - W^1} = \frac{\pi(\mathbf{x}^1)}{\pi(\mathbf{x}^0)}.$$

Otherwise, the old state $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^0 = \mathbf{x}$ is returned as the new state.

2.2.4 Stability in high dimensions

Finke and Thiery (2023) proved that the Particle-RWM algorithm circumvents the curse of dimensionality if the proposal variance is scaled as $\delta_t \in \mathrm{O}(D^{-1})$ (see also Malory, 2021, for a proof for non-Gaussian exchangeable proposals but in the case where the model factorises over time). However, from the literature on classical MCMC algorithms, it is well known that faster convergence rates can be achieved by incorporating gradient information into the proposal (Roberts and Rosenthal, 1998). Thus, in the next section, we extend the Particle-RWM to allow for gradient-informed proposals.

3 Particle extensions of MALA and aMALA

3.1 Particle-aMALA

We now propose *Particle-aMALA*, a method which extends the Particle-RWM algorithm from Finke and Thiery (2023) by allowing for the use of gradient information in the proposal. For the moment, gradients are taken w.r.t. the filtering densities and we employ an indicator $\kappa \in \{0, 1\}$ to permit switching off the use of gradient information.

We now write

$$M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t) \coloneqq \mathrm{N}(\mathbf{x}_t; \mathbf{u}_t, \tfrac{\delta_t}{2}\mathbf{I}), \tag{5}$$

$$G'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t) \coloneqq Q_t(\mathbf{x}_{t-1:t}) \frac{\mathrm{N}(\mathbf{u}_t; \mathbf{x}_t + \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t} \log \pi_t(\mathbf{x}_{1:t}), \frac{\delta_t}{2}\mathbf{I})}{\mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \frac{\delta_t}{2}\mathbf{I})}, \tag{6}$$

as well as $Q'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t) \coloneqq M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t) G'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t)$, where we note that

$$\nabla_{\mathbf{x}_t} \log \pi_t(\mathbf{x}_{1:t}) = \nabla_{\mathbf{x}_t} \log Q_t(\mathbf{x}_{t-1:t}).$$

A single iteration of the Particle-aMALA is then as follows.

Algorithm 3 (Particle-aMALA) *Implement Algorithm 1 but replace the particle proposal (Step 1c) and the weight calculation (Step 1d) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t} \log \pi_t(\mathbf{x}_{1:t}), \frac{\delta_t}{2}\mathbf{I})$*, and* $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{u}_t, \frac{\delta_t}{2}\mathbf{I})$*, for* $n \in [N]_0 \setminus \{k_t\}$*,*

1d. for $n \in [N]_0$*, set* $w_t^n \propto G'_t(\mathbf{x}_{t-1:t}^{(n)}; \mathbf{u}_t)$*,*

and also replace $Q_{t+1}(\,\cdot\,)$ *in the backward kernel in Step 3 by* $Q'_{t+1}(\,\cdot\,; \mathbf{u}_{t+1})$*.*

Step 1c marginally samples $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t} \log \pi_t(\mathbf{x}_{1:t}), \delta_t\mathbf{I})$, for $n \neq k_t$. This follows from Lemma 31 in Appendix C.

Proposition 4 (validity of Particle-aMALA) *Using Algorithm 3 to sample* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *induces a Markov kernel* $P_{\text{Particle-aMALA}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

3.2 Particle-MALA

In this section, we analytically integrate out the auxiliary variables $\mathbf{u}_t$ appearing in the weights of the Particle-aMALA. A single iteration of the resulting methodology – which we term the *Particle-MALA* – is as follows, where we write

$$\log H_{t,\boldsymbol{\phi}}(\mathbf{x}, \bar{\mathbf{x}}) \coloneqq \frac{1}{\delta_t}\big[2\boldsymbol{\phi}^{\mathrm{T}}(\bar{\mathbf{x}} - \mathbf{x}) - \tfrac{N}{N+1}\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi}\big]. \tag{7}$$

Algorithm 5 (Particle-MALA) *Implement Algorithm 1 but replace the particle proposal (Step 1c) and the weight calculation (Step 1d) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log\pi_t(\mathbf{x}_{1:t}), \frac{\delta_t}{2}\mathbf{I})$*, and* $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{u}_t, \frac{\delta_t}{2}\mathbf{I})$*, for* $n \in [N]_0 \setminus \{k_t\}$*,*

1d. set $\bar{\mathbf{x}}_t := \frac{1}{N+1}\sum_{n=0}^N \mathbf{x}_t^n$ *and, for* $n \in [N]_0$*,*

$$w_t^n \propto Q_t(\mathbf{x}_{t-1:t}^{(n)}) H_{t,\kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t^n}\log Q_t(\mathbf{x}_{t-1:t}^{(n)})}(\mathbf{x}_t^n, \bar{\mathbf{x}}_t).$$

Step 1d pre-computes $\bar{\mathbf{x}}_t$ to ensure that the algorithm can still be implemented in $\mathrm{O}(N)$ operations even though the weight of the nth particle now depends on the values of all $N+1$ particles. However, note that the auxiliary variables $\mathbf{u}_t$ no longer appear in the weights.

Remark 6 (Particle-aMALA 'exactly approximates' Particle-MALA) *Note that the Particle-aMALA differs from the Particle-MALA only in the definition of the weights (and the backward-sampling weights). This allows us to interpret the former as a 'noisy' version of the latter. Indeed, write the unnormalised weight of the nth particle at time-t in the Particle-aMALA as* $w_t^n(\mathbf{u}_t)$*, whilst* w_t^n *denotes the corresponding weight under the Particle-MALA (which does not depend on the auxiliary variable* $\mathbf{u}_t$*). Then we have*

$$\frac{w_t^n(\mathbf{u}_t)}{w_t^{k_t}(\mathbf{u}_t)} = \frac{w_t^n}{w_t^{k_t}} \times \frac{q_t^{-n}(\mathbf{u}_t|\mathbf{x}_t^{-n}, \mathbf{x}_t^n; \mathcal{H}_{t-1})}{q_t^{-k_t}(\mathbf{u}_t|\mathbf{x}_t^{-k_t}, \mathbf{x}_t^{k_t}; \mathcal{H}_{t-1})},$$

where $q_t^{-n}(\mathbf{u}_t|\mathbf{x}_t^{-n}, \mathbf{x}_t^n; \mathcal{H}_{t-1}) = \mathrm{N}(\mathbf{u}_t; \bar{\mathbf{x}}_t + \kappa\frac{\delta_t}{2(N+1)}\nabla_{\mathbf{x}_t^n}\log Q_t(\mathbf{x}_{t-1:t}^{(n)}), \frac{\delta_t}{2(N+1)}\mathbf{I})$ *is the conditional distribution of* $\mathbf{u}_t$ *under the joint distribution of all random variables generated by Algorithm 3 up to (and including) time* t *assuming the reference particle at time* t *is placed in position* n *(and* $\mathcal{H}_{t-1}$ *denotes the history of the particle system, i.e. all particles and ancestor indices up to time* $t-1$*). This conditional distribution follows from Lemma 31 in Appendix C. In particular, we therefore have*

$$\mathbb{E}\left[\frac{w_t^n(\mathbf{u}_t)}{w_t^{k_t}(\mathbf{u}_t)}\right] = \frac{w_t^n}{w_t^{k_t}},$$

where the expectation is taken w.r.t. $q_t^{-k_t}(\mathbf{u}_t|\mathbf{x}_t^{-k_t}, \mathbf{x}_t^{k_t}; \mathcal{H}_{t-1})$*. Interestingly, for the Particle-RWM algorithm (recovered by setting* $\kappa = 0$*), the 'auxiliary' and 'marginal' variants are statistically equivalent.*

Proposition 7 (validity of Particle-MALA) *Using Algorithm 5 to sample* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *induces a Markov kernel* $P_{\text{Particle-MALA}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

3.3 Particle-aMALA+

In this section, we extend the Particle-aMALA in a different manner: we now modify the algorithm so that the proposal distributions incorporate gradients w.r.t. the joint smoothing distribution π_T rather than w.r.t. the filters, π_t. This can be beneficial if there is a significant discrepancy between the marginal distribution of $\mathbf{x}_t$ under the former and the latter as is typically the case if D is large. Indeed, this discrepancy is likely the reason for the decay in performance of Particle-aMALA and Particle-MALA for very large D visible in Figure 1a.

For $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_t)$ and $G'_t(\mathbf{x}_{t-1:t};\mathbf{u}_t)$ still defined as in the Particle-aMALA algorithm (i.e., as in (5) and (6)), we now write

$$G'_t(\mathbf{x}_{t-2:t};\mathbf{u}_{t-1:t}) \coloneqq G'_t(\mathbf{x}_{t-1:t};\mathbf{u}_t)\frac{\mathrm{N}(\mathbf{u}_{t-1};\mathbf{x}_{t-1}+\kappa\frac{\delta_{t-1}}{2}\nabla_{\mathbf{x}_{t-1}}\log\pi_T(\mathbf{x}_{1:T}),\frac{\delta_{t-1}}{2}\mathbf{I})}{\mathrm{N}(\mathbf{u}_{t-1};\mathbf{x}_{t-1}+\kappa\frac{\delta_{t-1}}{2}\nabla_{\mathbf{x}_{t-1}}\log\pi_{t-1}(\mathbf{x}_{1:t-1}),\frac{\delta_{t-1}}{2}\mathbf{I})},$$

as well as $Q'_t(\mathbf{x}_{t-2:t};\mathbf{u}_{t-1:t}) \coloneqq M'_t(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_t)G'_t(\mathbf{x}_{t-2:t};\mathbf{u}_{t-1:t})$, where we note that

$$\nabla_{\mathbf{x}_t}\log\pi_T(\mathbf{x}_{1:T}) = \nabla_{\mathbf{x}_t}[\log Q_t(\mathbf{x}_{t-1:t})+\log Q_{t+1}(\mathbf{x}_{t:t+1})].$$

A single iteration of the resulting 'smoothing-gradient' methodology – which we term the *Particle-aMALA+* – is then as follows.

Algorithm 8 (Particle-aMALA+) *Implement Algorithm 1 but replace the particle proposal (Step 1c), the weight calculation (Step 1d), and backward sampling (Step 3) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t+\kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log\pi_T(\mathbf{x}_{1:T}),\frac{\delta_t}{2}\mathbf{I})$*, and* $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{u}_t,\frac{\delta_t}{2}\mathbf{I})$*, for* $n\in[N]_0\backslash\{k_t\}$*,*

1d. for $n\in[N]_0$*, set* $w_t^n \propto G'_t(\mathbf{x}_{t-2:t}^{(n)};\mathbf{u}_{t-1:t})$*,*

3. for $t=T-1,\dots,1$*, sample* $l_t=i\in[N]_0$ *w.p.*

$$\frac{W_t^i Q'_{t+1}((\mathbf{x}_{t-1:t}^{(i)},\mathbf{x}_{t+1}^{l_{t+1}});\mathbf{u}_{t:t+1})Q'_{t+2}((\mathbf{x}_t^i,\mathbf{x}_{t+1}^{l_{t+1}},\mathbf{x}_{t+2}^{l_{t+2}});\mathbf{u}_{t+1:t+2})}{\sum_{n=0}^N W_t^n Q'_{t+1}((\mathbf{x}_{t-1:t}^{(n)},\mathbf{x}_{t+1}^{l_{t+1}});\mathbf{u}_{t:t+1})Q'_{t+2}((\mathbf{x}_t^n,\mathbf{x}_{t+1}^{l_{t+1}},\mathbf{x}_{t+2}^{l_{t+2}});\mathbf{u}_{t+1:t+2})}.$$

In Step 3, we recall the convention that any quantity with 'time' index $t>T$ should be ignored, so that $Q'_{T+1}\equiv 1$. Some comments about Algorithm 8 are in order.

- Step 1c marginally samples $\mathbf{x}_t^n \sim \mathrm{N}(\mathbf{x}_t+\kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log\pi_T(\mathbf{x}_{1:T}),\delta_t\mathbf{I})$, for $n\neq k_t$. This is in contrast to the Particle-aMALA and Particle-MALA, whose (marginal) proposal distribution is centred around $\mathbf{x}_t+\kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log\pi_t(\mathbf{x}_{1:t})$.
- Steps 1d and 3 are similar to the weight-calculation and backward-sampling steps in the previous algorithms. The only difference here is that the model is now no longer (first-order) Markov in the sense that the (incremental) weights at time t now also depend on the state at time $t-2$.

Proposition 9 (validity of Particle-aMALA+) *Using Algorithm 8 to sample* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *induces a Markov kernel* $P_{\text{Particle-MALA}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

3.4 Relationship with other methods

We end this section by relating the proposed algorithms to existing methodologies.

1. **Generalisation of Particle-RWM and RWM.** If $\kappa=0$, then the algorithms introduced in this section (Particle-aMALA, Particle-MALA and Particle-aMALA+) do not make use of any gradient information and reduce to the Particle-RWM algorithm. In particular, if $T=N=1$, they thus reduce to the RWM algorithm.

2. **Generalisation of aMALA.** For $\kappa = 1$, the Particle-aMALA (and similarly the Particle-aMALA+) algorithm generalise the *auxiliary MALA (aMALA)* from Titsias and Papaspiliopoulos (2018) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where we again suppress the 'time' subscript $t = 1$ everywhere. Given that the current state of the Markov chain is $\mathbf{x} = \mathbf{x}^0$ (we can assume that $k = 0$ without loss of generality), Step 1c of Algorithm 3 first refreshes the auxiliary variable by sampling $\mathbf{u} \sim \mathrm{N}(\mathbf{x}^0 + \frac{\delta}{2}\nabla \log \pi(\mathbf{x}^0), \frac{\delta}{2}\mathbf{I})$ and then proposes $\mathbf{x}^1 \sim \mathrm{N}(\mathbf{u}, \frac{\delta}{2}\mathbf{I})$. The remaining steps return $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\mathrm{aMALA}}(\mathbf{x}^0, \mathbf{x}^1; \mathbf{u})$, where

$$\alpha_{\mathrm{aMALA}}(\mathbf{x}^0, \mathbf{x}^1; \mathbf{u}) \coloneqq \frac{1 - W^0}{1 - W^1} = \frac{\pi(\mathbf{x}^1)\,\mathrm{N}(\mathbf{u}; \mathbf{x}^1 + \frac{\delta}{2}\nabla \log \pi(\mathbf{x}^1), \frac{\delta}{2}\mathbf{I})\,\mathrm{N}(\mathbf{x}^0; \mathbf{u}, \frac{\delta}{2}\mathbf{I})}{\pi(\mathbf{x}^0)\,\mathrm{N}(\mathbf{u}; \mathbf{x}^0 + \frac{\delta}{2}\nabla \log \pi(\mathbf{x}^0), \frac{\delta}{2}\mathbf{I})\,\mathrm{N}(\mathbf{x}^1; \mathbf{u}, \frac{\delta}{2}\mathbf{I})}.$$

   Otherwise, the old state $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^0 = \mathbf{x}$ is returned as the new state. This induces the same Markov chain on $\mathcal{X}$ as the aMALA from Titsias and Papaspiliopoulos (2018) (the only difference relates to a re-centring of the auxiliary variables $\mathbf{u}$ previously discussed in Corenflos and Särkkä (2023) but this does not change the law of the Markov chain on the marginal space which does not include the auxiliary variable).

3. **Generalisation of MALA.** Still taking $\kappa = 1$, the Particle-MALA generalises the *Metropolis-adjusted Langevin algorithm (MALA)* (Besag, 1994) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where use the same notational conventions as in the case of aMALA above. Step 1c of Algorithm 5 then marginally proposes $\mathbf{x}^1 \sim \mathrm{N}(\mathbf{x}^0 + \frac{\delta}{2}\nabla \log \pi(\mathbf{x}^0), \delta\mathbf{I})$. The remaining steps return $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\mathrm{MALA}}(\mathbf{x}^0, \mathbf{x}^1)$, where

$$\alpha_{\mathrm{MALA}}(\mathbf{x}^0, \mathbf{x}^1) \coloneqq \frac{1 - W^0}{1 - W^1} = \frac{\pi(\mathbf{x}^1)\,\mathrm{N}(\mathbf{x}^0; \mathbf{x}^1 + \frac{\delta}{2}\nabla \log \pi(\mathbf{x}^1), \delta\mathbf{I})}{\pi(\mathbf{x}^0)\,\mathrm{N}(\mathbf{x}^1; \mathbf{x}^0 + \frac{\delta}{2}\nabla \log \pi(\mathbf{x}^0), \delta\mathbf{I})}.$$

   Otherwise, the old state $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^0 = \mathbf{x}$ is returned as the new state.

   In particular, Remark 6 shows that we can view the aMALA as a 'noisy' version of MALA (as already mentioned in Titsias and Papaspiliopoulos, 2018) because, dropping the time subscript again, by Lemma 31:

$$\alpha_{\mathrm{aMALA}}(\mathbf{x}^0, \mathbf{x}^1; \mathbf{u}) = \alpha_{\mathrm{MALA}}(\mathbf{x}^0, \mathbf{x}^1)\frac{\mathrm{N}(\mathbf{u}; \bar{\mathbf{x}} + \frac{\delta}{4}\nabla \log \pi(\mathbf{x}^1), \frac{\delta}{4}\mathbf{I})}{\mathrm{N}(\mathbf{u}; \bar{\mathbf{x}} + \frac{\delta}{4}\nabla \log \pi(\mathbf{x}^0), \frac{\delta}{4}\mathbf{I})},$$

   where $\bar{\mathbf{x}} = (\mathbf{x}^0 + \mathbf{x}^1)/2$, and hence

$$\mathbb{E}[\alpha_{\mathrm{aMALA}}(\mathbf{x}^0, \mathbf{x}^1; \mathbf{u})] = \alpha_{\mathrm{MALA}}(\mathbf{x}^0, \mathbf{x}^1),$$

   where the expectation is w.r.t. the conditional distribution of $\mathbf{u}$ under the joint distribution of the random variables sampled in Step 1c of the Particle-aMALA, i.e. w.r.t. $\mathrm{N}(\bar{\mathbf{x}} + \frac{\delta}{4}\nabla \log \pi(\mathbf{x}^0), \frac{\delta}{4}\mathbf{I})$. In other words, this algorithm is the same as MALA except that the acceptance ratio is 'randomised' in the sense that it is multiplied by a non-negative random variable whose expectation is 1. Other examples of such algorithms can be found in Ceperley and Dewing (1999); Nicholls et al. (2012); see also Finke (2015, Section 3.3.3) for a discussion as well as Andrieu and Vihola (2016, page 2669) for a simple argument showing that the asymptotic variance of aMALA cannot be smaller than that of MALA.

4 Particle extensions of mGRAD and aGRAD

4.1 Particle-aGRAD

The gradient-informed algorithms (Particle-MALA, etc) developed in Section 3 can be expected to improve upon the Particle-RWM algorithm in the same way that aMALA/MALA improve upon the RWM algorithm. However, they may underperform compared to the CSMC algorithm when the prior dynamics of the latent states are highly informative in the same way that MALA can underperform relative to the IMH algorithm (with prior as proposal) if the prior is highly informative. Additionally, note that the algorithms from Section 3 employ proposals that are *separable* in the sense that, given the reference path, the marginal proposal distribution of $\mathbf{x}_t^n$ does not depend on the ancestor particle $\mathbf{x}_{t-1}^{a_{t-1}^n}$ (that is, separability implies that the weight-calculation and resampling steps could be postponed until *after* all particles have been proposed); such separable proposals can be expected to perform poorly if the latent states are highly correlated across time.

In this section, we further incorporate (conditionally) Gaussian prior dynamics into the particle proposals and thus interpolate between the CSMC algorithm and the gradient-informed algorithms of Section 3. Our construction generalises the aGRAD and mGRAD algorithms of Titsias and Papaspiliopoulos (2018). In particular, the algorithms introduced in this section do not imply separable proposals, i.e., the proposal kernel for particle $\mathbf{x}_t^n$ will generally depend on its ancestor particle $\mathbf{x}_{t-1}^{a_{t-1}^n}$.

Specifically, in this section, we consider the special case of the generic Feynman–Kac model from (1) in which we can find a decomposition $Q_t(\mathbf{x}_{t-1:t}) = M_t(\mathbf{x}_t|\mathbf{x}_{t-1})G_t(\mathbf{x}_{t-1:t})$, such that

$$M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1})), \tag{8}$$

is a Gaussian transition density whose mean $\mathbf{m}_t(\mathbf{x}_{t-1})$ and non-singular covariance matrix $\mathbf{C}_t(\mathbf{x}_{t-1})$ may depend on the previous state $\mathbf{x}_{t-1}$, for $t > 1$; and that $M_1(\mathbf{x}_1) = \mathrm{N}(\mathbf{x}_1; \mathbf{m}_1, \mathbf{C}_1)$.

Example 2 (state-space model, continued) *The methods proposed in this section immediately apply with $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq f_t(\mathbf{x}_t|\mathbf{x}_{t-1})$ if the state-space model has conditionally Gaussian dynamics, i.e. if $f_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$, by taking $G_t(\mathbf{x}_{t-1:t}) = g_t(\mathbf{y}_t|\mathbf{x}_t)$. However, they may often still apply to state-space models with non-Gaussian dynamics via a change of measure, i.e., by taking $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$ and $G_t(\mathbf{x}_{t-1:t}) = f_t(\mathbf{x}_t|\mathbf{x}_{t-1})g_t(\mathbf{y}_t|\mathbf{x}_t)/\,\mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$, or through a suitable transformation.*

The first method proposed in this section is termed *Particle-aGRAD*. Conditional on the auxiliary variables $\mathbf{u}_{1:T}$, it can be viewed as a CSMC algorithm whose proposal kernels are those of the fully-adapted auxiliary particle filter for the state-space model defined by the Gaussian transitions $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$ from (8) and 'pseudo observations' $\mathbf{u}_t$ with $p(\mathbf{u}_t|\mathbf{x}_t) = \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \frac{\delta_t}{2}\mathbf{I})$. We now write

$$\begin{aligned} M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t) &:= p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \\ &\propto \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))\, \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \tfrac{\delta_t}{2}\mathbf{I}) \\ &\propto \mathrm{N}(\mathbf{x}_t; \mathbf{m}'_t(\mathbf{x}_{t-1}, \mathbf{u}_t), \mathbf{C}'_t(\mathbf{x}_{t-1})), \end{aligned} \tag{9}$$

with

$$\mathbf{m}'_t(\mathbf{x}, \mathbf{u}) := \mathbf{m}_t(\mathbf{x}) + \mathbf{A}_t(\mathbf{x})[\mathbf{u} - \mathbf{m}_t(\mathbf{x})], \tag{10}$$

$$\mathbf{C}'_t(\mathbf{x}) := (\mathbf{I} - \mathbf{A}_t(\mathbf{x}))\mathbf{C}_t(\mathbf{x}) = \tfrac{\delta_t}{2}\mathbf{A}_t(\mathbf{x}), \tag{11}$$

$$\mathbf{A}_t(\mathbf{x}) := (\mathbf{C}_t(\mathbf{x}) + \tfrac{\delta_t}{2}\mathbf{I})^{-1}\mathbf{C}_t(\mathbf{x}),$$

as well as

$$G'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t) := Q_t(\mathbf{x}_{t-1:t})\frac{\mathrm{N}(\mathbf{u}_t; \mathbf{x}_t + \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{I})}{M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)}, \tag{12}$$

and $Q'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t) := M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)G'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t)$. A single iteration of the Particle-aGRAD algorithm is as follows.

Algorithm 10 (Particle-aGRAD) *Implement Algorithm 1 but replace the particle proposal (Step 1c) and the weight calculation (Step 1d) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{I})$, *and* $\mathbf{x}_t^n \sim M'_t(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_t)$, *for* $n \in [N]_0 \setminus \{k_t\}$,

1d. for $n \in [N]_0$, *set* $w_t^n \propto G'_t(\mathbf{x}_{t-1:t}^{(n)}; \mathbf{u}_t)$,

and also replace $Q_{t+1}(\,\cdot\,)$ *in the backward kernel in Step 3 by* $Q'_{t+1}(\,\cdot\,; \mathbf{u}_t)$.

Proposition 11 (validity of Particle-aGRAD) *Using Algorithm 10 to sample* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *induces a Markov kernel* $P_{\text{Particle-aGRAD}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

4.2 Particle-mGRAD

In this section, we analytically integrate out the auxiliary variables $\mathbf{u}_t$ which appeared in the weights of the Particle-aGRAD algorithm. Here we consider the case when the covariance matrices appearing in the conditionally Gaussian mutation kernel (8) do not depend on the previous state, i.e.,

$$\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t, \tag{13}$$

which then also implies that $\mathbf{A}_t(\mathbf{x}_{t-1}) = \mathbf{A}_t$. A single iteration of the resulting methodology – which we term the *Particle-mGRAD* algorithm – is as follows, where we write

$$\begin{aligned} \log H_{t,\phi}(\mathbf{x}, \mathbf{v}, \bar{\mathbf{x}}, \bar{\mathbf{v}}) = {} & \tfrac{1}{2}(\mathbf{x} - \mathbf{v})^{\mathrm{T}}((\tfrac{\delta_t}{2}\mathbf{A}_t)^{-1} + \mathbf{G}_t)(\mathbf{x} - \mathbf{v}) \\ & - [\tfrac{1}{2}N(\mathbf{x} + \boldsymbol{\phi})^{\mathrm{T}}\mathbf{A}_t + (\mathbf{x} - \mathbf{v})^{\mathrm{T}}]\mathbf{G}_t(\mathbf{x} + \boldsymbol{\phi}) \\ & + (N + 1)(\bar{\mathbf{x}} - \bar{\mathbf{v}})^{\mathrm{T}}\mathbf{G}_t(\mathbf{v} + \boldsymbol{\phi}), \end{aligned}$$

for $\mathbf{G}_t := \frac{2}{\delta_t}(\mathbf{I} + N\mathbf{A}_t)^{-1}$.

Algorithm 12 (Particle-mGRAD) *Implement Algorithm 1 but replace the particle proposal (Step 1c) and the weight calculation (Step 1d) by*

1c. *sample* $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{I})$ *and* $\mathbf{x}_t^n \sim M_t'(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_t)$, *for* $n \in [N]_0 \setminus \{k_t\}$,

1d. *set* $\bar{\mathbf{x}}_t := \frac{1}{N+1}\sum_{n=0}^N \mathbf{x}_t^n$, $\mathbf{v}_t^n := (\mathbf{I} - \mathbf{A}_t)\mathbf{m}_t(\mathbf{x}_{t-1}^{a_{t-1}^n})$, $\bar{\mathbf{v}}_t := \frac{1}{N+1}\sum_{n=0}^N \mathbf{v}_t^n$, *and, for* $n \in [N]_0$,

$$w_t^n \propto Q_t(\mathbf{x}_{t-1:t}^{(n)}) H_{t,\kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t^n}\log G_t(\mathbf{x}_{t-1:t}^{(n)})}(\mathbf{x}_t^n, \mathbf{v}_t^n, \bar{\mathbf{x}}_t, \bar{\mathbf{v}}_t). \tag{14}$$

Remark 13 (Particle-aGRAD 'exactly approximates' Particle-mGRAD) *In analogue to the relationship between Particle-aMALA and Particle-MALA discussed in Remark 6, Particle-aGRAD is a noisy version of Particle-mGRAD. That is, letting* $w_t^n(\mathbf{u}_t)$ *and* w_t^n *be the unnormalised weights under Particle-aGRAD and Particle-mGRAD, respectively, we have*

$$\mathbb{E}\left[\frac{w_t^n(\mathbf{u}_t)}{w_t^{k_t}(\mathbf{u}_t)}\right] = \frac{w_t^n}{w_t^{k_t}},$$

where the expectation is taken with respect to the conditional distribution of $\mathbf{u}_t$ *under the joint distribution of all random variables generated by Algorithm 10 up to (and including) time* t.

Proposition 14 (validity of Particle-mGRAD) *Using Algorithm 12 to sample* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *induces a Markov kernel* $P_{\text{Particle-aGRAD}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

4.3 Particle-aGRAD+

While the algorithm of Section 10 incorporates information from the smoothing distribution by merit of not modifying the latent dynamics, it may happen that the potential $G_t(\mathbf{x}_{t-1:t})$ strongly depends on $\mathbf{x}_{t-1}$. In this case, considering the 'myopic' gradient information $\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t})$ may not suffice to improve the mixing of the algorithm and information from $\mathbf{x}_{t-1}$ may then be beneficial. Similarly to Section 3.3, in this section, we extend the Particle-aGRAD algorithm to incorporate gradients w.r.t. the 'smoothing potential' $G_{1:T}(\mathbf{x}_{1:T}) = \prod_{t=1}^T G_t(\mathbf{x}_{t-1:t})$ rather than w.r.t. the 'filtering potential' $\prod_{s=1}^t G_s(\mathbf{x}_{s-1:s})$.

For $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)$ and $G_t'(\mathbf{x}_{t-1:t}; \mathbf{u}_t)$ still defined as in the Particle-aGRAD algorithm (i.e., as in (9) and (12)), we now write

$$G_t'(\mathbf{x}_{t-2:t}; \mathbf{u}_{t-1:t}) := G_t'(\mathbf{x}_{t-1:t}; \mathbf{u}_t)\frac{\mathrm{N}(\mathbf{u}_{t-1}; \mathbf{x}_{t-1} + \kappa\frac{\delta_{t-1}}{2}\nabla_{\mathbf{x}_{t-1}}\log G_{1:T}(\mathbf{x}_{1:T}), \frac{\delta_{t-1}}{2}\mathbf{I})}{\mathrm{N}(\mathbf{u}_{t-1}; \mathbf{x}_{t-1} + \kappa\frac{\delta_{t-1}}{2}\nabla_{\mathbf{x}_{t-1}}\log G_{t-1}(\mathbf{x}_{t-2:t-1}), \frac{\delta_{t-1}}{2}\mathbf{I})},$$

as well as $Q_t'(\mathbf{x}_{t-2:t}; \mathbf{u}_{t-1:t}) := M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)G_t'(\mathbf{x}_{t-2:t}; \mathbf{u}_{t-1:t})$, where we note that

$$\nabla_{\mathbf{x}_t}\log G_{1:T}(\mathbf{x}_{1:T}) = \nabla_{\mathbf{x}_t}\log[G_t(\mathbf{x}_{t-1:t}) + \log G_{t+1}(\mathbf{x}_{t:t+1})].$$

A single iteration of the resulting 'smoothing-gradient' methodology – which we term the *Particle-aGRAD+* algorithm – is as follows.

Algorithm 15 (Particle-aGRAD+) *Implement Algorithm 1 but replace the particle proposal (Step 1c), the weight calculation (Step 1d), and backward sampling (Step 3) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa \frac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log G_{1:T}(\mathbf{x}_{1:T}), \frac{\delta_t}{2}\mathbf{I})$, *and* $\mathbf{x}_t^n \sim M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)$, *for* $n \in [N]_0 \setminus \{k_t\}$,

1d. for $n \in [N]_0$, *set* $w_t^n \propto G_t'(\mathbf{x}_{t-2:t}^{(n)}; \mathbf{u}_{t-1:t})$,

3. for $t = T-1, \dots, 1$, *sample* $l_t = i \in [N]_0$ *w.p.*

$$\frac{W_t^i Q_{t+1}'((\mathbf{x}_{t-1:t}^{(i)}, \mathbf{x}_{t+1}^{l_{t+1}}); \mathbf{u}_{t:t+1}) Q_{t+2}'((\mathbf{x}_t^i, \mathbf{x}_{t+1}^{l_{t+1}}, \mathbf{x}_{t+2}^{l_{t+2}}); \mathbf{u}_{t+1:t+2})}{\sum_{n=0}^N W_t^n Q_{t+1}'((\mathbf{x}_{t-1:t}^{(n)}, \mathbf{x}_{t+1}^{l_{t+1}}); \mathbf{u}_{t:t+1}) Q_{t+2}'((\mathbf{x}_t^n, \mathbf{x}_{t+1}^{l_{t+1}}, \mathbf{x}_{t+2}^{l_{t+2}}); \mathbf{u}_{t+1:t+2})}.$$

Note that if $G_t(\mathbf{x}_{t-1:t}) = G_t(\mathbf{x}_t)$ does not depend on $\mathbf{x}_{t-1}$, then the Particle-aGRAD+ algorithm coincides with the Particle-aGRAD algorithm. However, when $G_t(\mathbf{x}_{t-1:t})$ varies highly in $\mathbf{x}_{t-1}$, their behaviours may differ substantially.

Proposition 16 (validity of Particle-aGRAD+) *Sampling* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *via Algorithm 15 induces a Markov kernel* $P_{\text{Particle-aGRAD+}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

4.4 Twisted Particle-aGRAD(+)

Recall that, conditionally on the auxiliary variables $\mathbf{u}_{1:T}$, the Particle-aGRAD algorithm could be viewed as a CSMC algorithm whose proposal kernels $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ are those of the fully-adapted auxiliary particle filter for the state-space model which is defined by the Gaussian transitions $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$ from (8) and observation densities $p(\mathbf{u}_t|\mathbf{x}_t) = \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \frac{\delta_t}{2}\mathbf{I})$.

In this section (and in this section only), we make the more restrictive assumption that the transition kernel from (8) is not only Gaussian but also affine, i.e.,

$$\mathbf{m}_t(\mathbf{x}_{t-1}) = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{b}_t, \quad \text{and} \quad \mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t, \tag{15}$$

for some $\mathbf{F}_t \in \mathbb{R}^{D \times D}$, $\mathbf{b}_t \in \mathbb{R}^D$, and some covariance matrix $\mathbf{C}_t \in \mathbb{R}^{D \times D}$. Under (15), we can then go one step further and implement the fully *twisted* particle filter (Whiteley and Lee, 2014; Guarniero et al., 2017; Heng et al., 2020) proposal which conditions on *all* future pseudo observations $\mathbf{u}_{t:T}$. That is, we now write

$$\begin{aligned} M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{t:T}) &\coloneqq p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t:T}) \\ &\propto \int_{\mathcal{X}^{T-t}} \left[\prod_{s=t}^T \mathrm{N}(\mathbf{x}_s; \mathbf{F}_s \mathbf{x}_{s-1} + \mathbf{b}_s, \mathbf{C}_s)\, \mathrm{N}(\mathbf{u}_s; \mathbf{x}_s, \tfrac{\delta_s}{2}\mathbf{I})\right] \mathrm{d}\mathbf{x}_{t+1:T} \\ &\propto \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t' \mathbf{x}_{t-1} + \mathbf{b}_t', \mathbf{C}_t'), \\ G_t'(\mathbf{x}_{t-1:t}; \mathbf{u}_{t:T}) &\coloneqq Q_t(\mathbf{x}_{t-1:t}) \frac{\mathrm{N}(\mathbf{u}_t; \mathbf{x}_t + \kappa \frac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{I})}{M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{t:T})}, \end{aligned} \tag{16}$$

as well as $Q_t'(\mathbf{x}_{t-1:t}; \mathbf{u}_{t:T}) \coloneqq M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{t:T}) G_t'(\mathbf{x}_{t-1:t}; \mathbf{u}_{t:T})$. Here, $\mathbf{b}_t' \in \mathbb{R}^D$ and $\mathbf{F}_t', \mathbf{C}_t' \in \mathbb{R}^{D \times D}$ can be obtained via Kalman-filtering recursions as explained in Appendix B.

A single iteration of the resulting methodology – which we term the *twisted Particle-aGRAD* algorithm – is then exactly as the Particle-aGRAD (Algorithm 10), except that $M'_t(\,\cdot\,|\,\cdot\,;\mathbf{u}_t)$, $G'_t(\,\cdot\,;\mathbf{u}_t)$ and $Q'_t(\,\cdot\,;\mathbf{u}_t)$ from Section 4.1 are replaced by $M'_t(\,\cdot\,|\,\cdot\,;\mathbf{u}_{t:T})$, $G'_t(\,\cdot\,;\mathbf{u}_{t:T})$, and $Q'_t(\,\cdot\,;\mathbf{u}_{t:T})$ from this section. When the potential functions $G_t(\mathbf{x}_{t-1:t})$ vary in $\mathbf{x}_{t-1}$, then we can further construct a *twisted Particle-aGRAD+* algorithm by replacing $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)$ in Algorithm 15 and in the denominator of $G'_t(\mathbf{x}_{t-2:t};\mathbf{u}_{t-1:t})$ by $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_{t:T})$.

Proposition 17 (validity of the twisted Particle-aGRAD/Particle-aGRAD+) *Sampling $\tilde{\mathbf{x}}_{1:T}$ given $\mathbf{x}_{1:T}$ via the twisted Particle-aGRAD or twisted Particle-aGRAD+ algorithm induces a Markov kernel which leaves π_T invariant.*

4.5 Relationship with other methods

The algorithms proposed above relate to existing methods as follows.

1. **Generalisation of aGRAD.** For $\kappa = 1$, the Particle-aGRAD algorithm (and similarly the Particle-aGRAD+ algorithm as well as the twisted versions of either) generalises the *auxiliary gradient (aGRAD)* algorithm from Titsias and Papaspiliopoulos (2018, called 'aGrad-z' therein) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where we again suppress the 'time' subscript $t = 1$ everywhere so that $\pi(\mathbf{x}) \propto M(\mathbf{x})G(\mathbf{x})$, where $M(\mathbf{x}) = \mathrm{N}(\mathbf{x};\mathbf{m},\mathbf{C})$. Given that the current state of the Markov chain is $\mathbf{x} = \mathbf{x}^0$ (we can assume that $k = 0$ without loss of generality), Step 1c of Algorithm 10 first refreshes the auxiliary variable by sampling $\mathbf{u} \sim \mathrm{N}(\mathbf{x}^0 + \frac{\delta}{2}\nabla \log G(\mathbf{x}^0), \frac{\delta}{2}\mathbf{I})$ and then proposes $\mathbf{x}^1 \sim \mathrm{N}((\mathbf{I}-\mathbf{A})\mathbf{m} + \mathbf{A}\mathbf{u}, \frac{\delta}{2}\mathbf{A})$, for $\mathbf{A} = (\mathbf{C} + \frac{\delta}{2}\mathbf{I})^{-1}\mathbf{C}$. The remaining steps return $\tilde{\mathbf{x}} := \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\mathrm{aGRAD}}(\mathbf{x}^0,\mathbf{x}^1;\mathbf{u})$, where
$$\begin{aligned}\alpha_{\mathrm{aGRAD}}(\mathbf{x}^0,\mathbf{x}^1;\mathbf{u}) &:= \frac{1-W^0}{1-W^1}\\ &= \frac{\pi(\mathbf{x}^1)\,\mathrm{N}(\mathbf{u};\mathbf{x}^1+\frac{\delta}{2}\nabla\log G(\mathbf{x}^1),\frac{\delta}{2}\mathbf{I})\,\mathrm{N}(\mathbf{x}^0;(\mathbf{I}-\mathbf{A})\mathbf{m}+\mathbf{A}\mathbf{u},\frac{\delta}{2}\mathbf{A})}{\pi(\mathbf{x}^0)\,\mathrm{N}(\mathbf{u};\mathbf{x}^0+\frac{\delta}{2}\nabla\log G(\mathbf{x}^0),\frac{\delta}{2}\mathbf{I})\,\mathrm{N}(\mathbf{x}^1;(\mathbf{I}-\mathbf{A})\mathbf{m}+\mathbf{A}\mathbf{u},\frac{\delta}{2}\mathbf{A})}.\end{aligned}$$
Otherwise, the old state $\tilde{\mathbf{x}} := \mathbf{x}^0 = \mathbf{x}$ is returned as the new state.

2. **Generalisation of mGRAD.** Still taking $\kappa = 1$, the Particle-mGRAD algorithm generalises the *marginal gradient (mGRAD)* algorithm from Titsias and Papaspiliopoulos (2018) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where we use the same notational conventions as in the case of aGRAD above. Step 1c of Algorithm 12 then marginally proposes $\mathbf{x}^1 \sim \mathrm{N}((\mathbf{I}-\mathbf{A})\mathbf{m} + \mathbf{A}[\mathbf{x}^0 + \frac{\delta}{2}\nabla\log G(\mathbf{x}^0)], \mathbf{B})$, where $\mathbf{B} := \frac{\delta}{2}\mathbf{A}^2 + \mathbf{A}$. The remaining steps return $\tilde{\mathbf{x}} := \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\mathrm{mGRAD}}(\mathbf{x}^0,\mathbf{x}^1)$, where
$$\alpha_{\mathrm{mGRAD}}(\mathbf{x}^0,\mathbf{x}^1) := \frac{1-W^0}{1-W^1} = \frac{\pi(\mathbf{x}^1)\,\mathrm{N}((\mathbf{I}-\mathbf{A})\mathbf{m}+\mathbf{A}[\mathbf{x}^1+\frac{\delta}{2}\nabla\log G(\mathbf{x}^1)],\mathbf{B})}{\pi(\mathbf{x}^0)\,\mathrm{N}((\mathbf{I}-\mathbf{A})\mathbf{m}+\mathbf{A}[\mathbf{x}^0+\frac{\delta}{2}\nabla\log G(\mathbf{x}^0)],\mathbf{B})}.$$
Otherwise, the old state $\tilde{\mathbf{x}} := \mathbf{x}^0 = \mathbf{x}$ is returned as the new state. In particular, by Remark 13, in analogue to Section 3.4, we can again interpret aGRAD as a version of mGRAD with 'randomised' acceptance ratio.

3. **Generalisation of a 'preconditioned' Particle-RWM algorithm.** If $\kappa = 0$, then the Particle-aGRAD and Particle-aGRAD+ algorithms reduce to a method recently proposed in Corenflos and Särkkä (2023, Section 4.3), which can be seen as a 'preconditioned' version of the Particle-RWM algorithm.

4.6 Interpolation between CSMC and Particle-MALA/Particle-aMALA

The Particle-MALA (and related methods) proposed in Section 3 may be outperformed by the CSMC algorithm in the case when the prior dynamics are highly informative – in the same way that MALA may be outperformed by the IMH algorithm (with prior as proposal) if the prior dominates the posterior. For instance, in the extreme case that all the potential functions are constant, the CSMC algorithm proposes N trajectories (in addition to the reference path) that are IID samples from π_T (assuming an adaptive or low-variance conditional resampling scheme is used) while the N trajectories proposed by Particle-MALA are still highly correlated with the reference path.

Put differently, the user is faced with the 'tuning problem' of having to decide between the CSMC algorithm on the one hand and the Particle-MALA (and related methods) on the other hand. In this section, we show that the Particle-mGRAD algorithm resolves this tuning problem in the sense that it can be viewed as interpolating between CSMC and Particle-MALA. Specifically, Proposition 18 shows that Particle-mGRAD reduces to the CSMC algorithm if the prior dynamics are highly informative. Conversely, Proposition 19 shows that Particle-mGRAD reduces to the Particle-MALA if the prior dynamics are uninformative. The same results hold for the auxiliary-variable versions: Particle-aMALA and Particle-aGRAD.

We make the following assumptions (assumed to hold for all $t \in [T]$):

A1 For any $\mathbf{x}_t \in \mathcal{X}$, $\mathbf{m}_t(\mathbf{x}_{t-1}) = \mathbf{m}_t$, $\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t$ and $G_t(\mathbf{x}_{t-1:t}) = G_t(\mathbf{x}_t)$ are constant in $\mathbf{x}_{t-1}$, with G_t uniformly bounded on $\mathcal{X}$ and $\mathbf{C}_t$ invertible.

A2 There exist $C_0, C_1 \geq 0$ such that $\kappa \|\nabla \log G_t(\mathbf{x}_t)\|_2 \leq C_0 + C_1 \|\mathbf{x}_t\|_2$.

A3 $\max_{d \in [D]} \int_{\mathcal{X}} x_{t,d}^2 G_t(\mathbf{x}_t) \, \mathrm{d}\mathbf{x}_t < \infty$, where $x_{t,d}$ is the dth component of $\mathbf{x}_t$.

Whenever $T > 1$, Assumption **A1** is strong because it requires the Feynman–Kac model to factorise over time. However, we expect that it could be relaxed at the cost of greatly complicating the arguments. Indeed, note that the model used in Figure 2 does not satisfy this assumption. Assumption **A2** is rather mild, e.g. it holds in a state-space model with Gaussian measurement errors.

In the following, for each $t \in [T]$, we will consider a sequence of prior covariance matrices $(\mathbf{C}_{t,k})_{k \geq 1}$. We will therefore add the subscript k to any quantity which depends on $\mathbf{C}_{t,k}$. We also let $\lambda(\mathbf{A})$ denote the set of eigenvalues of some matrix $\mathbf{A}$. The following propositions are proved in Appendix E.

Proposition 18 *For some $D, T, N \geq 1$, assume **A1**–**A2**, and assume that there exists a sequence $(\lambda_k)_{k \geq 1}$ in $(0, \infty)$ with $\max\{\lambda(\mathbf{C}_{1,k}), \ldots, \lambda(\mathbf{C}_{T,k})\} \leq \lambda_k \to 0$ as $k \to \infty$. Then for any $\varepsilon > 0$, there exists a sequence $(F_{T,k})_{k \geq 1}$ of subsets of $\mathcal{X}^T$ with $\lim_{k \to \infty} \pi_{T,k}(F_{T,k}) = 1$ such that*

1. $\sup_{\mathbf{x}_{1:T} \in F_{T,k}} \| P_{\text{Particle-mGRAD},k}(\,\cdot\,|\mathbf{x}_{1:T}) - P_{\text{CSMC},k}(\,\cdot\,|\mathbf{x}_{1:T}) \|_{\text{TV}} \in \mathrm{O}(\lambda_k^{(1-\varepsilon)/4})$;
2. $\sup_{\mathbf{x}_{1:T} \in F_{T,k}} \| P_{\text{Particle-aGRAD},k}(\,\cdot\,|\mathbf{x}_{1:T}) - P_{\text{CSMC},k}(\,\cdot\,|\mathbf{x}_{1:T}) \|_{\text{TV}} \in \mathrm{O}(\lambda_k^{(1-\varepsilon)/4})$.

Proposition 19 *For some $D, T, N \geq 1$, assume **A1–A3**, and assume that there exists a sequence $(\lambda_k)_{k\geq 1}$ in $(0, \infty)$ with $\min\{\lambda(\mathbf{C}_{1,k}), \dots, \lambda(\mathbf{C}_{T,k})\} \geq \lambda_k \to \infty$ as $k \to \infty$. Then for any $\varepsilon > 0$, there exists a sequence $(F_{T,k})_{k\geq 1}$ of subsets of $\mathcal{X}^T$ with $\lim_{k\to\infty} \pi_{T,k}(F_{T,k}) = 1$ such that*

1. $\sup_{\mathbf{x}_{1:T} \in F_{T,k}} \| P_{\text{Particle-mGRAD},k}(\,\cdot\,|\mathbf{x}_{1:T}) - P_{\text{Particle-MALA},k}(\,\cdot\,|x) \|_{\text{TV}} \in \mathrm{O}(\lambda_k^{-(1-\varepsilon)/4})$;
2. $\sup_{\mathbf{x}_{1:T} \in F_{T,k}} \| P_{\text{Particle-aGRAD},k}(\,\cdot\,|\mathbf{x}_{1:T}) - P_{\text{Particle-aMALA},k}(\,\cdot\,|x) \|_{\text{TV}} \in \mathrm{O}(\lambda_k^{-(1-\varepsilon)/4})$.

As per Sections 2.1.2, 3.4 and 4.5, taking $T = N = 1$ in Propositions 18 and 19 immediately imply that the aGRAD/mGRAD algorithm can be viewed as automatically interpolating between the IMH algorithm with prior as proposal (if the prior is highly informative) and aMALA/MALA (if the prior is highly diffuse). To our knowledge, this interpretation has not been pointed out in the literature. It provides new intuition for the noteworthy performance of aGRAD/mGRAD in Titsias and Papaspiliopoulos (2018).

4.7 Complexity

An iteration of Particle-aGRAD or Particle-aGRAD+ requires computing $T(N + 1)$ gain matrices $\mathbf{A}_t(\mathbf{x}_{t-1}^{a_{t-1}^n}) \in \mathbb{R}^{D\times D}$; and all of these, in general, have a cubic cost in the latent-state dimension D. While this may be reasonable for small enough systems and will be helpful for informative likelihoods, the computational quickly outweighs the statistical benefits of the method. However, when the dynamics have additive noise (13), $\mathbf{A}_t$ does not depend on $\mathbf{x}_{t-1}$. In this case, only T gain matrices are needed and these can be pre-computed, only paying the cubic cost in the dimension upfront rather than at each iteration.

The same applies for the Particle-mGRAD algorithm for which we always require (13) to hold (the auxiliary variables could still be integrated out if (13) is relaxed, but only at the cost of a cubic computational complexity in the number of particles).

However, as for the Particle-RWM algorithm and Particle-MALA-type methods, we need to calibrate the step-size parameters δ_t which changes the gain matrices (so that pre-computation is not possible during the calibration stage). Thankfully, because $\mathbf{A}_t$ and $\mathbf{C}_t$ have the same eigenvectors no matter what δ_t is, it is possible to use similar spectral methods as in Titsias and Papaspiliopoulos (2018) to reduce the complexity of changing δ_t to quadratic.

At first sight, the complexity of the twisted Particle-aGRAD seems quadratic in T as the proposal kernel $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t:T}) = \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t'\mathbf{x}_{t-1} + \mathbf{b}_t', \mathbf{C}_t')$ requires processing $T - t$ auxiliary variables for each time t. However, in Appendix B, we show how $\mathbf{F}_t'$, $\mathbf{b}_t'$ and $\mathbf{C}_t'$ can all be pre-computed based on standard Kalman filter recursions (Kalman, 1960), preserving the linear cost in T and N.

5 Experimental validation and comparison

5.1 Multivariate stochastic volatility model

In this section, we illustrate the efficiency of our methods on a multivariate stochastic volatility model often used as a benchmark for high-dimensional sequential Monte Carlo methodology (see, e.g., Guarniero et al., 2017). This model is a state-space model with a non-linear observation equation:

$$g_t(\mathbf{y}_t|\mathbf{x}_t) = \mathrm{N}(\mathbf{y}_t; \mathbf{0}, \operatorname{diag}(\exp \mathbf{x}_t)),$$

where exp is applied element-wise and where $\mathbf{0}$ is a D-dimensional vector of zeros. The prior on the latent variables is defined through auto-regressive Gaussian dynamics, i.e. for $t > 1$:

$$f_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t), \tag{17}$$

where $\mathbf{m}_t(\mathbf{x}_{t-1}) \coloneqq \varphi \mathbf{x}_{t-1}$ and $\mathbf{C}_t \in \mathbb{R}^{D \times D}$ has diagonal entries τ and off-diagonal entries $\tau\rho$. The initial distribution $f_1(\mathbf{x}_1) = \mathrm{N}(\mathbf{x}_1; \mathbf{m}_1, \mathbf{C}_1)$ is the stationary distribution under the dynamics (17), i.e., $\mathbf{m}_1 \coloneqq \mathbf{0}$ and $\mathbf{C}_1 \coloneqq \mathbf{C}_t/(1 - \varphi^2)$. Here, $\varphi \in (-1, 1)$ is some autocorrelation coefficient, $\rho \in (-1, 1)$ is some intra-asset correlation coefficient and $\tau > 0$.

Throughout our experiments, we take $\varphi = 0.9$, $\rho = 0.25$, and $\tau \in \{0.1, 0.5, 1, 2\}$. The eigenvalues of $\mathbf{C}_t$ are then proportional to τ, i.e., a small value of τ corresponds to highly informative prior dynamics (as in Proposition 18) while a large value of τ corresponds to weakly informative prior dynamics (as in Proposition 19). To make our observations robust to the choice of data set, for each τ, we simulated $M = 5$ independent sets of $T = 128$ observations from the multivariate stochastic volatility model with $D = 30$, i.e., each state $\mathbf{x}_t$ takes values in $\mathcal{X} = \mathbb{R}^{30}$. To make results more easily comparable, experiments for different values of τ use the same random number generator seed.

5.2 Simulation study setup

In addition to the methods proposed in Sections 3 and 4 – potentially without the use of gradient information by taking $\kappa = 0$ – we consider the following benchmark methods:

1. **CSMC.** The CSMC algorithm with bootstrap proposals (Algorithm 1).

2. **Particle-RWM.** The Particle-RWM algorithm (Algorithm 2) from Finke and Thiery (2023) (the special case of Particle-aMALA/Particle-MALA/Particle-aMALA+ if $\kappa = 0$).

3. **MALA and aMALA.** The N-proposal MALA and aMALA which correspond to the Particle-aMALA and Particle-MALA proposed in this work with a single time step (applied to the path-space representation of the Feynman–Kac model, i.e. with a single $(D \times T)$-dimensional state).

4. **aGRAD.** The N-proposal aGRAD algorithm, which corresponds to the Particle-aGRAD proposed in this work with a single time step (again on the path space). We note that we implemented aGRAD using the auxiliary Kalman perspective of Corenflos and Särkkä (2023), making the method complexity scale linearly with T rather

than quadratically with T as in the original version of Titsias and Papaspiliopoulos (2018). We do not compare to mGRAD because computing its particle weights (and hence acceptance ratio) has quadratic complexity in T.

All algorithms use $N + 1 = 32$ particles, and those employing resampling use the conditional 'killing' resampling method (Karppinen et al., 2023), more stable than multinomial resampling, especially with highly informative priors. In each of $M = 5$ independent experiments, algorithms start from the same trajectory generated by a bootstrap particle filter using 32 particles. The samplers run for 10 000 steps to calibrate step-size parameters δ_t, detailed below (note that calibration stabilises much faster). For CSMC, which requires no calibration, the initial 10 000 steps are discarded as warm-up. After calibration, $J = 4$ independent chains start at the final calibration sample, running for $K = 50\,000$ iterations, with the first 5000 discarded as burn-in to decorrelate the chains. Reported statistics are based on these J independent chains.

The step-size parameters δ_t are calibrated for a 75 % acceptance rate, as explained in Appendix F. This slightly exceeds recommendations by, e.g., Roberts and Rosenthal (2001); Titsias and Papaspiliopoulos (2018). This is because we use multiple proposals and the optimal acceptance rate is expected to increase accordingly. Here, 'acceptance rate at time t' refers to the relative frequency of with which the state $\mathbf{x}_t$ is updated. Figure 7 in Appendix G.1 shows stable acceptance rates around 75 % for all methods except CSMC across all time steps. Figure 6 in Appendix G.1 displays calibrated δ_t values.

Experiments ran on a shared computational cluster with identical configurations (32 GB RAM, four processor cores, on shared machines with 2×64-core AMD EPYC 7713 CPUs, clock speed 2.0 GHz). Nonetheless, cluster idiosyncrasies may be present, potentially impacting slower methods like Particle-aMALA+ and Particle-mGRAD.

5.3 Breakdown of CSMC, aMALA and MALA

Our results indicate that CSMC, aMALA, and MALA failed to explore the right regions of the space for all of our chosen levels of informativeness of the latent dynamics ($\tau \in \{0.1, 0.5, 1, 2\}$). Specifically, Figures 8 and 9 in Appendix G.2 show that both the estimated marginal posterior means and also the energy traces of CSMC, aMALA and MALA differ substantially from those of all the other algorithms. Here, 'energy trace' refers to $\log \pi_T(\mathbf{x}_{1:T}) + \text{const}$ computed on the sampled trajectories throughout the sampling procedure. Since CSMC, aMALA and MALA thus do not produce reliable approximations of the distribution of interest, we omit these methods from our discussions in the sequel.

In the remainder of this section, we compare the remaining algorithms in terms of the *effective sample size (ESS)* computed using the method of Vehtari et al. (2021) with $J = 4$ independent chains. We also compare the algorithms in terms of ESS per second (ESS/s). The latter corresponds to the time it would take to obtain a 'perfect' sample using the Markov chain. In the main manuscript, we only show results for the median ESS and averaged over all T time steps. Appendix G.3 shows detailed results for the minimum and maximum ESS and ESS/s (which are qualitatively similar to the median case) separately for each time step $t = 1, \dots, T$.

5.4 Benefits of exploiting gradient information

Figure 3 compares the median ESS ('unnormalised') and median ESS/s ('per second') of Particle-aMALA, Particle-MALA and Particle-aMALA+, i.e., for those methods which do not make any Gaussian assumption about the prior dynamics. Recall that these differ from the baseline: the Particle-RWM algorithm, only in the use of gradient information. Thus, the left panel in Figure 3 illustrates the benefits (in terms of ESS) of exploiting gradient information. Notably:

- the improvement of Particle-MALA over Particle-aMALA is marginal at best. Possibly, the difference between both algorithms decreases with N but this calls for further investigation;
- the 'smoothing-gradient' variant Particle-aMALA+ dominates all other alternatives for all values of τ, with up to three times the performance of Particle-RWM and twice that of the 'filter-gradient' variants Particle-aMALA and Particle-MALA;
- the performance of all shown methods improves as τ increases: this is because the posterior distribution then decorrelates in time, and, therefore, the fact that they all use proposals which are separable (in the sense discussed in Section 4) stops being penalising.

The right panel in Figure 3 shows that the use of gradient information is still beneficial even when accounting for the cost of gradient calculation. However, the relative performance of the gradient-based methods is now less clear: whilst Particle-aMALA+ has the highest sampling efficiency, it incurs additional overheads due to computing twice as many gradients as Particle-aMALA and Particle-MALA and due to dealing with non-Markovian potentials.

5.5 Benefits of exploiting Gaussian prior dynamics

In this section, we demonstrate that exploiting the latent (conditionally) Gaussian dynamics of the model (as done by Particle-aGRAD, Particle-mGRAD and twisted Particle-aGRAD) can improve the sampling efficiency.

First, in Figure 4, we illustrate the performance of those methods which require (at most) *conditionally* Gaussian prior dynamics as in (8), i.e., of Particle-aGRAD and Particle-mGRAD (note that the later also requires $\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t$ (13) to retain linear computational complexity in N). In terms of ESS, these methods improve upon the 'filter-gradient' methods Particle-aMALA and Particle-MALA but they are still dominated by the 'smoothing-gradient' method Particle-aMALA+. However, the picture is less clear when accounting for computation time.

Second, in Figure 5, we illustrate the performance of the twisted Particle-aGRAD which requires *unconditionally* Gaussian prior dynamics as in (15). As a baseline, we use the aGRAD algorithm from Titsias and Papaspiliopoulos (2018) as it makes the same assumption. The twisted Particle-aGRAD strongly outperforms this baseline and also all the other algorithms. Furthermore, the dominance of the twisted Particle-aGRAD algorithm does not disappear when accounting for the computation time. This is because, in contrast to Particle-aMALA+, its modified model is still Markovian and because it only requires the computation of a single gradient per particle and time step.

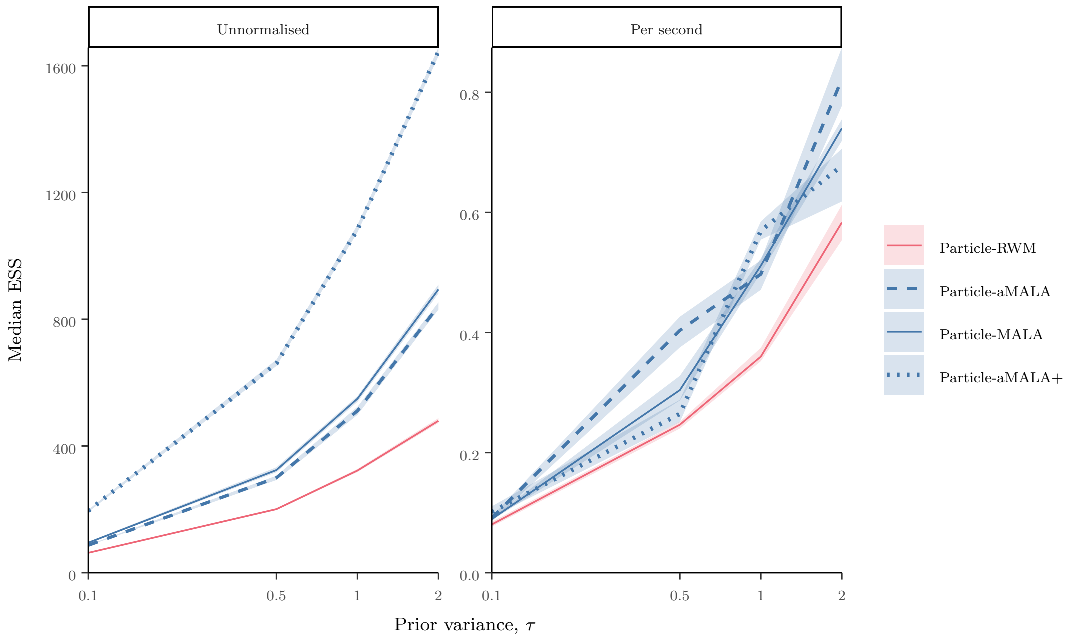


Figure 3: Performance of those proposed methods which do not require (conditionally or unconditionally) Gaussian prior dynamics compared with the existing Particle-RWM algorithm as a baseline.

6 Conclusion

6.1 Summary

We have proposed a methodology for Bayesian inference about the latent states in high-dimensional state-space models and beyond. Our methodology combines the CSMC algorithm (Andrieu et al., 2010) with sophisticated 'classical' MCMC algorithms like MALA (Besag, 1994), aMALA (Titsias and Papaspiliopoulos, 2018), aGRAD/mGRAD (Titsias, 2011; Titsias and Papaspiliopoulos, 2018) or PCNL (Cotter et al., 2013) to retain *the best of both worlds:*

- from the CSMC algorithm, our methods retain the ability to exploit the model's 'decorrelation-over-time' structure which permits favourable scaling with the number of time steps, T;
- from 'classical' MCMC algorithms, our methods retain the ability to use gradient-informed, local proposals which permits favourable scaling with the dimension of the states, D.

Most of our proposed algorithms (except the 'marginal' ones) leverage an auxiliary-variable perspective recently proposed in Corenflos and Särkkä (2023). We name our algorithms Particle-aMALA, Particle-MALA, Particle-aGRAD, Particle-mGRAD and Particle-PCNL. This is motivated by the fact that if $T = N = 1$ (where $N \in \mathbb{N}$ is the number of particles), they reduce to the 'classical' MCMC algorithms: aMALA, MALA, PCNL, aGRAD and

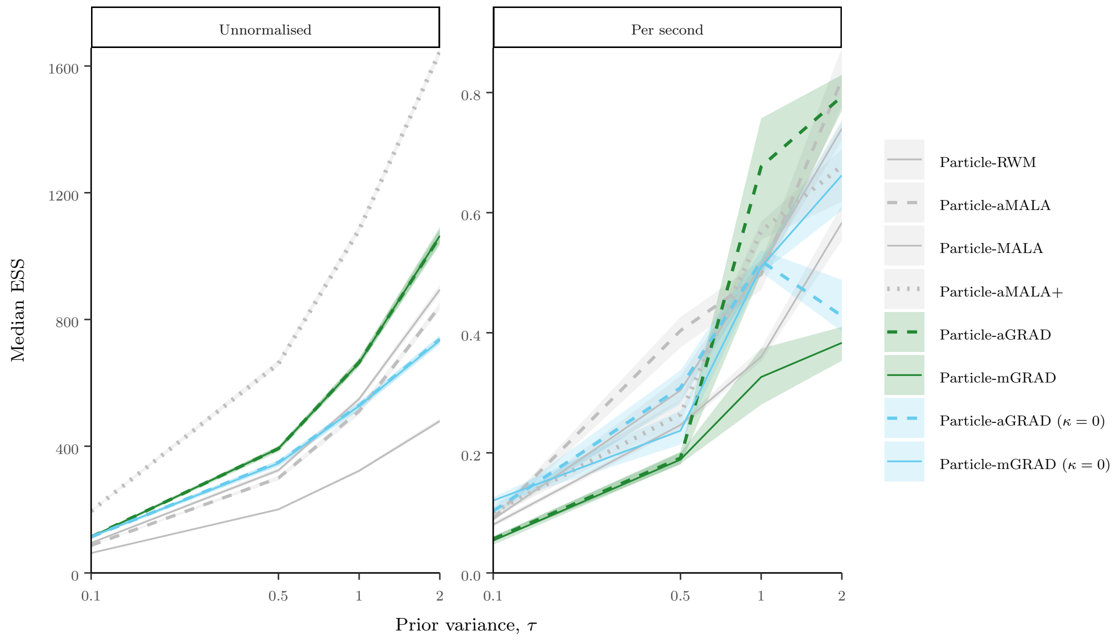


Figure 4: Performance of the proposed methods which require only *conditionally* Gaussian prior dynamics (8), i.e., $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$. The Particle-mGRAD algorithm (with any $\kappa \in \{0, 1\}$) also requires that $\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t$ is constant to avoid superlinear computational complexity in N. Results that were already shown in the previous figure are greyed out.

mGRAD, respectively. Furthermore, if $T = 1$ but $N > 1$, our methods constitute novel multi-proposal versions of such ‘classical’ MCMC algorithms which may themselves be of interest with a view to exploiting parallelisation.

The generalisation of such ‘classical’ MCMC algorithms to $T > 1$ time steps is, however, not unique. And so we have presented additional variants named Particle-aMALA+, Particle-aGRAD+ and *twisted* Particle-aGRAD/Particle-aGRAD+. These can be viewed as ‘lookahead’ methods because their proposals employ ‘smoothing’ rather than ‘filter’ gradients or use information contained in future auxiliary variables. Notably, if $N = 1$ but $T > 1$, then the IMH, RWM, aMALA and aGRAD algorithm can still be recovered as a special case of slightly modified versions of the CSMC, Particle-RWM, Particle-aMALA+, and twisted Particle-aGRAD+ algorithms (and also of the twisted Particle-aGRAD algorithm if $G_t(\mathbf{x}_{t-1:t})$ is constant in $\mathbf{x}_{t-1}$). Specifically, this modification would entail that the latter use no resampling (i.e., they instead set $a_t^n = n$ for all $n \in [N]_0$ and all $t \in [T-1]$), use *ancestral tracing* instead of backward sampling (i.e., they instead set $l_t = a_t^{l_{t+1}}$ for all $t \in [T-1]$) and use $\delta_1 = \ldots = \delta_T$.

We have further proved that the Particle-aGRAD/Particle-mGRAD algorithms have the desirable property that they naturally recover (a) the CSMC algorithm if the prior dynamics are highly informative (i.e., if the target posterior distribution is dominated by the prior); (b) the Particle-aMALA/Particle-MALA if the prior dynamics are completely

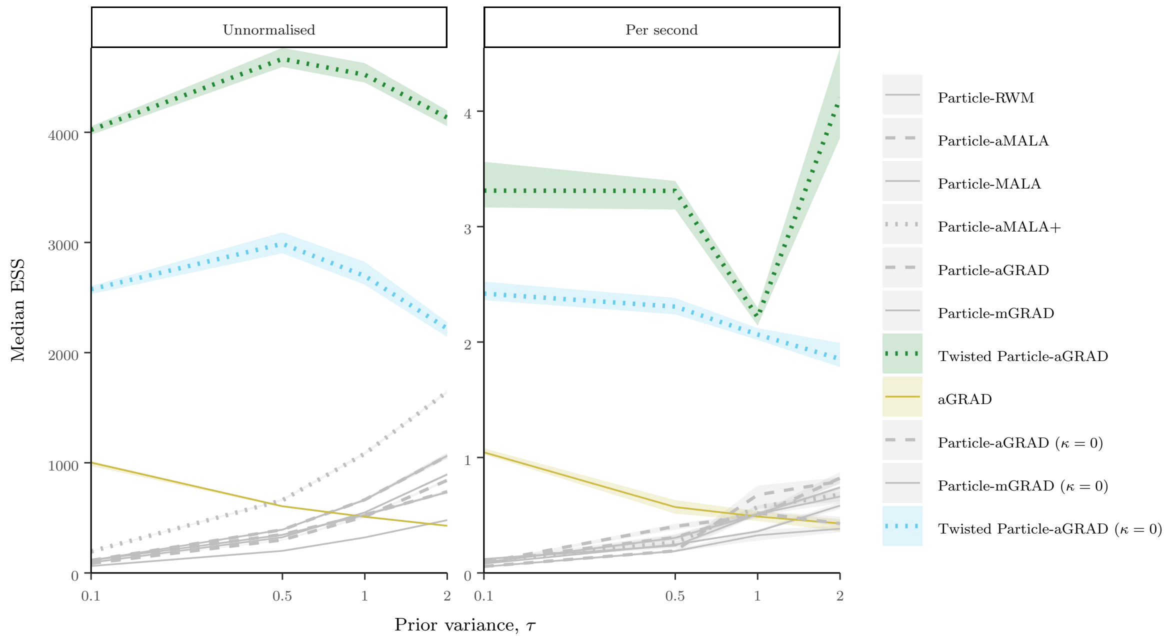


Figure 5: Performance of the proposed methods which require *unconditionally* Gaussian prior dynamics (15), i.e., $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{b}_t, \mathbf{C}_t)$, compared with aGRAD (which also requires (15)) as baseline. Results that were already shown in the previous two figures are greyed out. The abnormally large computation time of the twisted Particle-aGRAD for $\kappa = \tau = 1$ was likely caused by some computational-cluster idiosyncrasies.

uninformative (i.e., if the target posterior distribution is dominated by the likelihood). This property independently helps explain the impressive performance of aGRAD and mGRAD reported in Titsias and Papaspiliopoulos (2018).

Our methods have enabled Bayesian inference in a multivariate stochastic volatility model with $D = 30$ assets and $T = 128$ observations (3840 unknowns in total) in which neither CSMC nor aMALA/MALA gave reliable estimates. In particular, in this application, our twisted Particle-aGRAD algorithm strongly outperformed the existing sophisticated aGRAD algorithm – even when accounting for computation time.

6.2 Limitations

The main limitations of our methods are the same as in all gradient-based 'classical' MCMC algorithms. First, they require continuously differentiable target densities (more precisely, the densities $Q_t(\mathbf{x}_{t-1:t})$ need to be computable and differentiable pointwise). This requirement is slightly softened for the methods of Section 4 where only the likelihood $G_t(\mathbf{x}_{t-1:t})$ is required to be differentiable, at the cost of needing (at least conditionally) Gaussian prior dynamics $M_t(\mathbf{x}_t|\mathbf{x}_{t-1})$. The favourable scaling with the dimension D also typically requires target densities to be sufficiently smooth (see, e.g., Vogrinc and Kendall, 2021, for counterexamples). Second, while it improves mixing properties, locality in MCMC is often detrimental when exploring multi-modal posteriors. This is inherited by our methods which, too, explore the space by local moves.

6.3 Extensions

Our work opens up multiple avenues for further research.

- The algorithms proposed in this work can be extended to more general graphical models, i.e., they can be combined with suitably 'conditional' versions of the divide-&-conquer sequential Monte Carlo algorithm from Lindsten et al. (2017). For instance, for a particular graphical model, such a 'conditional' scheme was recently described in Corenflos et al. (2022, Section 3).

- Particle-aMALA can be incorporated straightforwardly into the methodology from Corenflos et al. (2022) to reduce the computation time per MCMC update from $\mathrm{O}(T)$ to $\mathrm{O}(\log_2 T)$ (for some fixed dimension D) on parallel architectures. While less directly obvious (because of the non-Markovianity of the auxiliary target), the smoothing-gradient version Particle-aMALA+ is likely parallelisable, too, by simply extending the framework to compute weight functions over three time steps rather than two. It is however less clear that Particle-MALA is parallelisable, as the marginalisation has to be done across two time steps rather than one as presented in Section 3.2.

- All our algorithms can be straightforwardly extended to use other resampling schemes than conditional multinomial resampling, e.g., conditional systematic resampling. In fact, in our experiments, we used the conditional killing resampling which is stable under low-informative likelihoods (Karppinen et al., 2023), a regime that may happen in our case when δ_t takes very small values at calibration time.

- In this work, we have left aside the question of choosing δ_t and have elected to take it to correspond to a 75 % acceptance rate throughout. It is however clear that its optimal value (and the optimal value of the acceptance rate) depends on the number of proposals N and on the dimension D. An optimal-scaling analysis (see Roberts and Rosenthal, 2001, and references therein) of the methods proposed in this work is therefore needed. An optimal-scaling analysis for a related algorithm without backward sampling and without gradient or prior-informed proposals can be found in Malory (2021).

- In Section 4 (in which we propose Particle-aGRAD and variations thereof), we have assumed that the covariance matrices $\mathbf{C}_t(\mathbf{x}_{t-1})$ (or $\mathbf{C}_t$) are non-singular. However, it is worth noting that proposal kernels used by the methods from Section 4 remain valid if the covariance matrices are singular, in the sense that they are *still* absolutely continuous w.r.t. the true dynamics. However, the use of backward sampling is no longer possible for such degenerate dynamics. Instead, one must resort to ancestral tracing, i.e., taking $l_t \coloneqq a_t^{l_{t+1}}$, for $t = T-1, \dots, 1$. However, in this case, N needs to grow with T at a suitable rate which depends on the stability properties of the model, but at least linearly (Andrieu et al., 2018; Lindsten et al., 2015). An alternative is to fix N but decrease the step sizes δ_t with t (which would automatically occur when using adaptation based on acceptance rates as considered in work), as considered in Malory (2021) for a related method.

- Our proposed algorithms consider solely first-order gradient information. A natural extension would therefore be to incorporate second-order expansions or preconditioned and adaptive versions of the Particle-MALA variants. Another obvious direction of study is to extend our methodology to other MCMC kernels, such as the recently proposed Barker's robust proposal (Livingstone and Zanella, 2022), or non-reversible discrete-time kernels such as the discrete bouncy particle sampler Sherlock and Thiery (2022). Other natural extensions would consist of adapting the methodology to non-continuous spaces, e.g., using methods from Zanella (2020); Rhodes and Gutmann (2022), or constrained spaces.

Author contributions

A.C. and A.F. jointly developed the methodology, writing was primarily done by A.F., A.C. implemented and conducted the experiments, after which both A.C. and A.F. edited and reviewed the final manuscript.

Appendix A. Particle extensions of PCN(L)

A.1 Particle-aPCNL

In this section, we extend the *preconditioned Crank–Nicolson–Langevin (PCNL)* algorithm (and also the *preconditioned Crank–Nicolson (PCN)* algorithm recovered by setting $\kappa = 0$) (Cotter et al., 2013) to $T > 1$ time steps and $N > 1$. As a by-product, we derive an 'auxiliary-variable' version of PCNL which was mentioned, but not explicitly stated, in Titsias and Papaspiliopoulos (2018). Throughout this section, we assume the prior dynamics are conditionally Gaussian, $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$ as in (8).

Throughout this section, we use the parametrisation of the PCNL algorithm from Titsias and Papaspiliopoulos (2018)[3], i.e., we set

$$\beta_t \coloneqq \frac{2}{2 + \delta_t} \in (0, 1),$$

where $\delta_t > 0$ is again the step size at time t. Note that this implies that $\frac{1-\beta_t}{\beta_t} = \frac{\delta_t}{2}$.

The first method proposed in this section is termed *Particle-aPCNL*. Conditional on the auxiliary variables $\mathbf{u}_{1:T}$, it can be viewed as a CSMC algorithm whose proposal kernels are those of the fully-adapted auxiliary particle filter for the state-space model defined by the Gaussian transitions $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t(\mathbf{x}_{t-1}), \mathbf{C}_t(\mathbf{x}_{t-1}))$ from (8) and 'pseudo observations' $\mathbf{u}_t$ with $p(\mathbf{u}_t|\mathbf{x}_t) = \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \frac{\delta_t}{2}\mathbf{C}_t(\mathbf{x}_{t-1}))$. We now write

$$\begin{aligned} M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) &\coloneqq p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \\ &\propto M_t(\mathbf{x}_t|\mathbf{x}_{t-1})\, \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \tfrac{\delta_t}{2}\mathbf{C}_t(\mathbf{x}_{t-1})) \\ &\propto \mathrm{N}(\mathbf{x}_t; \mathbf{m}'_t(\mathbf{x}_{t-1}, \mathbf{u}_t), \mathbf{C}'_t(\mathbf{x}_{t-1})), \end{aligned} \tag{18}$$

with

$$\mathbf{m}'_t(\mathbf{x}_{t-1}, \mathbf{u}_t) \coloneqq \beta_t \mathbf{u}_t + (1 - \beta_t)\mathbf{m}_t(\mathbf{x}_{t-1}), \tag{19}$$

$$\mathbf{C}'_t(\mathbf{x}_{t-1}) \coloneqq (1 - \beta_t)\mathbf{C}_t(\mathbf{x}_{t-1}), \tag{20}$$

as well as

$$G'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t) \coloneqq Q_t(\mathbf{x}_{t-1:t}) \frac{\mathrm{N}(\mathbf{u}_t; \mathbf{x}_t + \kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1})\nabla_{\mathbf{x}_t} \log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{C}_t(\mathbf{x}_{t-1}))}{M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)}, \tag{21}$$

and $Q'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t) \coloneqq M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)G'_t(\mathbf{x}_{t-1:t}; \mathbf{u}_t)$. Here, $\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1}) \in \mathbb{R}^{D\times D}$ is some preconditioning matrix whose choice is discussed in Section A.5 below.

A single iteration of the Particle-aPCNL algorithm is then as follows.

3. In the parametrisation from Cotter et al. (2013), we would have $\delta_t \in [0, 2]$ $\beta_t = \frac{2-\delta_t}{2+\delta_t}$

Algorithm 20 (Particle-aPCNL) *Implement Algorithm 1 but replace the particle proposal (Step 1c) and the weight calculation (Step 1d) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1})\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{C}_t(\mathbf{x}_{t-1}))$, *and* $\mathbf{x}_t^n \sim M_t'(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_t)$, *for* $n \in [N]_0 \setminus \{k_t\}$,

1d. for $n \in [N]_0$, *set* $w_t^n \propto G_t'(\mathbf{x}_{t-1:t}^{(n)}; \mathbf{u}_t)$,

and also replace $Q_{t+1}(\,\cdot\,)$ *in the backward kernel in Step 3 by* $Q_{t+1}'(\,\cdot\,;\mathbf{u}_t)$.

Proposition 21 (validity of Particle-aPCNL) *Sampling* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *via Algorithm 20 induces a Markov kernel* $P_{\text{Particle-aPCNL}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

A.2 Particle-PCNL

In this section, in analogy to the Particle-MALA and Particle-mGRAD algorithms from the main manuscript, we analytically integrate out the auxiliary variables $\mathbf{u}_t$ which appeared in the weights of the Particle-aPCNL algorithm. As in the case of the Particle-mGRAD algorithm, we assume that the covariance matrices appearing in the conditionally Gaussian mutation kernel (8) do not depend on the previous state, i.e., $\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t$ (13).

A single iteration of the resulting methodology – which we term the *Particle-PCNL* algorithm – is as follows, where we write

$$\begin{aligned}\log H_{t,\boldsymbol{\phi}}(\mathbf{x},\mathbf{v},\bar{\mathbf{x}},\bar{\mathbf{v}}) = \tfrac{1}{2}(\beta_t^{-1} + N + 1)(\mathbf{x}-\mathbf{v})^{\mathrm{T}}\mathbf{G}_t(\mathbf{x}-\mathbf{v})\\ - \tfrac{1}{2}N\beta_t(\mathbf{x}+\boldsymbol{\phi})^{\mathrm{T}}\mathbf{G}_t(\mathbf{x}+\boldsymbol{\phi})\\ + (N+1)(\bar{\mathbf{x}}-\bar{\mathbf{v}})^{\mathrm{T}}\mathbf{G}_t(\mathbf{v}+\boldsymbol{\phi})\\ - (\mathbf{x}-\mathbf{v})^{\mathrm{T}}\mathbf{G}_t(\mathbf{x}+\boldsymbol{\phi}),\end{aligned}$$

for

$$\mathbf{G}_t \coloneqq \frac{\beta_t}{(1-\beta_t)(1+N\beta_t)}\mathbf{C}_t^{-1} = \frac{2(\delta_t+2)}{\delta_t(\delta_t+2+N)}\mathbf{C}_t^{-1}. \tag{22}$$

Algorithm 22 (Particle-PCNL) *Implement Algorithm 1 but replace the particle proposal (Step 1c) and the weight calculation (Step 1d) by*

1c. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1})\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t}), \frac{\delta_t}{2}\mathbf{C}_t)$, *and* $\mathbf{x}_t^n \sim M_t'(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_t)$, *for* $n \in [N]_0 \setminus \{k_t\}$,

1d. set $\bar{\mathbf{x}}_t \coloneqq \frac{1}{N+1}\sum_{n=0}^N \mathbf{x}_t^n$, $\mathbf{v}_t^n \coloneqq (1-\beta_t)\mathbf{m}_t(\mathbf{x}_{t-1}^{a_{t-1}^n})$, $\bar{\mathbf{v}}_t \coloneqq \frac{1}{N+1}\sum_{n=0}^N \mathbf{v}_t^n$, *and, for* $n \in [N]_0$,

$$w_t^n \propto Q_t(\mathbf{x}_{t-1:t}^{(n)}) H_{t,\kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1}^{a_{t-1}^n})\nabla_{\mathbf{x}_t^n}\log G_t(\mathbf{x}_{t-1:t}^{(n)})}(\mathbf{x}_t^n, \mathbf{v}_t^n, \bar{\mathbf{x}}_t, \bar{\mathbf{v}}_t).$$

In the same way as outlined in Remarks 6 and 13, the Particle-aPCNL algorithm can be viewed as an ‘exact approximation’ of the Particle-PCNL algorithm.

Proposition 23 (validity of Particle-PCNL) *Sampling $\tilde{\mathbf{x}}_{1:T}$ given $\mathbf{x}_{1:T}$ via Algorithm 22 induces a Markov kernel $P_{\text{Particle-PCNL}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ which leaves π_T invariant.*

A.3 Particle-aPCNL+

In this section, similar to the Particle-aMALA+ and Particle-aGRAD+ algorithms from the main manuscript, we extend the Particle-aPCNL algorithm to incorporate gradients w.r.t. the 'smoothing' potential $G_{1:T}(\mathbf{x}_{1:T}) = \prod_{t=1}^T G_t(\mathbf{x}_{t-1:t})$ rather than w.r.t. the 'filtering' potential $\prod_{s=1}^t G_s(\mathbf{x}_{s-1:s})$.

For $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)$ and $G'_t(\mathbf{x}_{t-1:t}, \mathbf{u}_t)$ still defined as in the Particle-aPCNL algorithm (i.e., as in (18) and (21)), we now write

$$\begin{aligned}&G'_t(\mathbf{x}_{t-2:t}, \mathbf{u}_{1:T})\\ &\coloneqq G'_t(\mathbf{x}_{t-1:t}, \mathbf{u}_t)\frac{\mathrm{N}(\mathbf{u}_{t-1}; \mathbf{x}_{t-1} + \kappa\frac{\delta_{t-1}}{2}\widetilde{\mathbf{C}}_{t-1}(\mathbf{x}_{t-2})\nabla_{\mathbf{x}_{t-1}} \log G_{1:T}(\mathbf{x}_{1:T}), \frac{\delta_{t-1}}{2}\mathbf{C}_{t-1}(\mathbf{x}_{t-2}))}{\mathrm{N}(\mathbf{u}_{t-1}; \mathbf{x}_{t-1} + \kappa\frac{\delta_{t-1}}{2}\widetilde{\mathbf{C}}_{t-1}(\mathbf{x}_{t-2})\nabla_{\mathbf{x}_{t-1}} \log G_{t-1}(\mathbf{x}_{t-2:t-1}), \frac{\delta_{t-1}}{2}\mathbf{C}_{t-1}(\mathbf{x}_{t-2}))},\end{aligned}$$

as well as $Q'_t(\mathbf{x}_{t-2:t}; \mathbf{u}_{t-1:t}) \coloneqq M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_t)G'_t(\mathbf{x}_{t-2:t}; \mathbf{u}_{t-1:t})$, where we note that

$$\nabla_{\mathbf{x}_t} \log G_{1:T}(\mathbf{x}_{1:T}) = \nabla_{\mathbf{x}_t}[\log G_t(\mathbf{x}_{t-1:t}) + \log G_{t+1}(\mathbf{x}_{t:t+1})].$$

A single iteration of the resulting methodology – which we term the *Particle-aPCNL+* algorithm – is as follows.

Algorithm 24 (Particle-aPCNL+) *Implement Algorithm 1 but replace the particle proposal (Step 1c), the weight calculation (Step 1d), and backward sampling (Step 3) by*

1c. *sample* $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1})\nabla_{\mathbf{x}_t} \log G_T(\mathbf{x}_{1:T}), \frac{\delta_t}{2}\mathbf{C}_t(\mathbf{x}_{t-1}))$, *and* $\mathbf{x}_t^n \sim M'_t(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_t)$, *for* $n \in [N]_0 \setminus \{k_t\}$,

1d. *for* $n \in [N]_0$, *set* $w_t^n \propto G'_t(\mathbf{x}_{t-2:t}^{(n)}; \mathbf{u}_{t-1:t})$,

3. *for* $t = T-1, \dots, 1$, *sample* $l_t = i \in [N]_0$ w.p.

$$\frac{W_t^i Q'_{t+1}((\mathbf{x}_{t-1:t}^{(i)}, \mathbf{x}_{t+1}^{l_{t+1}}); \mathbf{u}_{t:t+1})Q'_{t+2}((\mathbf{x}_t^i, \mathbf{x}_{t+1}^{l_{t+1}}, \mathbf{x}_{t+2}^{l_{t+2}}); \mathbf{u}_{t+1:t+2})}{\sum_{n=0}^N W_t^n Q'_{t+1}((\mathbf{x}_{t-1:t}^{(n)}, \mathbf{x}_{t+1}^{l_{t+1}}); \mathbf{u}_{t:t+1})Q'_{t+2}((\mathbf{x}_t^n, \mathbf{x}_{t+1}^{l_{t+1}}, \mathbf{x}_{t+2}^{l_{t+2}}); \mathbf{u}_{t+1:t+2})}.$$

Note that if $G_t(\mathbf{x}_{t-1:t}) = G_t(\mathbf{x}_t)$ does not depend on $\mathbf{x}_{t-1}$, then the Particle-aPCNL+ algorithm coincides with the Particle-aPCNL algorithm. However, when $G_t(\mathbf{x}_{t-1:t})$ varies highly in $\mathbf{x}_{t-1}$, their behaviours may differ substantially.

Proposition 25 (validity of Particle-aPCNL+) *Sampling $\tilde{\mathbf{x}}_{1:T}$ given $\mathbf{x}_{1:T}$ via Algorithm 24 induces a Markov kernel $P_{\text{Particle-aPCNL+}}(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ which leaves π_T invariant.*

A.4 Twisted Particle-aPCNL(+)

In analogue to the twisted Particle-aGRAD and twisted Particle-aGRAD+ algorithms, we can again construct 'twisted' versions of the Particle-aPCNL and Particle-aPCNL+ algorithms, under the assumption that $M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{b}_t, \mathbf{C}_t)$, i.e., (15).

We start with the twisted Particle-aPCNL algorithm. We now write

$$\begin{aligned}
M'_t(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_{t:T}) &\coloneqq p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_{t:T}) \\
&\propto \int_{\mathcal{X}^{T-t}} \left[\prod_{s=t}^{T} \mathrm{N}(\mathbf{x}_s;\mathbf{F}_s\mathbf{x}_{s-1}+\mathbf{b}_s,\mathbf{C}_s)\,\mathrm{N}(\mathbf{u}_s;\mathbf{x}_s,\tfrac{\delta_s}{2}\mathbf{C}_t)\right]\mathrm{d}\mathbf{x}_{t+1:T} \\
&\propto \mathrm{N}(\mathbf{x}_t;\mathbf{F}'_t\mathbf{x}_{t-1}+\mathbf{b}'_t,\mathbf{C}'_t), \qquad (23)\\
G'_t(\mathbf{x}_{t-1:t};\mathbf{u}_{t:T}) &\coloneqq Q_t(\mathbf{x}_{t-1:t})\frac{\mathrm{N}(\mathbf{u}_t;\mathbf{x}_t+\kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1})\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t}),\frac{\delta_t}{2}\mathbf{C}_t)}{M'_t(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_{t:T})},
\end{aligned}$$

as well as $Q'_t(\mathbf{x}_{t-1:t};\mathbf{u}_{t:T}) \coloneqq M'_t(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_{t:T})G'_t(\mathbf{x}_{t-1:t};\mathbf{u}_{t:T})$. Here, $\mathbf{b}'_t \in \mathbb{R}^D$ and $\mathbf{F}'_t, \mathbf{C}'_t \in \mathbb{R}^{D\times D}$ can again be obtained via the Kalman-filtering recursions given in Appendix B.

A single iteration of the resulting methodology – which we term the *twisted Particle-aPCNL* algorithm – is then exactly as the Particle-aPCNL algorithm (Algorithm 20), except that $M'_t(\,\cdot\,|\,\cdot\,;\mathbf{u}_t)$, $G'_t(\,\cdot\,;\mathbf{u}_t)$ and $Q'_t(\,\cdot\,;\mathbf{u}_t)$ from Section A.1 are replaced by $M'_t(\,\cdot\,|\,\cdot\,;\mathbf{u}_{t:T})$, $G'_t(\,\cdot\,;\mathbf{u}_{t:T})$ and $Q'_t(\,\cdot\,;\mathbf{u}_{t:T})$ from this section. When the potential functions $G_t(\mathbf{x}_{t-1:t})$ vary in $\mathbf{x}_{t-1}$, then we can further construct a *twisted Particle-aGRAD+* algorithm by replacing $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)$ in Algorithm 24 and in the denominator of $G'_t(\mathbf{x}_{t-2:t};\mathbf{u}_{t-1:t})$ by $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_{t:T})$.

Proposition 26 (validity of the twisted Particle-aPCNL/Particle-aPCNL+) *Sampling $\tilde{\mathbf{x}}_{1:T}$ given $\mathbf{x}_{1:T}$ via the twisted Particle-aPCNL or twisted Particle-aPCNL+ algorithm induces a Markov kernel which leaves π_T invariant.*

A.5 Choice of preconditioning matrix

There is some degree of freedom in choosing the preconditioning matrices $\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1}) \in \mathbb{R}^{D\times D}$ in the algorithms presented above.

1. A simple option which does not require further assumptions is to take

$$\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1}) \coloneqq \mathbf{C}_t(\mathbf{x}_{t-1}).$$

2. If we make the stronger model assumption that $\mathbf{m}_t(\mathbf{x}_{t-1}) = \mathbf{F}_t\mathbf{x}_{t-1}+\mathbf{b}_t$ and $\mathbf{C}_t(\mathbf{x}_{t-1}) = \mathbf{C}_t$ (15) (which is assumed to hold for the twisted versions of the Particle-aPCNL and Particle-aPCNL+ algorithms anyway), then we could alternatively set

$$\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1}) = \widetilde{\mathbf{C}}_t \coloneqq \sum_{s=1}^{T} \mathbf{\Sigma}_{s,t}, \qquad (24)$$

where $\mathbf{\Sigma}_{s,t} \in \mathbb{R}^{D\times D}$ is the block (s,t) in the covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{TD\times TD}$ of the prior dynamics $M_{1:T}(\mathbf{x}_{1:T})$, i.e.,

$$\mathbf{\Sigma}_{s,t} = \begin{cases} \mathbf{F}_s\mathbf{\Sigma}_{s-1,t}, & \text{if } s > t,\\ \mathbf{\Sigma}_{t,s}^{\mathrm{T}}, & \text{if } s < t,\\ \mathbf{\Sigma}_t, & \text{if } s = t, \end{cases}$$

where $\mathbf{\Sigma}_t$ can be found via the recursion from Step 1 of Algorithm 27 from Appendix B. As discussed below, this specification has the potentially useful implication that the algorithm reduces to an 'auxiliary-variable' version of the PCNL algorithm on the path space in the absence of resampling and backward sampling. Unfortunately, evaluating the preconditioning matrices $\widetilde{\mathbf{C}}_t$ is likely to incur a quadratic computational complexity in T which we prefer to avoid.

3. A compromise (which retains linear computational complexity in T) may be to truncate the above sum by setting

$$\widetilde{\mathbf{C}}_t \coloneqq \sum_{s=(t-L)\vee 1}^{(t+L)\wedge T} \mathbf{\Sigma}_{s,t},$$

for some $L \in [T]_0$ (note that this still requires the model assumption (15)).

A.6 Relationship with other methods

The algorithms proposed above relate to existing methods as follows.

1. **Generalisation of aPCNL.** For $\kappa = 1$, the Particle-aPCNL algorithm (and similarly the Particle-aPCNL+ algorithm as well as the twisted versions of either) generalises an *auxiliary preconditioned Crank–Nicolson–Langevin (aPCNL)* algorithm (which was mentioned but not explicitly derived in Titsias and Papaspiliopoulos (2018)) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where we again suppress the 'time' subscript $t = 1$ everywhere so that $\pi(\mathbf{x}) \propto M(\mathbf{x})G(\mathbf{x})$, where $M(\mathbf{x}) = \mathrm{N}(\mathbf{x}; \mathbf{m}, \mathbf{C})$. We also take $\widetilde{\mathbf{C}} \coloneqq \mathbf{C}$. Given that the current state of the Markov chain is $\mathbf{x} = \mathbf{x}^0$ (we can assume that $k = 0$ without loss of generality), Step 1c of Algorithm 20 first refreshes the auxiliary variable by sampling $\mathbf{u} \sim \mathrm{N}(\mathbf{x}^0 + \frac{\delta}{2}\mathbf{C}\nabla \log G(\mathbf{x}^0), \frac{\delta}{2}\mathbf{C})$ and then proposes $\mathbf{x}^1 \sim \mathrm{N}((1-\beta)\mathbf{m} + \beta\mathbf{u}, (1-\beta)\mathbf{C})$. The remaining steps return $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\text{aPCNL}}(\mathbf{x}^0, \mathbf{x}^1; \mathbf{u})$, where

$$\begin{aligned}\alpha_{\text{aPCNL}}&(\mathbf{x}^0, \mathbf{x}^1; \mathbf{u})\\ &\coloneqq \frac{1 - W^0}{1 - W^1}\\ &= \frac{\pi(\mathbf{x}^1)\,\mathrm{N}(\mathbf{u}; \mathbf{x}^1 + \frac{\delta}{2}\mathbf{C}\nabla \log G(\mathbf{x}^1), \frac{\delta}{2}\mathbf{C})\,\mathrm{N}(\mathbf{x}^0; (1-\beta)\mathbf{m} + \beta\mathbf{u}, (1-\beta)\mathbf{C})}{\pi(\mathbf{x}^0)\,\mathrm{N}(\mathbf{u}; \mathbf{x}^0 + \frac{\delta}{2}\mathbf{C}\nabla \log G(\mathbf{x}^0), \frac{\delta}{2}\mathbf{C})\,\mathrm{N}(\mathbf{x}^1; (1-\beta)\mathbf{m} + \beta\mathbf{u}, (1-\beta)\mathbf{C})}.\end{aligned}$$

Otherwise, the old state $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^0 = \mathbf{x}$ is returned as the new state. If $N = 1$ and $\kappa = 1$ but $T > 1$ then the aPCNL algorithm could still be recovered as a special case of (a slightly modified version of) the twisted Particle-aPCNL+ algorithm (and also of the twisted Particle-aPCNL algorithm if the potential functions $G_t(\mathbf{x}_{t-1:t})$ do not depend on $\mathbf{x}_{t-1}$) if the latter uses no resampling and ancestral tracing instead of backward sampling, and if $\delta_1 = \ldots = \delta_T$ and if the preconditioning matrices are specified via (24). However, we do not recommend this choice of preconditioning matrix as it leads to squared computational complexity in T (also incurred by aPCNL).

2. **Generalisation of PCNL.** Still taking $\kappa = 1$, the Particle-PCNL algorithm generalises the *preconditioned Crank–Nicolson–Langevin (PCNL)* algorithm (Cotter et al., 2013) in the sense that the former reduces to the latter if $T = N = 1$. This can be seen as follows, where we use the same notational conventions as in the case of aPCNL above. Step 1c of Algorithm 22 then marginally proposes $\mathbf{x}^1 \sim \mathrm{N}((1-\beta)\mathbf{m} + \beta[\mathbf{x}^0 + \frac{\delta}{2}\mathbf{C}\nabla \log G(\mathbf{x}^0)], (1-\beta^2)\mathbf{C})$. The remaining steps return $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^1$ as the new state with acceptance probability $1 \wedge \alpha_{\mathrm{PCNL}}(\mathbf{x}^0, \mathbf{x}^1)$, where

$$\begin{aligned}\alpha_{\mathrm{PCNL}}(\mathbf{x}^0, \mathbf{x}^1) &\coloneqq \frac{1 - W^0}{1 - W^1} \\ &= \frac{\pi(\mathbf{x}^1)\,\mathrm{N}((1-\beta)\mathbf{m} + \beta[\mathbf{x}^1 + \frac{\delta}{2}\nabla \log G(\mathbf{x}^1)], (1-\beta^2)\mathbf{C})}{\pi(\mathbf{x}^0)\,\mathrm{N}((1-\beta)\mathbf{m} + \beta[\mathbf{x}^0 + \frac{\delta}{2}\nabla \log G(\mathbf{x}^0)], (1-\beta^2)\mathbf{C})}.\end{aligned}$$

Otherwise, the old state $\tilde{\mathbf{x}} \coloneqq \mathbf{x}^0 = \mathbf{x}$ is returned as the new state. In particular, in analogue to Section 3.4, we can again interpret aPCNL as a version of PCNL with 'randomised' acceptance ratio.

Appendix B. Twisted proposals

In this section, we detail the mutation kernel $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{t:T})$ used by the twisted Particle-aGRAD/Particle-aGRAD+ (16) and twisted Particle-aPCNL/Particle-aPCNL+ (23) algorithms. This mutation kernel can be thought of as the fully-twisted particle-filter proposal for the state-space model which is defined by the Gaussian transitions $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = M_t(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{b}_t, \mathbf{C}_t)$ from (15) and observation densities $p(\mathbf{u}_t|\mathbf{x}_t) = \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \frac{\delta_t}{2}\mathbf{V}_t)$, where we take $\mathbf{V}_t \coloneqq \mathbf{I}$ in the case of the twisted Particle-aGRAD or twisted Particle-aGRAD+ algorithm and $\mathbf{V}_t \coloneqq \mathbf{C}_t$ in the case of the twisted Particle-aPCNL or twisted Particle-aPCNL+ algorithm:

$$M'_t(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{t:T}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{1:T}) = \mathrm{N}(\mathbf{x}_t; \mathbf{F}'_t\mathbf{x}_{t-1} + \mathbf{b}'_t, \mathbf{C}'_t).$$

General algorithm. Algorithm 27 explains how the twisted-proposal parameters $\mathbf{b}'_t \in \mathbb{R}^D$ and $\mathbf{F}'_t, \mathbf{C}'_t \in \mathbb{R}^{D\times D}$ can be calculated at linear complexity in T, independently of the total number of particles, N. Notably, Algorithm 27 does not require $\mathbf{C}_t$ to be invertible.

Algorithm 27 (twisted-proposal parameters) *At the start of an iteration of the twisted Particle-aGRAD, Particle-aGRAD+, Particle-aPCNL or Particle-aPCNL+ algorithm (after having sampled all the auxiliary variables $\mathbf{u}_{1:T}$ upfront – e.g., as in Algorithm 33 from Appendix D.1 – which is possible because these only depend on the reference path),*

1. *recursively compute the moments of $p(\mathbf{x}_t) = \mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, for $t = 1, \dots, T$, as*
$$\begin{aligned}\boldsymbol{\mu}_t &\coloneqq \mathbf{F}_t \boldsymbol{\mu}_{t-1} + \mathbf{b}_t,\\ \boldsymbol{\Sigma}_t &\coloneqq \mathbf{F}_t \boldsymbol{\Sigma}_{t-1} \mathbf{F}_t^{\mathrm{T}} + \mathbf{C}_t,\end{aligned}$$
if $t > 1$, and with initial condition $\boldsymbol{\mu}_1 = \mathbf{b}_1$ and $\boldsymbol{\Sigma}_1 = \mathbf{C}_1$,

2. *recursively compute the moments of the time-reversed state transition kernels $p(\mathbf{x}_t|\mathbf{x}_{t+1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t^{\leftarrow}\mathbf{x}_{t+1} + \mathbf{b}_t^{\leftarrow}, \mathbf{C}_t^{\leftarrow})$, for $t = T-1, \dots, 1$, as*
$$\begin{aligned}\mathbf{F}_t^{\leftarrow} &\coloneqq \mathbf{F}_{t+1}\boldsymbol{\Sigma}_t\boldsymbol{\Sigma}_{t+1}^{-1},\\ \mathbf{b}_t^{\leftarrow} &\coloneqq \boldsymbol{\mu}_t - \mathbf{F}_t^{\leftarrow}\boldsymbol{\mu}_{t+1},\\ \mathbf{C}_t^{\leftarrow} &\coloneqq \boldsymbol{\Sigma}_t - \mathbf{F}_t^{\leftarrow}\boldsymbol{\Sigma}_t\mathbf{F}_{t+1}^{\mathrm{T}}.\end{aligned}$$

3. *run the Kalman filtering recursion for the time-reversed state-space model (i.e., with observation densities $p(\mathbf{u}_t|\mathbf{x}_t) = \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \frac{\delta_t}{2}\mathbf{V}_t)$, initial distribution $p(\mathbf{x}_t)$ and time-reversed state transitions $p(\mathbf{x}_t|\mathbf{x}_{t+1})$ found in Steps 1 and 2) to compute the moments of $p(\mathbf{x}_t|\mathbf{u}_{t:T}) = \mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{\leftarrow}, \boldsymbol{\Sigma}_{t|t}^{\leftarrow})$, for $t = T, \dots, 1$,*

4. *set $\mathbf{F}_1' \coloneqq \mathbf{0}_{D\times D}$, $\mathbf{b}_1' \coloneqq \boldsymbol{\mu}_{1|1}^{\leftarrow}$ as well as $\mathbf{C}_1' \coloneqq \boldsymbol{\Sigma}_{1|1}^{\leftarrow}$, and, for $t = 2, \dots, T$,*
$$\begin{aligned}\mathbf{F}_t' &\coloneqq \boldsymbol{\Sigma}_{t|t}^{\leftarrow}\mathbf{F}_{t-1}^{\leftarrow}(\mathbf{C}_{t-1}^{\leftarrow} + \mathbf{F}_{t-1}^{\leftarrow}\boldsymbol{\Sigma}_{t|t}^{\leftarrow}\{\mathbf{F}_{t-1}^{\leftarrow}\}^{\mathrm{T}})^{-1},\\ \mathbf{b}_t' &\coloneqq \boldsymbol{\mu}_{t|t}^{\leftarrow} - \mathbf{F}_t'(\mathbf{F}_{t-1}^{\leftarrow}\boldsymbol{\mu}_{t|t}^{\leftarrow} + \mathbf{b}_{t-1}^{\leftarrow}),\\ \mathbf{C}_t' &\coloneqq (\mathbf{I} - \mathbf{F}_t'\mathbf{F}_{t-1}^{\leftarrow})\boldsymbol{\Sigma}_{t|t}^{\leftarrow}.\end{aligned}$$

Algorithm 27 is justified by the decomposition

$$\begin{aligned}M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{t:T}) &\propto p(\mathbf{x}_{t-1:t}|\mathbf{u}_{t:T})\\ &= p(\mathbf{x}_{t-1}|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{u}_{t:T})\\ &= \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t^{\leftarrow}\mathbf{x}_{t+1} + \mathbf{b}_t^{\leftarrow}, \mathbf{C}_t^{\leftarrow})\,\mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{\leftarrow}, \boldsymbol{\Sigma}_{t|t}^{\leftarrow}) \qquad (25)\\ &\propto \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t'\mathbf{x}_{t-1} + \mathbf{b}_t', \mathbf{C}_t'), \qquad (26)\end{aligned}$$

where, as described in Algorithm 27, $p(\mathbf{x}_t|\mathbf{u}_{t:T}) = \mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{\leftarrow}, \boldsymbol{\Sigma}_{t|t}^{\leftarrow})$ is the time-t filter for the time-reversed state-space model with the same observation densities $p(\mathbf{u}_t|\mathbf{x}_t) = \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t, \frac{\delta_t}{2}\mathbf{V}_t)$ as before but with initial distribution $p(\mathbf{x}_t) = \mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_T, \boldsymbol{\Sigma}_T)$ and time-reversed state transitions $p(\mathbf{x}_t|\mathbf{x}_{t+1}) = \mathrm{N}(\mathbf{x}_t; \mathbf{F}_t^{\leftarrow}\mathbf{x}_{t+1} + \mathbf{b}_t^{\leftarrow}, \mathbf{C}_t^{\leftarrow})$. Thus, in Algorithm 27:

- Step 1 calculates the marginal prior distributions of the states. To see this, note that this step is effectively the Kalman-filter recursion without observations. Note that if the prior dynamics are stationary with stationary distribution $\mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then Step 1 can be skipped (because then $\boldsymbol{\mu}_t = \boldsymbol{\mu}$ and $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}$, for any $t \in [T]$).

- Step 2 computes the parameters of the time-reversed transition kernels and follows from standard Gaussian algebra (see, e.g., Särkkä and Svensson, 2023, Section A.1) by noting that
$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t+1} \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \boldsymbol{\mu}_t \\ \boldsymbol{\mu}_{t+1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_t & \mathbf{F}_{t+1}\boldsymbol{\Sigma}_t \\ \boldsymbol{\Sigma}_t\mathbf{F}_{t+1}^{\mathrm{T}} & \boldsymbol{\Sigma}_{t+1} \end{bmatrix}\right).$$
- Step 4 derives (26) from (25) and corresponds to a single update step of a Kalman filter (Särkkä and Svensson, 2023, Chapter 6, Equation 6.21), where $\mathbf{x}_{t-1}$ plays the rôle of an observation.

Alternative algorithm for invertible covariance matrices. If $\mathbf{C}_t$ is invertible for all $t \in [T]$, then the twisted-proposal parameters can be alternatively computed via Algorithm 28 which may be slightly simpler to implement for some users and which may provide additional numerical advantages in the case of explosive prior dynamics.

Algorithm 28 (twisted-proposal parameters: alternative) *At the start of an iteration of the twisted Particle-aGRAD, Particle-aGRAD+, Particle-aPCNL or Particle-aPCNL+ algorithm (after having sampled all the auxiliary variables* $\mathbf{u}_{1:T}$ *upfront – e.g., as in Algorithm 33 from Appendix D.1 – which is possible because these only depend on the reference path),*

1. *run the Kalman filtering recursion to compute the moments of* $p(\mathbf{x}_t|\mathbf{u}_{1:t-1}) = \mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})$*, for* $t = 1, \dots, T$*,*
2. *run the Kalman smoothing (a.k.a. Rauch–Tung–Striebel smoothing) recursion to compute the moments of* $p(\mathbf{x}_t|\mathbf{u}_{1:T}) = \mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|T}, \boldsymbol{\Sigma}_{t|T})$*, for* $t = T, T-1, \dots, 1$*,*
3. *for* $t \in [T]$*, set*
$$\begin{aligned} \mathbf{C}'_t &\coloneqq [\mathbf{C}_t^{-1} + \boldsymbol{\Sigma}_{t|T}^{-1} - \boldsymbol{\Sigma}_{t|t-1}^{-1}]^{-1}, \\ \mathbf{F}'_t &\coloneqq \mathbf{C}'_t[\mathbf{C}_t^{-1}\mathbf{b}_t + \boldsymbol{\Sigma}_{t|T}^{-1}\boldsymbol{\mu}_{t|T} - \boldsymbol{\Sigma}_{t|t-1}^{-1}\boldsymbol{\mu}_{t|t-1}], \\ \mathbf{b}'_t &\coloneqq \mathbf{C}'_t\mathbf{C}_t^{-1}\mathbf{F}_t. \end{aligned}$$

Algorithm 28 is justified by the decomposition

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{1:T}) &\propto p(\mathbf{x}_{t-1:t}, \mathbf{u}_{1:T}) \\ &= p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{u}_{1:T})p(\mathbf{x}_t|\mathbf{u}_{1:T}) \\ &\propto \frac{p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_t|\mathbf{u}_{1:t-1})}p(\mathbf{x}_t|\mathbf{u}_{1:T}) \\ &= \frac{\mathrm{N}(\mathbf{x}_t; \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{b}_t, \mathbf{C}_t)}{\mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})}\mathrm{N}(\mathbf{x}_t; \boldsymbol{\mu}_{t|T}, \boldsymbol{\Sigma}_{t|T}) \\ &\propto \mathrm{N}(\mathbf{x}_t; \mathbf{F}'_t\mathbf{x}_{t-1} + \mathbf{b}'_t, \mathbf{C}'_t). \end{aligned}$$

Appendix C. Integrating out the auxiliary variables

In this section, we prove a few lemmata which are used in subsequent sections.

- **Lemmata 29 and 30.** Lemmata 29 and 30 derive the determinant and inverse of a certain simple block matrix which appears repeatedly in the remainder of this section and also in Appendix E.
- **Lemma 31.** Lemma 31 will allow us to derive marginal proposal distributions of the various algorithms, i.e., the distribution of $\mathbf{x}_t^{-k_t} = (\mathbf{x}_t^0, \dots, \mathbf{x}_t^{k_t-1}, \mathbf{x}_t^{k_t+1}, \dots, \mathbf{x}_t^N)$ conditional on k_t, $\mathbf{x}_t^{k_t} = \mathbf{x}_t$ and all the particles and ancestor indices with time indices $s < t$, but with the auxiliary variables $\mathbf{u}_{1:T}$ integrated out.
- **Lemma 32.** Lemma 32 will allow us to evaluate the particle weights used in the 'marginal' algorithms (Particle-MALA, Particle-mGRAD and Particle-PCNL) at linear complexity in N although the weight of the nth particle depends on the values of all other particles.

C.1 Properties of a particular block matrix

Let $\mathbf{I}_M$ denote the $(M \times M)$ identity matrix. When $M = D$ we continue to leave out the subscript. Furthermore, let $\mathbf{1}_{M \times N} \in \{1\}^{M \times N}$ and denote a matrix in which every element is 1. For matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{D \times D}$, define the block matrix

$$\mathcal{M}_N(\mathbf{A}, \mathbf{B}) \coloneqq \mathbf{I}_N \otimes \mathbf{A} + \mathbf{1}_{N \times N} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}+\mathbf{B} & \mathbf{B} & \dots & \mathbf{B} \\ \mathbf{B} & \mathbf{A}+\mathbf{B} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{B} \\ \mathbf{B} & \dots & \mathbf{B} & \mathbf{A}+\mathbf{B} \end{bmatrix} \in \mathbb{R}^{(DN) \times (DN)} \tag{27}$$

Lemma 29 *For $N, D \in \mathbb{N}$, let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{D \times D}$. Then,*

$$\det(\mathcal{M}_N(\mathbf{A}, \mathbf{B})) = \det(\mathbf{A})^{N-1} \det(\mathbf{A} + N\mathbf{B}).$$

Proof Subtracting the last row of $\mathcal{M}_N(\mathbf{A}, \mathbf{B})$ from all other rows and then adding the sum of the first $N-1$ columns to the last column gives the upper-triangular block matrix

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \mathbf{A} & \mathbf{0} \\ \mathbf{B} & \dots & \mathbf{B} & \mathbf{A}+N\mathbf{B} \end{bmatrix}.$$

This proves the result. ☐

Lemma 30 *For $N, D \in \mathbb{N}$, let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{D \times D}$, such that $\mathbf{A}$ and $(\mathbf{A} + N\mathbf{B})$ are invertible. Then*

$$\mathcal{M}_N(\mathbf{A}, \mathbf{B})^{-1} = \mathcal{M}_N(\mathbf{A}^{-1}, -(\mathbf{A} + N\mathbf{B})^{-1}\mathbf{B}\mathbf{A}^{-1}).$$

Proof We must have $\mathcal{M}_N(\mathbf{A},\mathbf{B})\mathcal{M}_N(\mathbf{F},\mathbf{G}) = \mathbf{I}_{DN}$ and hence

$$\begin{aligned}(\mathbf{A}+\mathbf{B})(\mathbf{F}+\mathbf{G}) + (N-1)\mathbf{B}\mathbf{G} &= \mathbf{I},\\ \mathbf{B}(\mathbf{F}+\mathbf{G}) + (\mathbf{A}+\mathbf{B})\mathbf{G} + (N-2)\mathbf{B}\mathbf{G} &= \mathbf{0}.\end{aligned}$$

This implies $\mathbf{F} = \mathbf{A}^{-1}$ and $\mathbf{G} = -(\mathbf{A}+N\mathbf{B})^{-1}\mathbf{B}\mathbf{A}^{-1}$. □

C.2 Conditional and marginal proposal distributions

In this section, for any tuple $(\mathbf{z}_0,\dots,\mathbf{z}_N)$ of values in $\mathcal{X} \coloneqq \mathbb{R}^D$ and any $n \in [N]_0$, we write $\mathbf{z}_{-n} \coloneqq (\mathbf{z}_0,\dots,\mathbf{z}_{n-1},\mathbf{z}_{n+1},\dots,\mathbf{z}_N)$. Given some $N \in \mathbb{N}$, $n \in [N]_0$, $\mathbf{x}_n, \boldsymbol{\phi}_n \in \mathcal{X}$, we consider the following joint distribution on $\mathcal{X}^{N+1}$:

$$q_{-n}(\mathbf{u}, \mathbf{x}_{-n}|\mathbf{x}_n) \coloneqq \mathrm{N}(\mathbf{u}; \mathbf{x}_n + \boldsymbol{\phi}_n, \mathbf{E}) \prod_{\substack{m=0\\ m\neq n}}^{N} \mathrm{N}(\mathbf{x}_m; \mathbf{v}_m + \mathbf{H}_m\mathbf{u}, \mathbf{D}_m), \tag{28}$$

where, for any $m \in [N]_0$, $\mathbf{v}_m \in \mathcal{X}$ and $\mathbf{H}_m \in \mathbb{R}^{D\times D}$, and $\mathbf{D}_m, \mathbf{E} \in \mathbb{R}^{D\times D}$ are positive definite and symmetric.

To simplify the presentation – and with some abuse of notation since we use the same symbols for tuples and their vectorised versions – we write

$$\mathbf{x}_{-n} \coloneqq \begin{bmatrix}\mathbf{x}_0\\ \vdots\\ \mathbf{x}_{n-1}\\ \mathbf{x}_{n+1}\\ \vdots\\ \mathbf{x}_n\end{bmatrix}, \quad \mathbf{v}_{-n} \coloneqq \begin{bmatrix}\mathbf{v}_0\\ \vdots\\ \mathbf{v}_{n-1}\\ \mathbf{v}_{n+1}\\ \vdots\\ \mathbf{v}_n\end{bmatrix}, \quad \mathbf{H}_{-n} \coloneqq \begin{bmatrix}\mathbf{H}_0\\ \vdots\\ \mathbf{H}_{n-1}\\ \mathbf{H}_{n+1}\\ \vdots\\ \mathbf{H}_n\end{bmatrix}, \quad \mathbf{D}_{-n} \coloneqq \operatorname{diag}\left(\begin{bmatrix}\mathbf{D}_0\\ \vdots\\ \mathbf{D}_{n-1}\\ \mathbf{D}_{n+1}\\ \vdots\\ \mathbf{D}_n\end{bmatrix}\right),$$

where, in the last expression, diag induces a block-diagonal matrix. With this notation, we can formulate (28) equivalently as

$$q_{-n}(\mathbf{u}, \mathbf{x}_{-n}|\mathbf{x}_n) = \mathrm{N}(\mathbf{u}; \mathbf{x}_n + \boldsymbol{\phi}_n, \mathbf{E})\, \mathrm{N}(\mathbf{x}_{-n}; \mathbf{v}_{-n} + \mathbf{H}_{-n}\mathbf{u}, \mathbf{D}_{-n}). \tag{29}$$

Lemma 31 *For any $n \in [N]_0$ and $\mathbf{x}_n \in \mathcal{X}$,*

1. *the marginal distribution of $\mathbf{x}_{-n}$ given $\mathbf{x}_n$ under (29) is*

$$q_{-n}(\mathbf{x}_{-n}|\mathbf{x}_n) = \mathrm{N}(\mathbf{x}_{-n}; \boldsymbol{\mu}_{-n}, \boldsymbol{\Sigma}_{-n}),$$

where

$$\begin{aligned}\boldsymbol{\mu}_{-n} &\coloneqq \mathbf{v}_{-n} + \mathbf{H}_{-n}(\mathbf{x}_n + \boldsymbol{\phi}_n),\\ \boldsymbol{\Sigma}_{-n} &\coloneqq \mathbf{D}_{-n} + \mathbf{H}_{-n}\mathbf{E}\mathbf{H}_{-n}^{\mathrm{T}};\end{aligned}$$

2. *the conditional distribution of $\mathbf{u}$ given $\mathbf{x}_{-n}$ and $\mathbf{x}_n$ under (29) is*

$$q_{-n}(\mathbf{u}|\mathbf{x}_{-n}, \mathbf{x}_n) = \mathrm{N}(\mathbf{u}; \mathbf{x}_n + \boldsymbol{\phi}_n + \mathbf{K}[\mathbf{x}_{-n} - \mathbf{v}_{-n} - \mathbf{H}_{-n}(\mathbf{x}_n + \boldsymbol{\phi}_n)], (\mathbf{I} - \mathbf{K}\mathbf{H}_{-n})\mathbf{E}),$$

where $\mathbf{K} \coloneqq \mathbf{E}\mathbf{H}_{-n}^{\mathrm{T}}(\mathbf{D}_{-n} + \mathbf{H}_{-n}\mathbf{E}\mathbf{H}_{-n}^{\mathrm{T}})^{-1}$.

Proof This follows by simple algebra (see, e.g., Särkkä and Svensson, 2023, Appendix A.1). □

Lemma 32 *Assume now that* $\mathbf{H}_m = \mathbf{H}$ *and* $\mathbf{D}_m = \mathbf{D}$*, for any* $m \in [N]_0$*. Then, with the notation from Lemma 31,*

$$q_{-n}(\mathbf{x}_{-n}|\mathbf{x}_n) \propto H_{\phi_n}(\mathbf{x}_n, \mathbf{v}_n, \bar{\mathbf{x}}, \bar{\mathbf{v}}) I((\mathbf{x}_m - \mathbf{v}_m)_{m=0}^N),$$

where

1. $\mathbf{z}_{0:N} \mapsto I(\mathbf{z}_{0:N})$ *is invariant under any permutation of its arguments;*
2. $\bar{\mathbf{x}} := \frac{1}{N+1}\sum_{m=0}^N \mathbf{x}_n$*, and* $\bar{\mathbf{v}} := \frac{1}{N+1}\sum_{m=0}^N \mathbf{v}_n$*, and*

$$\begin{aligned}
&\log H_\phi(\mathbf{x}, \mathbf{v}, \bar{\mathbf{x}}, \bar{\mathbf{v}}) \\
&\quad = \tfrac{1}{2}(\mathbf{x}-\mathbf{v})^{\mathrm{T}}(\mathbf{D}^{-1}+\mathbf{G})(\mathbf{x}-\mathbf{v}) \\
&\qquad - \tfrac{1}{2}N(\mathbf{x}+\boldsymbol{\phi})^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}(\mathbf{x}+\boldsymbol{\phi}) \\
&\qquad - (N+1)(\bar{\mathbf{x}}-\bar{\mathbf{v}})^{\mathrm{T}}[\mathbf{G}(\mathbf{x}-\mathbf{v}) - (\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}(\mathbf{x}+\boldsymbol{\phi})] \\
&\qquad - (\mathbf{x}-\mathbf{v})^{\mathrm{T}}(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}(\mathbf{x}+\boldsymbol{\phi}),
\end{aligned}$$

whose evaluation complexity does not depend on N*. Here,*

$$\begin{aligned}
\mathbf{G} &:= (\mathbf{D}+N\mathbf{H}\mathbf{E}\mathbf{H}^{\mathrm{T}})^{-1}\mathbf{H}\mathbf{E}\mathbf{H}^{\mathrm{T}}\mathbf{D}^{-1} && (30)\\
&= \mathbf{D}^{-1}\mathbf{H}\mathbf{E}(\mathbf{E}+N\mathbf{E}\mathbf{H}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{H}\mathbf{E})^{-1}\mathbf{E}\mathbf{H}^{\mathrm{T}}\mathbf{D}^{-1}. && (31)
\end{aligned}$$

Proof The equivalence of (30) and (31) follows from the *push-through identity* (Henderson and Searle, 1981). By assumption, $\boldsymbol{\Sigma}_{-n} = \mathcal{M}_N(\mathbf{D}, \mathbf{H}\mathbf{E}\mathbf{H}^{\mathrm{T}})$. Thus, Lemma 30 gives

$$\boldsymbol{\Sigma}_{-n}^{-1} = \mathcal{M}_N(\mathbf{D}^{-1}, -\mathbf{G}).$$

In particular, letting $\otimes$ be the Kronecker product, this implies that

$$\begin{aligned}
\boldsymbol{\Sigma}_{-n}^{-1}\mathbf{H}_{-n} &= \mathbf{1}_{N\times 1}\otimes[(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}],\\
\mathbf{H}_{-n}^{\mathrm{T}}\boldsymbol{\Sigma}_{-n}^{-1}\mathbf{H}_{-n} &= N\mathbf{H}^{\mathrm{T}}(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}.
\end{aligned}$$

Therefore, defining

$$\mathbf{x} := \begin{bmatrix}\mathbf{x}_0\\ \vdots \\ \mathbf{x}_N\end{bmatrix}, \quad \mathbf{v} := \begin{bmatrix}\mathbf{v}_0\\ \vdots \\ \mathbf{v}_N\end{bmatrix}, \quad \boldsymbol{\Sigma} := \mathcal{M}_{N+1}(\mathbf{D}^{-1}, -\mathbf{G})^{-1},$$

we have

$$
\begin{aligned}
&q_{-n}(\mathbf{x}_{-n}|\mathbf{x}_n)\\
&\quad\propto \exp\left(-\tfrac{1}{2}\left[(\mathbf{x}_{-n}-\mathbf{v}_{-n}-\mathbf{H}_{-n}(\mathbf{x}_n+\boldsymbol{\phi}_n))^{\mathrm{T}}\boldsymbol{\Sigma}_{-n}^{-1}(\mathbf{x}_{-n}-\mathbf{v}_{-n}-\mathbf{H}_{-n}(\mathbf{x}_n+\boldsymbol{\phi}_n))\right]\right)\\
&\quad= \exp\left(-\tfrac{1}{2}\left[(\mathbf{x}_{-n}-\mathbf{v}_{-n})^{\mathrm{T}}\boldsymbol{\Sigma}_{-n}^{-1}(\mathbf{x}_{-n}-\mathbf{v}_{-n})\right.\right.\\
&\qquad\qquad\qquad + (\mathbf{x}_n+\boldsymbol{\phi}_n)^{\mathrm{T}}\mathbf{H}_{-n}^{\mathrm{T}}\boldsymbol{\Sigma}_{-n}^{-1}\mathbf{H}_{-n}(\mathbf{x}_n+\boldsymbol{\phi}_n)\\
&\qquad\qquad\qquad \left.\left. - 2(\mathbf{x}_{-n}-\mathbf{v}_{-n})^{\mathrm{T}}\boldsymbol{\Sigma}_{-n}^{-1}\mathbf{H}_{-n}(\mathbf{x}_n+\boldsymbol{\phi}_n)\right]\right)\\
&\quad= \exp\left(-\tfrac{1}{2}\left[(\mathbf{x}-\mathbf{v})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{v})\right.\right.\\
&\qquad\qquad\qquad - (\mathbf{x}_n-\mathbf{v}_n)^{\mathrm{T}}(\mathbf{D}^{-1}-\mathbf{G})(\mathbf{x}_n-\mathbf{v}_n)\\
&\qquad\qquad\qquad + 2(\mathbf{x}_{-n}-\mathbf{v}_{-n})^{\mathrm{T}}[\mathbf{1}_{N\times 1}\otimes \mathbf{G}](\mathbf{x}_n-\mathbf{v}_n)\\
&\qquad\qquad\qquad + N(\mathbf{x}_n+\boldsymbol{\phi}_n)^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}(\mathbf{x}_n+\boldsymbol{\phi}_n)\\
&\qquad\qquad\qquad \left.\left. - 2(\mathbf{x}_{-n}-\mathbf{v}_{-n})^{\mathrm{T}}\boldsymbol{\Sigma}_{-n}^{-1}\mathbf{H}_{-n}(\mathbf{x}_n+\boldsymbol{\phi}_n)\right]\right)\\
&\quad= \exp\left(-\tfrac{1}{2}\left[(\mathbf{x}-\mathbf{v})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{v})\right.\right.\\
&\qquad\qquad\qquad - (\mathbf{x}_n-\mathbf{v}_n)^{\mathrm{T}}(\mathbf{D}^{-1}-\mathbf{G})(\mathbf{x}_n-\mathbf{v}_n)\\
&\qquad\qquad\qquad + N(\mathbf{x}_n+\boldsymbol{\phi}_n)^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}(\mathbf{x}_n+\boldsymbol{\phi}_n)\\
&\qquad\qquad\qquad + 2(\mathbf{x}-\mathbf{v})^{\mathrm{T}}\left\{\mathbf{1}_{(N+1)\times 1}\otimes[\mathbf{G}(\mathbf{x}_n-\mathbf{v}_n)-(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}(\mathbf{x}_n+\boldsymbol{\phi}_n)]\right\}\\
&\qquad\qquad\qquad \left.\left. - 2(\mathbf{x}_n-\mathbf{v}_n)^{\mathrm{T}}[\mathbf{G}(\mathbf{x}_n-\mathbf{v}_n)-(\mathbf{D}^{-1}-N\mathbf{G})\mathbf{H}(\mathbf{x}_n+\boldsymbol{\phi}_n)]\right]\right)\\
&\quad= H_{\boldsymbol{\phi}_n}(\mathbf{x}_n,\mathbf{v}_n,\bar{\mathbf{x}},\bar{\mathbf{v}})I((\mathbf{x}_m-\mathbf{v}_m)_{m=0}^N),
\end{aligned}
$$

with

$$I((\mathbf{x}_m-\mathbf{v}_m)_{m=0}^N)\propto \exp(-\tfrac{1}{2}(\mathbf{x}-\mathbf{v})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{v})).$$

This completes the proof. □

Appendix D. Generic algorithms and proof of Propositions 4–26

In this section, we prove that the algorithms proposed in this work leave π_T invariant. To this end, we first prove the validity of two generic algorithms.

- **Generic auxiliary algorithm.** The first generic algorithm includes auxiliary variables $\mathbf{u}_t$ in the space and admits the 'auxiliary-variable' based algorithms: Particle-aMALA, Particle-aGRAD, Particle-aPCNL as well as their smoothing-gradient ('+') and twisted versions, as special cases. Its proof extends the auxiliary-variable interpretation of the Particle-RWM algorithm which was given in Corenflos and Särkkä (2023).

- **Generic marginal algorithm.** The second generic algorithm integrates out the auxiliary variables and admits the 'marginal' algorithms from the main manuscript (Particle-MALA, Particle-mGRAD, Particle-PCNL). Its proof relies on an argument previously given in Finke et al. (2016).

D.1 Generic auxiliary algorithm

Define an extended target distribution

$$\pi'_T(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \coloneqq \pi_T(\mathbf{x}_{1:T}) \prod_{t=1}^{T} \mathrm{N}(\mathbf{u}_t; \mathbf{x}_t + \Phi_t(\mathbf{x}_{1:T}), \mathbf{E}_t(\mathbf{x}_{t-1:t})), \tag{32}$$

where, for any $t \in [T]$, $\Phi_t \colon \mathcal{X}^T \to \mathcal{X}$ is a function satisfying

$$\Phi_t(\mathbf{x}_{1:T}) = \begin{cases} \boldsymbol{\phi}_T(\mathbf{x}_{t-T:T}), & \text{if } t = T, \\ \boldsymbol{\phi}_t(\mathbf{x}_{t-1:t}) + \boldsymbol{\psi}_t(\mathbf{x}_{t:t+1}), & \text{otherwise,} \end{cases}$$

and $\mathbf{E}_t(\mathbf{x}_{t-1:t}) \in \mathbb{R}^{D \times D}$ is some positive-definite symmetric matrix. Additionally, let

$$Q'_t(\mathbf{x}_{t-2:t}; \mathbf{u}_{1:T}) = M'_t(\mathbf{x}_t | \mathbf{x}_{t-1}; \mathbf{u}_{1:T}) G'_t(\mathbf{x}_{t-2:t}; \mathbf{u}_{1:T}),$$

for some mutation kernel $M'_t(\mathbf{x}_t | \mathbf{x}_{t-1}; \mathbf{u}_{1:T})$ and some potential function $G'_t(\mathbf{x}_{t-2:t}; \mathbf{u}_{1:T})$ (both of which may depend on some or all of $\mathbf{u}_1, \ldots, \mathbf{u}_t$) such that

$$\pi'_T(\mathbf{x}_{1:T} | \mathbf{u}_{1:T}) \propto \prod_{t=1}^{T} Q'_t(\mathbf{x}_{t-2:t}; \mathbf{u}_{1:T}).$$

Algorithm 33 (generic auxiliary algorithm) *Given* $\mathbf{x}_{1:T} \in \mathcal{X}^T$*, sample*

$$\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \Phi_t(\mathbf{x}_{1:T}), \mathbf{E}_t(\mathbf{x}_{t-1:t})),$$

for any $t = 1, \dots, T$ *and then*

1. *for* $t = 1, \dots, T$*,*
   (a) *sample* k_t *from a uniform distribution on* $[N]_0$ *and set* $\mathbf{x}_t^{k_t} := \mathbf{x}_t$*,*
   (b) *if* $t > 1$*, set* $a_{t-1}^{k_t} := k_{t-1}$ *and sample* $a_{t-1}^n = i$ *w.p.* W_{t-1}^i*, for* $n \in [N]_0 \setminus \{k_t\}$*,*
   (c) *sample* $\mathbf{x}_t^n \sim M_t'(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n}; \mathbf{u}_{1:T})$ *for* $n \in [N]_0 \setminus \{k_t\}$*,*
   (d) *for* $n \in [N]_0$*, set* $w_t^n \propto G_t'(\mathbf{x}_{t-2:t}^{(n)}; \mathbf{u}_{1:T})$*,*
   (e) *for* $n \in [N]_0$*, set* $W_t^n := w_t^n / \sum_{m=0}^N w_t^m$*;*
2. *sample* $i \in [N]_0 \setminus \{k_T\}$ *w.p.* $\dfrac{W_T^i}{1 - W_T^{k_T}}$*; set* $l_T := i$ *w.p.* $1 \wedge \dfrac{1 - W_T^{k_T}}{1 - W_T^i}$*; otherwise, set* $l_T := k_T$*;*
3. *for* $t = T-1, \dots, 1$*, sample* $l_t = i \in [N]_0$ *w.p.*
$$\frac{W_t^i Q_{t+1}'((\mathbf{x}_{t-1:t}^{(i)}, \mathbf{x}_{t+1}^{l_{t+1}}); \mathbf{u}_{1:T}) Q_{t+2}'((\mathbf{x}_t^i, \mathbf{x}_{t+1}^{l_{t+1}}, \mathbf{x}_{t+2}^{l_{t+2}}); \mathbf{u}_{1:T})}{\sum_{n=0}^N W_t^n Q_{t+1}'((\mathbf{x}_{t-1:t}^{(n)}, \mathbf{x}_{t+1}^{l_{t+1}}); \mathbf{u}_{1:T}) Q_{t+2}'((\mathbf{x}_t^n, \mathbf{x}_{t+1}^{l_{t+1}}, \mathbf{x}_{t+2}^{l_{t+2}}); \mathbf{u}_{1:T})};$$
4. *return* $\tilde{\mathbf{x}}_{1:T} := (\mathbf{x}_1^{l_1}, \dots, \mathbf{x}_t^{l_T})$*.*

Proposition 34 (validity of the generic auxiliary algorithm) *Sampling* $\tilde{\mathbf{x}}_{1:T}$ *given* $\mathbf{x}_{1:T}$ *via Algorithm 33 induces a Markov kernel* $P(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ *which leaves* π_T *invariant.*

Proof (of Proposition 34) The extended distribution from (32) admits $\pi_T(\mathbf{x}_{1:T})$ as a marginal. Therefore, a valid MCMC update for sampling from this extended distribution is given by alternating the following two steps. Given $\mathbf{x}_{1:T} \in \mathcal{X}^T$,

1. sample $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \Phi_t(\mathbf{x}_{1:T}), \mathbf{E}_t(\mathbf{x}_{t-1:t}))$, for $t = 1, \dots, T$;
2. run a standard CSMC algorithm with backward sampling (as in Algorithm 1) targeting $\pi_T'(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})$ but with $M_t(\mathbf{x}_t|\mathbf{x}_{t-1})$, $G_t(\mathbf{x}_{t-1:t})$, and $Q_t(\mathbf{x}_{t-1:t})$ replaced by $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1}; \mathbf{u}_{1:T})$, $G_t'(\mathbf{x}_{t-2:t}; \mathbf{u}_{1:T})$ and $Q_t'(\mathbf{x}_{t-2:t}; \mathbf{u}_{1:T})$, and with appropriate adjustments (e.g., of the backward kernels) to account for the possibility that the model may only be second-order Markov.

These to steps are equivalent to Algorithm 33. □

D.2 Generic marginal algorithm

Consider the same setting as above but now assume that for any $t \in [T]$, $\boldsymbol{\psi}_t \equiv \mathbf{0}$, so that $\Phi_t(\mathbf{x}_{1:T}) = \boldsymbol{\phi}_t(\mathbf{x}_{t-1:t})$ as well as that $\mathbf{E}_t(\mathbf{x}_{t-1:t}) = \mathbf{E}_t$ is independent of $\mathbf{x}_{t-1:t}$.

Furthermore, assume that $M'_t(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_{1:T}) = M'_t(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_t)$ only depends on the tth auxiliary variable $\mathbf{u}_t$ and, specifically, is a Gaussian distribution of the following form:

$$M'_t(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_t) \coloneqq \mathrm{N}(\mathbf{x}_t;\mathbf{v}_t(\mathbf{x}_{t-1}) + \mathbf{H}_t\mathbf{u}_t, \mathbf{D}_t),$$

where $\mathbf{v}_t(\mathbf{x}) \in \mathcal{X}$ whilst $\mathbf{H}_t, \mathbf{D}_t \in \mathbb{R}^{D\times D}$ do not depend on $\mathbf{x}_{t-1}$ and define

$$q_t^{-n}(\mathbf{x}_t^{-n}, \mathbf{u}_t|\mathbf{x}_t^n;\mathcal{H}_{t-1}) \coloneqq \mathrm{N}(\mathbf{u}_t;\mathbf{x}_t^n + \boldsymbol{\phi}_t(\mathbf{x}_{t-1:t}^{(n)}), \mathbf{E}_t)\prod_{\substack{m=0\\m\neq n}}^{N} M'_t(\mathbf{x}_t^m|\mathbf{x}_{t-1}^{a_{t-1}^m};\mathbf{u}_t),$$

where $\mathcal{H}_{t-1}$ is the history of the particle system up to time $t-1$, i.e., all particles and ancestor indices with 'time' subscript $s \leq t-1$. By Lemma 31 from Appendix C, we obtain a closed-form expression for

$$q_t^{-n}(\mathbf{x}_t^{-n}|\mathbf{x}_t^n;\mathcal{H}_{t-1}) \coloneqq \int_{\mathcal{X}} q_t^{-n}(\mathbf{x}_t^{-n}, \mathbf{u}_t|\mathbf{x}_t^n;\mathcal{H}_{t-1})\,\mathrm{d}\mathbf{u}_t.$$

Algorithm 35 (generic marginal algorithm) *Given $\mathbf{x}_{1:T} \in \mathcal{X}^T$:*

```
1. for $t = 1, \dots, T$,
    (a) sample $k_t$ from a uniform distribution on $[N]_0$ and set $\mathbf{x}_t^{k_t} \coloneqq \mathbf{x}_t$,
    (b) if $t > 1$, set $a_{t-1}^{k_t} \coloneqq k_{t-1}$ and sample $a_{t-1}^n = i$ w.p. $W_{t-1}^i$, for $n \in [N]_0 \setminus \{k_t\}$,
    (c) sample $\mathbf{x}_t^{-k_t} \sim q_t^{-k_t}(\mathbf{x}_t^{-k_t}|\mathbf{x}_t^{k_t};\mathcal{H}_{t-1})$
        (e.g. by sampling $\mathbf{u}_t \sim \mathrm{N}(\mathbf{x}_t + \boldsymbol{\phi}_t(\mathbf{x}_{t-1:t}), \mathbf{E}_t)$ and then $\mathbf{x}_t^n \sim M'_t(\,\cdot\,|\mathbf{x}_{t-1}^{a_{t-1}^n};\mathbf{u}_t)$ for $n \in [N]_0 \setminus \{k_t\}$),
    (d) for $n \in [N]_0$, set $w_t^n \propto Q_t(\mathbf{x}_{t-1:t}^{(n)})q_t^{-n}(\mathbf{x}_t^{-n}|\mathbf{x}_t^n;\mathcal{H}_{t-1})$,
    (e) for $n \in [N]_0$, set $W_t^n \coloneqq w_t^n / \sum_{m=0}^N w_t^m$;
2. sample $i \in [N]_0 \setminus \{k_T\}$ w.p. $\dfrac{W_T^i}{1 - W_T^{k_T}}$; set $l_T \coloneqq i$ w.p. $1 \wedge \dfrac{1 - W_T^{k_T}}{1 - W_T^i}$; otherwise, set $l_T \coloneqq k_T$;
3. for $t = T-1, \dots, 1$, sample $l_t = i \in [N]_0$ w.p. $\dfrac{W_t^i Q_{t+1}(\mathbf{x}_t^i, \mathbf{x}_{t+1}^{l_{t+1}})}{\sum_{n=0}^N W_t^n Q_{t+1}(\mathbf{x}_t^n, \mathbf{x}_{t+1}^{l_{t+1}})}$;
4. return $\tilde{\mathbf{x}}_{1:T} \coloneqq (\mathbf{x}_1^{l_1}, \dots, \mathbf{x}_t^{l_T})$.
```

Algorithm 35 can be implemented in $\mathrm{O}(N)$ operations because Lemma 31 from Appendix C allows us to write the weight in Step 1d as

$$\begin{aligned} w_t^n &\propto Q_t(\mathbf{x}_{t-1:t}^{(n)})q_t^{-n}(\mathbf{x}_t^{-n}|\mathbf{x}_t^n;\mathcal{H}_{t-1}) \\ &\propto Q_t(\mathbf{x}_{t-1:t}^{(n)})H_{t,\boldsymbol{\phi}_t(\mathbf{x}_{t-1:t}^{(n)})}(\mathbf{x}_t^n, \mathbf{v}_t^n, \bar{\mathbf{x}}_t, \bar{\mathbf{v}}_t), \end{aligned}$$

where $\mathbf{v}_t^n \coloneqq \mathbf{v}_t(\mathbf{x}_{t-1}^{a_{t-1}^n})$, $\bar{\mathbf{x}}_t \coloneqq \frac{1}{N+1}\sum_{m=0}^N \mathbf{x}_t^m$, $\bar{\mathbf{v}}_t \coloneqq \frac{1}{N+1}\sum_{m=0}^N \mathbf{v}_t^m$ and

$$\begin{aligned}\log H_{t,\phi}(\mathbf{x}, \mathbf{v}, \bar{\mathbf{x}}, \bar{\mathbf{v}}) = \tfrac{1}{2}(\mathbf{x}-\mathbf{v})^{\mathrm{T}}(\mathbf{D}_t^{-1}+\mathbf{G}_t)(\mathbf{x}-\mathbf{v})\\ - \tfrac{1}{2}N(\mathbf{x}+\phi)^{\mathrm{T}}\mathbf{H}_t^{\mathrm{T}}(\mathbf{D}_t^{-1}-N\mathbf{G}_t)\mathbf{H}_t(\mathbf{x}+\phi)\\ -(N+1)(\bar{\mathbf{x}}-\bar{\mathbf{v}})^{\mathrm{T}}[\mathbf{G}_t(\mathbf{x}-\mathbf{v})-(\mathbf{D}_t^{-1}-N\mathbf{G}_t)\mathbf{H}_t(\mathbf{x}+\phi)]\\ -(\mathbf{x}-\mathbf{v})^{\mathrm{T}}(\mathbf{D}_t^{-1}-N\mathbf{G}_t)\mathbf{H}_t(\mathbf{x}+\phi),\end{aligned}$$

with $\mathbf{G}_t \coloneqq (\mathbf{D}+N\mathbf{H}_t\mathbf{E}_t\mathbf{H}_t^{\mathrm{T}})^{-1}\mathbf{H}_t\mathbf{E}_t\mathbf{H}_t^{\mathrm{T}}\mathbf{D}_t^{-1}$ (see (31) for an alternative expression).

Proposition 36 (validity of the generic marginal algorithm) *Sampling $\tilde{\mathbf{x}}_{1:T}$ given $\mathbf{x}_{1:T}$ via Algorithm 35 induces a Markov kernel $P(\tilde{\mathbf{x}}_{1:T}|\mathbf{x}_{1:T})$ which leaves π_T invariant.*

Proof (of Proposition 36) We begin with a few observations.

1. Since the unnormalised weights satisfy

$$w_t^n \propto Q_t(\mathbf{x}_{t-1:t}^{(n)})q_t^{-n}(\mathbf{x}_t^{-n}|\mathbf{x}_t^n;\mathcal{H}_{t-1}), \tag{33}$$

   we have that

$$q_t^{-k_t}(\mathbf{x}_t^{-k_t}|\mathbf{x}_t^{k_t};\mathcal{H}_{t-1}) = \frac{w_t^{k_t}}{w_t^{l_t}}\frac{Q_t(\mathbf{x}_{t-1:t}^{(l_t)})}{Q_t(\mathbf{x}_{t-1:t}^{(k_t)})}q_t^{-l_t}(\mathbf{x}_t^{-l_t}|\mathbf{x}_t^{l_t};\mathcal{H}_{t-1}).$$

2. For a given set of final-time weights $\{W_T^n\}_{n\in[N]_0}$, let $R_T(\,\cdot\,|\,\cdot\,;\mathcal{H}_T)$ be the $\sum_{n=0}^N W_T^n\delta_n$-invariant Markov kernel used in Step 2 of Algorithm 35. That is, sampling $l_T \sim R_T(\,\cdot\,|k_T;\mathcal{H}_T)$ could be the forced-move update; or, in the more common specification of CSMC algorithms (Andrieu et al., 2010), i.e. without the forced-move update, we would simply have $R_T(l_T|k_T;\mathcal{H}_T) = W_T^{l_T}$. In either case, it can then be verified that

$$W_T^{k_T}R_T(l_T|k_T;\mathcal{H}_T) = W_T^{l_T}R_T(k_T|l_T;\mathcal{H}_T),$$

   for any $k_T, l_T \in [N]_0$.

3. Under Algorithm 35, we have the following identities (with probability 1): $\mathbf{x}_t = \mathbf{x}_t^{k_t}$ and $\mathbf{x}_t' = \mathbf{x}_t^{l_t}$, for $1 \le t \le T$, as well as $a_{t-1}^{k_t} = k_{t-1}$, for any $1 < t \le T$.

Putting these observations together then shows that the Algorithm 35 targets the following extended distribution (i.e., this is the distribution of all random variables obtained if we first sampled $\mathbf{x}_{1:T} \sim \pi_T$ and then ran Algorithm 35):

$$
\begin{aligned}
&\frac{\pi_T(\mathbf{x}_{1:T})}{(N+1)^T} \delta_{\mathbf{x}_{1:T}}(\mathbf{x}_{1:T}^{k_{1:T}}) \Bigg[\prod_{t=1}^{T} q_t^{-k_t}(\mathbf{x}_t^{-k_t} | \mathbf{x}_t^{k_t}; \mathcal{H}_{t-1})\Bigg] \Bigg[\prod_{t=2}^{T} \delta_{k_{t-1}}(a_{t-1}^{k_t}) \prod_{\substack{n=0 \\ n \neq k_t}}^{N} W_{t-1}^{a_{t-1}^n}\Bigg] \\
&\times R_T(l_T | k_T; \mathcal{H}_T) \Bigg[\prod_{t=1}^{T-1} \frac{w_t^{l_t} Q_{t+1}(\mathbf{x}_t^{l_t}, \mathbf{x}_{t+1}^{l_{t+1}})}{\sum_{m=0}^{N} w_t^m Q_{t+1}(\mathbf{x}_t^m, \mathbf{x}_{t+1}^{l_{t+1}})}\Bigg] \delta_{\mathbf{x}_{1:T}^{l_{1:T}}}(\tilde{\mathbf{x}}_{1:T}) \\
= &\frac{\pi_T(\tilde{\mathbf{x}}_{1:T})}{(N+1)^T} \delta_{\tilde{\mathbf{x}}_{1:T}}(\mathbf{x}_{1:T}^{l_{1:T}}) \Bigg[\prod_{t=1}^{T} q_t^{-l_t}(\mathbf{x}_t^{-l_t} | \mathbf{x}_t^{l_t}; \mathcal{H}_{t-1})\Bigg] \\
&\times \Bigg[\prod_{t=2}^{T} \frac{w_{t-1}^{a_{t-1}^{l_t}} Q_t(\mathbf{x}_{t-1}^{a_{t-1}^{l_t}}, \mathbf{x}_t^{l_{t+1}})}{\sum_{m=0}^{N} w_{t-1}^m Q_t(\mathbf{x}_{t-1}^m, \mathbf{x}_t^{l_t})} \prod_{\substack{n=0 \\ n \neq l_t}}^{N} W_{t-1}^{a_{t-1}^n}\Bigg] \\
&\times R_T(k_T | l_T; \mathcal{H}_T) \Bigg[\prod_{t=1}^{T-1} \delta_{a_{t-1}^{k_t}}(k_{t-1})\Bigg] \delta_{\mathbf{x}_{1:T}^{k_{1:T}}}(\mathbf{x}_{1:T}),
\end{aligned}
$$

where the r.h.s. is the distribution obtained if we first sampled $\tilde{\mathbf{x}}_{1:T} \sim \pi_T$ and then ran Algorithm 35 algorithm but with ancestor sampling (Lindsten et al., 2012) instead of backward sampling. This is a modified version of the proof technique from Finke et al. (2016). In other words, if $\mathbf{x}_{1:T} \sim \pi_T$ and if $\tilde{\mathbf{x}}_{1:T}$ is sampled via Algorithm 35, then $\tilde{\mathbf{x}}_{1:T} \sim \pi_T$. This completes the proof. □

D.3 Invariance of the algorithms

We can now easily verify the validity of the 'auxiliary' algorithms (Particle-aMALA, Particle-aMALA+, Particle-aGRAD, Particle-aGRAD+, Particle-aPCNL, Particle-aPCNL+, and twisted Particle-aGRAD/Particle-aGRAD+/Particle-aPCNL/Particle-aPCNL+) by noting that these are special cases of Algorithm 33, and the validity of the 'marginal' algorithms (Particle-MALA, Particle-mGRAD, Particle-PCNL) by noting that these are special cases of Algorithm 35.

Proof (of Proposition 4) This follows by taking $\boldsymbol{\phi}_t(\mathbf{x}_{t-1:t}) \coloneqq \kappa \frac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log Q_t(\mathbf{x}_{t-1:t})$, $\boldsymbol{\psi}_t \equiv \mathbf{0}$, $M_t'(\mathbf{x}_t | \mathbf{x}_{t-1}; \mathbf{u}_{1:T}) = \mathrm{N}(\mathbf{x}_t; \mathbf{u}_t, \frac{\delta_t}{2}\mathbf{I})$ and $\mathbf{E}_t \equiv \frac{\delta_t}{2}\mathbf{I}$ in Proposition 34. □

Proof (of Proposition 7) This follows from Proposition 36 with the same setting as in Proposition 4. In particular, in this case, $\mathbf{v}_t \equiv \mathbf{0}$, $\mathbf{H}_t = \mathbf{I}$, $\mathbf{D}_t = \mathbf{E}_t = \frac{\delta_t}{2}\mathbf{I}$. Consequently, (33) then simplifies to (7), where we have used that $\mathbf{G}_t = [\frac{\delta_t}{2}(N+1)]^{-1}\mathbf{I} = \mathbf{D}_t^{-1}/(N+1)$ and $\mathbf{D}_t^{-1} - N\mathbf{G}_t = \mathbf{G}_t$. □

Proof (of Proposition 9) This follows in the same way as the proof of Proposition 4 except that now $\boldsymbol{\psi}_t(\mathbf{x}_{t:t+1}) = \kappa \frac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log Q_{t+1}(\mathbf{x}_{t:t+1})$. □

Proof (of Proposition 11) This follows in the same way as the proof of Proposition 4 except that now $\boldsymbol{\phi}_t(\mathbf{x}_{t-1:t}) \coloneqq \kappa \frac{\delta_t}{2} \nabla_{\mathbf{x}_t} \log G_t(\mathbf{x}_{t-1:t})$, and $M_t'(\mathbf{x}_t | \mathbf{x}_{t-1}; \mathbf{u}_{1:T}) = \mathrm{N}(\mathbf{x}_t; \mathbf{m}_t'(\mathbf{x}_{t-1}, \mathbf{u}_t), \mathbf{C}_t'(\mathbf{x}_{t-1}))$, where $\mathbf{m}_t'(\mathbf{x}_{t-1}, \mathbf{u}_t)$ and $\mathbf{C}_t'(\mathbf{x}_{t-1})$ are defined in (10) and (11). □

Proof (of Proposition 14) This follows from Proposition 36 with the same setting as in Proposition 11. In particular, in this case, $\mathbf{H}_t = \mathbf{A}_t := (\mathbf{C}_t + \frac{2}{\delta_t}\mathbf{I})^{-1}\mathbf{C}_t$, $\mathbf{D}_t = \frac{\delta_t}{2}\mathbf{A}_t$ and $\mathbf{E}_t = \frac{\delta_t}{2}\mathbf{I}$. Consequently, (33) then simplifies to (14), where we have used that $\mathbf{A}_t$ is symmetric, that $\mathbf{H}_t^{\mathrm{T}}\mathbf{D}_t^{-1} = \mathbf{D}_t^{-1}\mathbf{H}_t = \frac{2}{\delta_t}\mathbf{I}$ and hence

$$\mathbf{D}_t^{-1} - N\mathbf{G}_t = \mathbf{A}_t^{-1}\mathbf{G}_t = \mathbf{G}_t\mathbf{A}_t^{-1}.$$

This completes the proof. □

Proof (of Proposition 16) This follows in the same way as the proof of Proposition 11 except that now $\boldsymbol{\psi}_t(\mathbf{x}_{t:t+1}) = \kappa\frac{\delta_t}{2}\nabla_{\mathbf{x}_t}\log G_{t+1}(\mathbf{x}_{t:t+1})$. □

Proof (of Proposition 17) This follows in the same way as the proof of Propositions 11 and 16, respectively, but with $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_{1:T}) = \mathrm{N}(\mathbf{x}_t;\mathbf{F}_t'\mathbf{x}_{t-1} + \mathbf{b}_t', \mathbf{C}_t')$. □

Proof (of Proposition 21) This follows in the same way as the proof of Proposition 4 except that now $\boldsymbol{\phi}_t(\mathbf{x}_{t-1:t}) := \kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_{t-1})\nabla_{\mathbf{x}_t}\log G_t(\mathbf{x}_{t-1:t})$, $\mathbf{E}_t(\mathbf{x}_{t-1:t}) := \frac{\delta_t}{2}\mathbf{C}_t(\mathbf{x}_{t-1})$ and $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_{1:T}) = \mathrm{N}(\mathbf{x}_t;\mathbf{m}_t'(\mathbf{x}_{t-1},\mathbf{u}_t),\mathbf{C}_t'(\mathbf{x}_{t-1}))$, where $\mathbf{m}_t'(\mathbf{x}_{t-1},\mathbf{u}_t)$ and $\mathbf{C}_t'(\mathbf{x}_{t-1})$ are defined in (19) and (20). □

Proof (of Proposition 23) This follows from Proposition 36 with the same setting as in Proposition 21. In particular, in this case, $\mathbf{H}_t = \beta_t\mathbf{I}$, $\mathbf{D}_t = (1-\beta_t)\mathbf{C}_t$ and $\mathbf{E}_t = \frac{\delta_t}{2}\mathbf{C}_t$. Consequently, (33) then simplifies to (22), where we have used that $\mathbf{E}\mathbf{H}_t^{\mathrm{T}}\mathbf{D}_t^{-1} = \mathbf{D}_t^{-1}\mathbf{H}_t\mathbf{E} = \mathbf{I}$ and hence

$$\begin{aligned}\mathbf{D}_t^{-1} + \mathbf{G}_t &= (\beta_t^{-1} + N + 1)\mathbf{G}_t,\\ \mathbf{D}_t^{-1} - N\mathbf{G}_t &= \beta_t^{-1}\mathbf{G}_t.\end{aligned}$$

This completes the proof. □

Proof (of Proposition 25) This follows in the same way as the proof of Proposition 21 except that now $\boldsymbol{\psi}_t(\mathbf{x}_{t:t+1}) = \kappa\frac{\delta_t}{2}\widetilde{\mathbf{C}}_t(\mathbf{x}_t)\nabla_{\mathbf{x}_t}\log G_{t+1}(\mathbf{x}_{t:t+1})$. □

Proof (of Proposition 26) This follows in the same way as the proof of Propositions 21 and 25, respectively, but with $M_t'(\mathbf{x}_t|\mathbf{x}_{t-1};\mathbf{u}_{1:T}) = \mathrm{N}(\mathbf{x}_t;\mathbf{F}_t'\mathbf{x}_{t-1} + \mathbf{b}_t', \mathbf{C}_t')$. □

Appendix E. Proof of Propositions 18 and 19

E.1 Preliminaries

For some given $N \in \mathbb{N}$, let Ψ^n denote either the *Boltzmann selection function* (with the convention $h^0 := 0$):

$$\Psi^n(h^{1:N}) := \frac{\exp(h^n)}{1 + \sum_{m=0}^{N}\exp(h^m)},$$

or the *Rosenbluth–Teller selection function*:

$$\Psi^n(h^{1:N}) := \begin{cases} \dfrac{\exp(h^n)}{1 + \sum_{m=1}^N \exp(h^m) - 1 \wedge \exp(h^n)}, & \text{if } n > 0, \\ 1 - \sum_{l=1}^N \Psi^l(h^{1:N}), & \text{if } n = 0. \end{cases}$$

In either case, Ψ^n is Lipschitz continuous with constant denoted $[\Psi^n]_{\mathrm{LIP}}$.

E.2 Marginal MCMC kernels in the special case: $T = 1$

For the moment, we assume that $T = 1$. To simplify the notation, we drop the 'time' subscripts $t = 1$. With this convention, for some bounded and differentiable $G : \mathbb{R}^D \to (0, \infty)$, define

$$\pi(\mathbf{x}) \propto \mathrm{N}(\mathbf{x}; \mathbf{m}, \mathbf{C}) G(\mathbf{x}).$$

The π-invariant Markov kernels induced by the (non-auxiliary variable based) algorithms discussed in this work can then be written as

$$P_a(\mathrm{d}\tilde{\mathbf{x}}|\mathbf{x}) = \sum_{l=0}^N \int_{\mathcal{X}^{N+2}} \delta_{\mathbf{x}}(\,\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0}|\mathbf{x}^0) \Psi^l(\{h_a^n(\mathbf{x}^{0:N})\}_{n=1}^N) \delta_{\mathbf{x}^l}(\mathrm{d}\tilde{\mathbf{x}}),$$

where have appealed to symmetry to always place the reference 'path' in position 0, and with

$$\begin{aligned} h_a^n(\mathbf{x}^{0:N}) &:= \log q_a^{-n}(\mathbf{x}^{-n}|\mathbf{x}^n) - \log q_a^{-0}(\mathbf{x}^{-0}|\mathbf{x}^0), \\ q_a^{-n}(\mathbf{x}^{-n}|\mathbf{x}^n) &= \mathrm{N}(\mathbf{x}^{-n}; \mathbf{m}_a(\mathbf{x}^n), \mathbf{C}_a), \end{aligned}$$

where $\mathbf{m}_a(\mathbf{x}^n) \in \mathbb{R}^{ND}$ is a suitable mean vector (which may depend on $\mathbf{x}^n \in \mathbb{R}^D$), $\mathbf{C}_a \in \mathbb{R}^{(ND)\times(ND)}$ a suitable variance, and where we again slightly abuse notation to let $\mathbf{x}^{-n}$ represent both the tuple $(\mathbf{x}^0, \dots, \mathbf{x}^{n-1}, \mathbf{x}^{n+1}, \dots, \mathbf{x}^N)$ and its vectorised form

$$\mathbf{x}^{-n} := \mathrm{vec}(\mathbf{x}^{-n}) = \begin{bmatrix} \mathbf{x}^0 \\ \vdots \\ \mathbf{x}^{n-1} \\ \mathbf{x}^{n+1} \\ \vdots \\ \mathbf{x}^N \end{bmatrix} \in \mathbb{R}^{ND}.$$

Additionally, 'a' is a placeholder for 'CSMC', 'Particle-MALA', or 'Particle-mGRAD'. Specifically, by the developments from Section C (Lemma 31 and its proof), and recalling that the block matrix operator $\mathcal{M}_N$ was defined in (27),

$$\begin{aligned}
\mathbf{m}_{\text{Particle-mGRAD}}(\mathbf{x}^n) &= \mathbf{1}_{N\times 1} \otimes [\mathbf{m} + \mathbf{A}(\mathbf{x}^n + \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m})],\\
\mathbf{C}_{\text{Particle-mGRAD}} &= \tfrac{\delta}{2}\mathcal{M}_N(\mathbf{A}, \mathbf{A}^2) = \tfrac{\delta}{2}[\mathbf{I}_N \otimes \mathbf{A} + \mathbf{1}_{N\times N} \otimes \mathbf{A}^2],\\
\mathbf{m}_{\text{CSMC}}(\mathbf{x}^n) &= \mathbf{1}_{N\times 1} \otimes \mathbf{m},\\
\mathbf{C}_{\text{CSMC}} &= \mathcal{M}_N(\mathbf{C}, \mathbf{0}_{D\times D}) = \mathbf{I}_N \otimes \mathbf{C},\\
\mathbf{m}_{\text{Particle-MALA}}(\mathbf{x}^n) &= \mathbf{1}_{N\times 1} \otimes [\mathbf{x}^n + \boldsymbol{\phi}(\mathbf{x}^n) + \boldsymbol{\varphi}(\mathbf{x}^n)],\\
\mathbf{C}_{\text{Particle-MALA}} &= \tfrac{\delta}{2}\mathcal{M}_N(\mathbf{I}, \mathbf{I}) = \tfrac{\delta}{2}[\mathbf{I}_{ND} + \mathbf{1}_{N\times N} \otimes \mathbf{I}],
\end{aligned}$$

where $\boldsymbol{\phi}(\mathbf{x}) \coloneqq \kappa\frac{\delta}{2}\nabla \log G(\mathbf{x})$ and $\boldsymbol{\varphi}(\mathbf{x}) \coloneqq \kappa\frac{\delta}{2}\nabla \log M(\mathbf{x})$ and with $\mathbf{A} \coloneqq (\frac{\delta}{2}\mathbf{C}^{-1} + \mathbf{I})^{-1} = \mathbf{C}(\mathbf{C} + \frac{\delta}{2}\mathbf{I})^{-1} = (\mathbf{C} + \frac{\delta}{2}\mathbf{I})^{-1}\mathbf{C}$.

Key to our proofs will be the following bound which follows from the triangle inequality and a telescoping-sum decomposition (here: a and b are again placeholders which take values in $\{\text{CSMC}, \text{Particle-MALA}, \text{Particle-mGRAD}\}$):

$$\begin{aligned}
&\|P_a(\,\cdot\,|\mathbf{x}) - P_b(\,\cdot\,|\mathbf{x})\|_{\text{TV}}\\
&\quad\leq \|q_a^{-0}(\,\cdot\,|\mathbf{x}) - q_b^{-0}(\,\cdot\,|\mathbf{x})\|_{\text{TV}}\\
&\qquad + \sum_{l=0}^{N} \int_{\mathcal{X}^{N+1}} \delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0}|\mathbf{x}^0) |\Psi^l(\{h_a^n(\mathbf{x}^{0:N})\}_{n=1}^N) - \Psi^l(\{h_b^n(\mathbf{x}^{0:N})\}_{n=1}^N)|\\
&\quad\leq \sqrt{\mathrm{KL}(q_a^{-0}(\,\cdot\,|\mathbf{x}) \| q_b^{-0}(\,\cdot\,|\mathbf{x}))}\\
&\qquad + \sum_{l=0}^{N} [\Psi^l]_{\text{LIP}} \int_{\mathcal{X}^{N+1}} \delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0}|\mathbf{x}^0) \sum_{n=1}^{N} |h_a^n(\mathbf{x}^{0:N}) - h_b^n(\mathbf{x}^{0:N})|\\
&\quad\leq C\Big[\sqrt{D_{a,b}^0(\mathbf{x})} + \sum_{n=0}^{N} D_{a,b}^n(\mathbf{x})\Big]. \qquad (34)
\end{aligned}$$

Here, the penultimate line follows from Pinsker's inequality and the Lipschitz continuity of the selection function; $C \geq 0$ is some constant which may depend on these Lipschitz constants and N and D; for the last inequality, we have defined

$$D_{a,b}^n(\mathbf{x}) \coloneqq \int_{\mathcal{X}^{N+1}} \delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0}|\mathbf{x}^0) |\log q_a^{-n}(\mathbf{x}^{-n}|\mathbf{x}^n) - \log q_b^{-n}(\mathbf{x}^{-n}|\mathbf{x}^n)|. \qquad (35)$$

E.3 Proofs of Part 1

Proof (of Part 1 of Proposition 18) By Assumption **A1**, the model factorises over time and so do the CSMC and Particle-mGRAD algorithms. Hence, without loss of generality, we prove the result in the case that $T = 1$ (and we drop the 'time' subscript $t = 1$ hereafter). Throughout the proof, we will also make repeated use of the fact that the eigenvalues of $\mathbf{A}_k$ are given by $(2\lambda_{k,d})/(2\lambda_{k,d} + \delta)$, for $d \in [D]$.

For $\varepsilon \geq 1$ the result is trivially true but meaningless. Fix $\varepsilon \in (0, 1)$.

$$F_k \coloneqq \{\mathbf{x} \in \mathbb{R}^D \mid \|\mathbf{x} - \mathbf{m}\|_2 \leq \lambda_k^{(1-\varepsilon)/2}\},$$

denote a ball of radius $\lambda_k^{(1-\varepsilon)/2}$ around $\mathbf{m}$, for any $k \geq 1$. We then have $\pi_k(F_k) = (1+H_k)^{-1}$, where, letting $F_k^{\mathrm{c}} := \mathcal{X} \setminus F_k$:

$$\begin{aligned}
H_k &:= \frac{\int_{F_k^{\mathrm{c}}} G(\mathbf{x})\, \mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{m}, \mathbf{C}_k)}{\int_{F_k} G(\mathbf{x})\, \mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{m}, \mathbf{C}_k)} \\
&\leq \frac{\sup_{x \in \mathcal{X}} G(\mathbf{x})}{\inf_{\mathbf{x} \in F_k} G(\mathbf{x})} \frac{\int_{F_k^{\mathrm{c}}} \mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{m}, \mathbf{C}_k)}{\int_{F_k} \mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{m}, \mathbf{C}_k)} \\
&\leq \frac{\sup_{x \in \mathcal{X}} G(\mathbf{x})}{\inf_{\mathbf{x} \in F_k} G(\mathbf{x})} \frac{\int_{\mathcal{X}} \mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{0}, \mathbf{I})\, \mathbb{I}\{\|\mathbf{x}\|_2 > \lambda_k^{-\varepsilon/2}\}}{\int_{\mathcal{X}} \mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{0}, \mathbf{I})\, \mathbb{I}\{\|\mathbf{x}\|_2 \leq \lambda_k^{-\varepsilon/2}\}} \\
&\to 0,
\end{aligned}$$

as $k \to \infty$, where we have used that G is bounded and that $(\inf_{\mathbf{x} \in F_k} G(\mathbf{x}))_{k \geq 1}$ is an increasing sequence in $(0, \infty)$ (since $(F_k)_{k \geq 1}$ is decreasing and F_1 is compact).

By the decomposition from (34), all that remains is to control the terms

$$\sup_{\mathbf{x}^0 \in F_k} D^n_{\text{CSMC},\text{Particle-mGRAD},k}(\mathbf{x}^0),$$

for arbitrary $n \in [N]_0$.

Firstly, by Lemma 29 from Appendix C, letting $\lambda(\mathbf{C}_k) = \{\lambda_{k,1}, \dots, \lambda_{k,D}\}$ denote the eigenvalues of $\mathbf{C}_k$ and noting that $\mathbf{A}_k$ is simultaneously diagonalisable with $\mathbf{A}_k^2$:

$$\begin{aligned}
&|\log(\det(\mathbf{C}_{\text{CSMC},k})) - \log(\det(\mathbf{C}_{\text{Particle-mGRAD},k}))| \\
&\quad = \Bigl|\sum_{d=1}^{D} N \log \lambda_{k,d} - (N-1)\log\Bigl(\frac{\delta\lambda_{k,d}}{2\lambda_{k,d}+\delta}\Bigr) - \log\left(\frac{\delta\lambda_{k,d}}{2\lambda_{k,d}+\delta} + \frac{2\delta N \lambda_{k,d}^2}{(2\lambda_{k,d}+\delta)^2}\right)\Bigr| \\
&\quad = \Bigl|\sum_{d=1}^{D} N \log\Bigl(\frac{2\lambda_{k,d}+\delta}{\delta}\Bigr) + \log\Bigl(\frac{\delta\lambda_{k,d}}{2\lambda_{k,d}+\delta}\Bigr) - \log\left(\frac{\delta\lambda_{k,d}}{2\lambda_{k,d}+\delta} + \frac{2\delta N \lambda_{k,d}^2}{(2\lambda_{k,d}+\delta)^2}\right)\Bigr| \\
&\quad = \Bigl|\sum_{d=1}^{D} N \log\Bigl(\frac{2\lambda_{k,d}+\delta}{\delta}\Bigr) + \log\Bigl(\frac{2\lambda_{k,d}+\delta}{2\lambda_{k,d}(N+1)+\delta}\Bigr)\Bigr| \in \mathrm{O}(\lambda_k).
\end{aligned} \tag{36}$$

Secondly, by Lemma 30 from Appendix C,

$$\mathbf{C}^{-1}_{\text{Particle-mGRAD},k} = \frac{2}{\delta}\mathcal{M}_N(\mathbf{A}_k^{-1}, -(\mathbf{I} + N\mathbf{A}_k)^{-1}),$$

and with the conventions that the sum symbol $\sum_i$ is shorthand for $\sum_{i \in [N]_0 \setminus \{n\}}$, that $\sum_j$ is shorthand for $\sum_{j \in [N]_0 \setminus \{n\}}$, that $\sum_{i \neq j}$ is shorthand for $\sum_{j \in [N]_0 \setminus \{n,i\}}$, and again writing

$\boldsymbol{\phi}(\mathbf{x}) = \kappa \frac{\delta}{2} \nabla \log G(\mathbf{x})$ we obtain:

$$
\begin{aligned}
&\big|(\mathbf{x}^{-n} - \mathbf{m}_{\text{CSMC}}(\mathbf{x}^n))^{\mathrm{T}} \mathbf{C}_{\text{CSMC},k}^{-1} (\mathbf{x}^{-n} - \mathbf{m}_{\text{CSMC}}(\mathbf{x}^n)) \\
&\quad - (\mathbf{x}^{-n} - \mathbf{m}_{\text{Particle-mGRAD},k}(\mathbf{x}^n))^{\mathrm{T}} \mathbf{C}_{\text{Particle-mGRAD},k}^{-1} (\mathbf{x}^{-n} - \mathbf{m}_{\text{Particle-mGRAD},k}(\mathbf{x}^n))\big| \\
&\qquad = \Big| \sum_i (\mathbf{x}^i - \mathbf{m})^{\mathrm{T}} \mathbf{C}_k^{-1} (\mathbf{x}^i - \mathbf{m}) \\
&\qquad\quad - \frac{2}{\delta} \sum_i (\mathbf{x}^i - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m})^{\mathrm{T}} \mathbf{A}_k^{-1} (\mathbf{x}^i - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m}) \\
&\qquad\quad + \frac{2}{\delta} \sum_i \sum_j (\mathbf{x}^i - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m})^{\mathrm{T}} (\mathbf{I} + N\mathbf{A}_k)^{-1} (\mathbf{x}^j - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m}) \Big| \\
&\qquad = \Big| \frac{2}{\delta} \sum_i \sum_j (\mathbf{x}^i - \mathbf{m})^{\mathrm{T}} (\mathbf{I} + N\mathbf{A}_k)^{-1} (\mathbf{x}^j - \mathbf{m}) \\
&\qquad\quad + \frac{2}{\delta} \sum_i (\mathbf{x}^i - \mathbf{m})^{\mathrm{T}} \Big(\frac{\delta}{2} \mathbf{C}_k^{-1} - \mathbf{A}_k^{-1} \Big) (\mathbf{x}^i - \mathbf{m}) \\
&\qquad\quad + \frac{2}{\delta} \sum_i (\mathbf{x}^i - \mathbf{m})^{\mathrm{T}} \Big[\mathbf{I} + \Big(\frac{4(N-1)}{\delta} - 1 \Big) (\mathbf{I} + N\mathbf{A}_k)^{-1} \mathbf{A}_k \Big] (\mathbf{x}^n - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m}) \\
&\qquad\quad + N (\mathbf{x}^n - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m})^{\mathrm{T}} \Big[\mathbf{A}_k + \Big(\frac{2(N-1)}{\delta} - 1 \Big) \mathbf{A}_k (\mathbf{I} + N\mathbf{A}_k)^{-1} \mathbf{A}_k \Big] (\mathbf{x}^n - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m}) \Big| \\
&\qquad \leq \frac{2}{\delta} \sum_i \sum_j \|\mathbf{x}^i - \mathbf{m}\|_2 \|\mathbf{x}^j - \mathbf{m}\|_2 \|(\mathbf{I} + N\mathbf{A}_k)^{-1}\|_{2,2} \\
&\qquad\quad + \frac{2}{\delta} \sum_i \|\mathbf{x}^i - \mathbf{m}\|_2 \|\mathbf{x}^i - \mathbf{m}\|_2 \Big\| \frac{\delta}{2} \mathbf{C}_k^{-1} - \mathbf{A}_k^{-1} \Big\|_{2,2} \\
&\qquad\quad + \frac{2}{\delta} \sum_i \|\mathbf{x}^i - \mathbf{m}\|_2 \|\mathbf{x}^n - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m}\|_2 \Big\| \mathbf{I} + \Big(\frac{4(N-1)}{\delta} - 1 \Big) (\mathbf{I} + N\mathbf{A}_k)^{-1} \mathbf{A}_k \Big\|_{2,2} \\
&\qquad\quad + N \|\mathbf{x}^n - \boldsymbol{\phi}(\mathbf{x}^n) - \mathbf{m}\|_2^2 \Big\| \mathbf{A}_k + \Big(\frac{2(N-1)}{\delta} - 1 \Big) \mathbf{A}_k (\mathbf{I} + N\mathbf{A}_k)^{-1} \mathbf{A}_k \Big\|_{2,2} \\
&\qquad \leq C \Big[\sum_i \sum_j \|\mathbf{x}^i - \mathbf{m}\|_2 \|\mathbf{x}^j - \mathbf{m}\|_2 \\
&\qquad\qquad + (1 + \|\mathbf{x}^n - \mathbf{m}\|_2) \sum_i \|\mathbf{x}^i - \mathbf{m}\|_2 \\
&\qquad\qquad + \lambda_k (1 + \|\mathbf{x}^n - \mathbf{m}\|_2)^2 \Big],
\end{aligned}
\tag{37}
$$

for some constant $C \geq 0$ which only depends on N, δ and $\mathbf{m}$. Here, we have used that all the matrices inside the operator norms are simultaneously diagonalisable with $\mathbf{C}_k$ (so that the operator norms can be bounded above by some function of λ_k):

$$\begin{aligned}
\|(\mathbf{I}+N\mathbf{A}_k)^{-1}\|_{2,2} &\leq 1,\\
\Big\|\frac{\delta}{2}\mathbf{C}_k^{-1}-\mathbf{A}_k^{-1}\Big\|_{2,2} &= 1,\\
\Big\|\mathbf{A}_k+\Big(\frac{2(N-1)}{\delta}-1\Big)\mathbf{A}_k(\mathbf{I}+N\mathbf{A}_k)^{-1}\mathbf{A}_k\Big\|_{2,2} &\leq C'\frac{2\lambda_k}{2\lambda_k+\delta}\leq C'\frac{2}{\delta}\lambda_k,\\
\Big\|\mathbf{I}+\Big(\frac{4(N-1)}{\delta}-1\Big)(\mathbf{I}+N\mathbf{A}_k)^{-1}\mathbf{A}_k\Big\|_{2,2} &\leq C'',
\end{aligned}$$

for other constants $C', C'' \geq 0$.

Furthermore, by definition of $(F_k)_{k\geq 1}$,

$$\sup_{\mathbf{x}\in F_k}\|\mathbf{x}-\mathbf{m}\|_2 \in \mathrm{O}(\lambda_k^{(1-\varepsilon)/2}).$$

Consequently, for $i, j \in [N]_0$:

$$\begin{aligned}
&\sup_{\mathbf{x}^0\in F_k}\int_{\mathcal{X}^N}\mathrm{N}(\mathrm{d}\mathbf{x}^{-0};\mathbf{m}_{\mathrm{CSMC}},\mathbf{C}_{\mathrm{CSMC},k})\|\mathbf{x}^i-\mathbf{m}\|_2\|\mathbf{x}^j-\mathbf{m}\|_2\\
&\quad\in\begin{cases}\mathrm{O}(\lambda_k^{(1-\varepsilon)}), & \text{if } i=j=0,\\ \mathrm{O}(\lambda_k^{(2-\varepsilon)/2}), & \text{if either } i=0 \text{ or } j=0,\\ \mathrm{O}(\lambda_k), & \text{if neither } i=0 \text{ nor } j=0,\end{cases}
\end{aligned}\tag{38}$$

as $\lambda_k \to 0$, and where the last two cases follow from the Cauchy–Schwarz inequality. Similarly, for $i \in [N]_0$,

$$\sup_{\mathbf{x}^0\in F_k}\int_{\mathcal{X}^N}\mathrm{N}(\mathrm{d}\mathbf{x}^{-0};\mathbf{m}_{\mathrm{CSMC}},\mathbf{C}_{\mathrm{CSMC},k})\|\mathbf{x}^i-\mathbf{m}\|_2\in\begin{cases}\mathrm{O}(\lambda_k^{(1-\varepsilon)/2}), & \text{if } i=0,\\ \mathrm{O}(\lambda_k^{1/2}), & \text{if } i\neq 0,\end{cases}\tag{39}$$

as $\lambda_k \to 0$. Combining the bounds from (36)–(39) then shows that

$$\sup_{\mathbf{x}^0\in F_k} D^n_{\mathrm{CSMC},\text{Particle-mGRAD},k}(\mathbf{x}^0)\in\mathrm{O}(\lambda_k^{(1-\varepsilon)/2}),$$

for any $n \in [N]_0$. Plugging these bounds into (34) completes the proof. □

Proof (of Part 1 of Proposition 19) By Assumption **A1**, the model factorises over time and so do the Particle-MALA and Particle-mGRAD algorithms. Hence, without loss of generality, we again only prove the result in the case that $T = 1$ (and we again drop the 'time' subscript $t = 1$ hereafter).

For $\varepsilon \geq 1$ the result is trivially true but meaningless. Fix $\varepsilon \in (0, 1)$. Since G is integrable (by Assumption **A3**) and since π_k is invariant to scaling of G by a positive constant factor, we assume that $\int_{\mathcal{X}} G(\mathbf{x})\,\mathrm{d}\mathbf{x} = 1$, without loss of generality, so that G can be viewed as a density (and we will also use the symbol G to denote the corresponding distribution). Let $\mathbf{m}_G$ and $\mathbf{C}_G$ be mean and variance of G (which exist by Assumption **A3**) and define

$$F_k \coloneqq \{\mathbf{x}\in\mathcal{X} \mid \|\mathbf{x}-\mathbf{m}\|_2 \vee \sqrt{(\mathbf{x}-\mathbf{m}_G)^{\mathrm{T}}\mathbf{C}_G^{-1}(\mathbf{x}-\mathbf{m}_G)} < \lambda_k^{\varepsilon/2}\}.$$

We then have $\pi_k(F_k) = (1 + H_k)^{-1}$, where, letting $\mathbf{Y} \sim G$ and letting $F_k^{\mathrm{c}} := \mathcal{X} \setminus F_k$:

$$\begin{aligned}
H_k &:= \frac{\int_{F_k^{\mathrm{c}}} G(\mathbf{x})\,\mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{m}, \mathbf{C}_k)}{\int_{F_k} G(\mathbf{x})\,\mathrm{N}(\mathrm{d}\mathbf{x}; \mathbf{m}, \mathbf{C}_k)}\\
&\leq \frac{\int_{F_k^{\mathrm{c}}} G(\mathbf{x})\,\mathrm{d}\mathbf{x}}{\inf_{\mathbf{x}\in F_k} \exp(-\frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \lambda_k^{-1}) \int_{F_k} G(\mathbf{x})\,\mathrm{d}\mathbf{x}}\\
&\leq \frac{\int_{F_k^{\mathrm{c}}} G(\mathbf{x})\,\mathrm{d}\mathbf{x}}{\int_{F_k} G(\mathbf{x})\,\mathrm{d}\mathbf{x}} \exp(\tfrac{1}{2}\lambda_k^{\varepsilon-1})\\
&= \mathbb{P}(\mathbf{Y} \in F_k^{\mathrm{c}}) \frac{\exp(\frac{1}{2}\lambda_k^{\varepsilon-1})}{\int_{F_k} G(\mathbf{x})\,\mathrm{d}\mathbf{x}}\\
&\leq \mathbb{P}\Bigl(\sqrt{(\mathbf{Y}-\mathbf{m}_G)^{\mathrm{T}}\mathbf{C}_G^{-1}(\mathbf{Y}-\mathbf{m}_G)} \geq \lambda_k^{\varepsilon/2}\Bigr) \frac{\exp(\frac{1}{2}\lambda_k^{\varepsilon-1})}{\int_{F_k} G(\mathbf{x})\,\mathrm{d}\mathbf{x}}\\
&\leq \frac{D}{\lambda_k^{\varepsilon}} \frac{\exp(\frac{1}{2}\lambda_k^{\varepsilon-1})}{\int_{F_k} G(\mathbf{x})\,\mathrm{d}\mathbf{x}}\\
&\to 0.
\end{aligned}$$

The penultimate line follows from the (multidimensional) Chebyshev's inequality and the last line uses that $F_k \to \mathcal{X}$ as $k \to \infty$.

By the decomposition from (34), all that remains is to control the terms

$$\sup_{\mathbf{x}^0 \in F_k} D^n_{\text{Particle-MALA},\text{Particle-mGRAD},k}(\mathbf{x}^0),$$

for arbitrary $n \in [N]_0$

Firstly, by Lemma 29 from Appendix C, letting $\lambda(\mathbf{C}_k) = \{\lambda_{k,1}, \dots, \lambda_{k,D}\}$ denote the eigenvalues of $\mathbf{C}_k$ and noting that $\mathbf{A}_k$ is simultaneously diagonalisable with $\mathbf{A}_k^2$:

$$\begin{aligned}
&|\log(\det(\mathbf{C}_{\text{Particle-MALA}})) - \log(\det(\mathbf{C}_{\text{Particle-mGRAD},k}))|\\
&\quad = \Bigl|\sum_{d=1}^{D} N \log\Bigl(\frac{2\lambda_{k,d}+\delta}{2\lambda_{k,d}}\Bigr) + \log\Bigl(\frac{2\lambda_{k,d}+\delta}{2\lambda_{k,d}+\delta/(N+1)}\Bigr)\Bigr| \in \mathrm{O}(\lambda_k^{-1}).
\end{aligned} \tag{40}$$

Secondly, by Lemma 30 from Appendix C, and again with the conventions that $\sum_i$ is shorthand for $\sum_{i\in[N]_0\setminus\{n\}}$, that $\sum_j$ is shorthand for $\sum_{j\in[N]_0\setminus\{n\}}$, that $\sum_{i\neq j}$ is shorthand for $\sum_{j\in[N]_0\setminus\{n,i\}}$, and writing $\boldsymbol{\phi}(\mathbf{x}) := \kappa\frac{\delta}{2}\nabla \log G(\mathbf{x})$ as well as $\boldsymbol{\varphi}_k(\mathbf{x}) := \kappa\frac{\delta}{2}\nabla \log M_k(\mathbf{x}) = \kappa\frac{\delta}{2}\mathbf{C}_k^{-1}(\mathbf{m}-\mathbf{x})$, so that $\boldsymbol{\phi}(\mathbf{x}) + \boldsymbol{\varphi}_k(\mathbf{x}) = \kappa\frac{\delta}{2}\nabla \log \pi_k(\mathbf{x})$:

$$\begin{aligned}
&\bigl|(\mathbf{x}^{-n} - \mathbf{m}_{\text{Particle-MALA},k}(\mathbf{x}^n))^{\mathrm{T}}\mathbf{C}^{-1}_{\text{Particle-MALA}}(\mathbf{x}^{-n} - \mathbf{m}_{\text{Particle-MALA},k}(\mathbf{x}^n)\\
&\quad - (\mathbf{x}^{-n} - \mathbf{m}_{\text{Particle-mGRAD},k}(\mathbf{x}^n))^{\mathrm{T}}\mathbf{C}^{-1}_{\text{Particle-mGRAD},k}(\mathbf{x}^{-n} - \mathbf{m}_{\text{Particle-mGRAD},k}(\mathbf{x}^n))\bigr|\\
&\qquad \leq C\lambda_k^{-1}\Bigl[\sum_i\sum_j \|\mathbf{x}^i-\mathbf{m}\|_2\|\mathbf{x}^j-\mathbf{m}\|_2 + (1+\|\mathbf{x}^n-\mathbf{m}\|_2)\sum_{i=0}^{N}\|\mathbf{x}^i-\mathbf{m}\|_2\Bigr],
\end{aligned} \tag{41}$$

for some constant $C \geq 0$ which only depends on N, δ and $\mathbf{m}$. Here, we have followed the same steps as for (37) and used that all the matrices inside the operator norms are simultaneously diagonalisable with $\mathbf{C}_k$ (so that the operator norms can be bounded above by some function of λ_k^{-1}).

Furthermore, by definition of F_k, we have

$$\sup_{\mathbf{x}\in F_k} \|\mathbf{x}-\mathbf{m}\|_2 \in \mathrm{O}(\lambda_k^{\varepsilon/2}),$$

as $\lambda_k \to \infty$. Consequently, for $i, j \in [N]_0$, by the Cauchy–Schwarz inequality:

$$\begin{aligned}\sup_{\mathbf{x}^0\in F_k} \int_{\mathcal{X}^N} &\mathrm{N}(\mathrm{d}\mathbf{x}^{-0}; \mathbf{m}_{\text{Particle-MALA},k}, \mathbf{C}_{\text{Particle-MALA}})\|\mathbf{x}^i-\mathbf{m}\|_2\|\mathbf{x}^j-\mathbf{m}\|_2\\ &\in \begin{cases}\mathrm{O}(\lambda_k^{\varepsilon}), & \text{if } i=j=0,\\ \mathrm{O}(\lambda_k^{\varepsilon/2}), & \text{if either } i=0 \text{ or } j=0,\\ \mathrm{O}(1), & \text{if neither } i=0 \text{ nor } j=0,\end{cases}\end{aligned} \tag{42}$$

as $\lambda_k \to \infty$. Similarly, for $i \in [N]_0$,

$$\sup_{\mathbf{x}^0\in F_k} \int_{\mathcal{X}^N} \mathrm{N}(\mathrm{d}\mathbf{x}^{-0}; \mathbf{m}_{\text{Particle-MALA},k}, \mathbf{C}_{\text{Particle-MALA}})\|\mathbf{x}^i-\mathbf{m}\|_2 \in \begin{cases}\mathrm{O}(\lambda_k^{\varepsilon/2}), & \text{if } i=0,\\ \mathrm{O}(1), & \text{if } i\neq 0,\end{cases} \tag{43}$$

as $\lambda_k \to \infty$. Combining the bounds from (40)–(43) then shows that

$$\sup_{\mathbf{x}^0\in F_k} D^n_{\text{Particle-MALA},\text{Particle-mGRAD},k}(\mathbf{x}^0) \in \mathrm{O}(\lambda_k^{-(1-\varepsilon)/2}),$$

for any $n \in [N]_0$. Plugging these bounds into (34) completes the proof. □

E.4 Auxiliary MCMC kernels in the special case: $T = 1$

The π-invariant Markov kernels induced by the auxiliary-variable based algorithms discussed in this work can then be written as

$$P_a(\mathrm{d}\tilde{\mathbf{x}}|\mathbf{x}) = \sum_{l=0}^{N} \int_{\mathcal{X}^{N+3}} \delta_{\mathbf{x}}(\,\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0}\times \mathrm{d}\mathbf{u}|\mathbf{x}^{-0},\mathbf{x}^0)\Psi^l(\{h_a^n(\mathbf{x}^{0:N},\mathbf{u})\}_{n=1}^N)\delta_{\mathbf{x}^l}(\mathrm{d}\tilde{\mathbf{x}}),$$

where have appealed to symmetry to always place the reference 'path' in position 0, where 'a' is now a placeholder for 'Particle-aGRAD', 'Particle-aMALA', or 'CSMC' and with

$$\begin{aligned}q_a^{-n}(\mathbf{x}^{-n},\mathbf{u}|\mathbf{x}^n) &= q_a^{-n}(\mathbf{x}^{-n}|\mathbf{x}^n)q_a^{-n}(\mathbf{u}|\mathbf{x}^{-n},\mathbf{x}^n),\\ h_a^n(\mathbf{x}^{0:N},\mathbf{u}) &\coloneqq \log q_a^{-n}(\mathbf{x}^{-n}|\mathbf{x}^n) - \log q_a^{-0}(\mathbf{x}^{-0}|\mathbf{x}^0)\\ &\qquad + \mathbb{I}\{a\neq \text{CSMC}\}[\log q_a^{-n}(\mathbf{u}|\mathbf{x}^{-n},\mathbf{x}^n) - \log q_a^{-0}(\mathbf{u}|\mathbf{x}^{-0},\mathbf{x}^0)],\\ q_a^{-n}(\mathbf{x}^{-n}|\mathbf{x}^n) &= \mathrm{N}(\mathbf{x}^{-n}; \mathbf{m}_a(\mathbf{x}^n), \mathbf{C}_a),\\ q_a^{-n}(\mathbf{u}|\mathbf{x}^{-n},\mathbf{x}^n) &= \mathrm{N}(\mathbf{u}; \boldsymbol{\nu}_a(\mathbf{x}^{-n},\mathbf{x}^n), \boldsymbol{\Upsilon}_a),\end{aligned}$$

where again $\mathbf{m}_a(\mathbf{x}^n) \in \mathbb{R}^{ND}$ and $\boldsymbol{\nu}_a(\mathbf{x}^{-n},\mathbf{x}^n) \in \mathbb{R}^D$ are suitable mean vector, and $\mathbf{C}_a \in \mathbb{R}^{(ND)\times(ND)}$, $\boldsymbol{\Upsilon}_a \in \mathbb{R}^{D\times D}$ are suitable covariance variance matrices, and we again write

$\mathbf{x}^{-n} := \mathrm{vec}(\mathbf{x}^{-n})$. Specifically,

$$\begin{aligned}
\mathbf{m}_{\text{Particle-aGRAD}}(\mathbf{x}^n) &= \mathbf{m}_{\text{Particle-mGRAD}}(\mathbf{x}^n) \\
\mathbf{C}_{\text{Particle-aGRAD}} &= \mathbf{C}_{\text{Particle-mGRAD}}, \\
\boldsymbol{\nu}_{\text{Particle-aGRAD}}(\mathbf{x}^{-n}, \mathbf{x}^n) &= (\mathbf{I} + N\mathbf{A})^{-1}[(N+1)\bar{\mathbf{x}} + \boldsymbol{\phi}(\mathbf{x}^n) + N(\mathbf{A} - \mathbf{I})\mathbf{m}], \\
\boldsymbol{\Upsilon}_{\text{Particle-aGRAD}} &= \tfrac{\delta}{2}(\mathbf{I} + N\mathbf{A})^{-1}, \\
\mathbf{m}_{\text{Particle-aMALA}}(\mathbf{x}^n) &= \mathbf{m}_{\text{Particle-aMALA}}(\mathbf{x}^n), \\
\mathbf{C}_{\text{Particle-MALA}} &= \mathbf{C}_{\text{Particle-MALA}}, \\
\boldsymbol{\nu}_{\text{Particle-aMALA}}(\mathbf{x}^{-n}, \mathbf{x}^n) &= \bar{\mathbf{x}} + \tfrac{1}{N+1}\boldsymbol{\phi}(\mathbf{x}^n), \\
\boldsymbol{\Upsilon}_{\text{Particle-aMALA}} &= \tfrac{\delta}{2(N+1)}\mathbf{I},
\end{aligned}$$

by Part 2 of Lemma 31 and Lemma 30 from Appendix C. Of course, the standard CSMC algorithm does not make use of the auxiliary variable $\mathbf{u}$, so we extend the space to include $\mathbf{u}$ with

$$\boldsymbol{\nu}_{\text{CSMC}}(\mathbf{x}^{-n}, \mathbf{x}^n) = \boldsymbol{\nu}_{\text{Particle-aGRAD}}(\mathbf{x}^{-n}, \mathbf{x}^n), \tag{44}$$

$$\boldsymbol{\Upsilon}_{\text{CSMC}} = \boldsymbol{\Upsilon}_{\text{Particle-aGRAD}}. \tag{45}$$

Key to our proofs will be the following bound which follows by the triangle inequality and a telescoping-sum decomposition (here: a is again a placeholder which takes values in $\{\text{CSMC}, \text{Particle-MALA}\}$ whilst we will always set $b = \text{Particle-aGRAD}$ and $q_a^{-m}(\,\cdot\,|\mathbf{x})$; and, unless otherwise stated, $q_b^{-m}(\,\cdot\,|\mathbf{x})$ denote the joint distributions on the space that includes the auxiliary variable $\mathbf{u}$):

$$\begin{aligned}
&\|P_a(\,\cdot\,|\mathbf{x}) - P_b(\,\cdot\,|\mathbf{x})\|_{\text{TV}} \\
&\quad\leq \|q_a^{-0}(\,\cdot\,|\mathbf{x}) - q_b^{-0}(\,\cdot\,|\mathbf{x})\|_{\text{TV}} \\
&\qquad + \sum_{l=0}^{N} \int_{\mathcal{X}^{N+2}} \delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0} \times \mathrm{d}\mathbf{u}|\mathbf{x}^0) \\
&\qquad\qquad \times |\Psi^l(\{h_a^n(\mathbf{x}^{0:N}, \mathbf{u})\}_{n=1}^N) - \Psi^l(\{h_b^n(\mathbf{x}^{0:N}, \mathbf{u})\}_{n=1}^N)| \\
&\quad\leq \sqrt{\mathrm{KL}(q_a^{-0}(\,\cdot\,|\mathbf{x}) \| q_b^{-0}(\,\cdot\,|\mathbf{x}))} \\
&\qquad + \sum_{l=0}^{N} [\Psi^l]_{\text{LIP}} \int_{\mathcal{X}^{N+2}} \delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0} \times \mathrm{d}\mathbf{u}|\mathbf{x}^0) \sum_{n=1}^{N} |h_a^n(\mathbf{x}^{0:N}, \mathbf{u}) - h_b^n(\mathbf{x}^{0:N}, \mathbf{u})| \\
&\quad\leq C\Big[\sqrt{D_{a,b}^0(\mathbf{x}) + E_{a,b}^0(\mathbf{x})} + \sum_{n=0}^{N} D_{a,b}^n(\mathbf{x}) + \widetilde{E}_{a,b}^n(\mathbf{x})\Big].
\end{aligned}$$

Here, the penultimate line follows from Pinsker's inequality and the Lipschitz continuity of the selection function; $C \geq 0$ is some constant which may depend on these Lipschitz constants and on N and D; $D_{a,b}^n(\mathbf{x})$ is defined exactly as in the marginal case (35). Furthermore, we have defined

$$E_{a,b}^n(\mathbf{x}) := \int_{\mathcal{X}^{N+1}} \delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}^0) q_a^{-0}(\mathrm{d}\mathbf{x}^{-0} \times \mathrm{d}\mathbf{u}|\mathbf{x}^0) |\log q_a^{-n}(\mathbf{u}|\mathbf{x}^{-n}, \mathbf{x}^n) - \log q_b^{-n}(\mathbf{u}|\mathbf{x}^{-n}, \mathbf{x}^n)|. \tag{46}$$

Finally, if $a \neq$ CSMC and $b \neq$ CSMC, we we have defined

$$\widetilde{E}^n_{\text{Particle-aMALA},\text{Particle-aGRAD}}(\mathbf{x}) \coloneqq E^n_{\text{Particle-aMALA},\text{Particle-aGRAD}}(\mathbf{x}),$$

whilst

$$\begin{aligned}&\widetilde{E}^n_{\text{CSMC},\text{Particle-aGRAD}}(\mathbf{x})\\ &\quad\coloneqq \int_{\mathcal{X}^{N+2}} \delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}^0) q^{-0}_{\text{CSMC}}(\mathrm{d}\mathbf{x}^{-0} \times \mathrm{d}\mathbf{u}|\mathbf{x}^0)\\ &\qquad\times |\log q^{-n}_{\text{Particle-aGRAD}}(\mathbf{u}|\mathbf{x}^{-n}, \mathbf{x}^n) - \log q^{-0}_{\text{Particle-aGRAD}}(\mathbf{u}|\mathbf{x}^{-0}, \mathbf{x}^0)|.\end{aligned}$$

E.5 Proofs of Part 2

Proof (of Part 2 of Proposition 18) Assume the same setting as in the proof of Part 1 of Proposition 18 with the same definition of F_k.

We proceed by controlling the terms in (46). Note that $D^n_{\text{CSMC},\text{Particle-aGRAD},k} = D^n_{\text{CSMC},\text{Particle-mGRAD},k}$. Hence, by the arguments from the proof of Part 1 of Proposition 18,

$$\sup_{\mathbf{x}\in F_k} D^n_{\text{CSMC},\text{Particle-aGRAD},k}(\mathbf{x}) \in \mathrm{O}(\lambda_k^{(1-\varepsilon)/2}).$$

Additionally, due to (44)–(45), $E^n_{\text{CSMC},\text{Particle-aGRAD}}(\mathbf{x}) = 0$. Finally, using similar arguments as in the proofs for the ‘marginal’ algorithm, we can verify that

$$\sup_{\mathbf{x}\in F_k} \widetilde{E}^n_{\text{CSMC},\text{Particle-aGRAD},k}(\mathbf{x}) \in \mathrm{O}(\lambda_k^{(1-\varepsilon)/2}).$$

This completes the proof. □

Proof (of Part 2 of Proposition 19) Assume the same setting as in the proof ofPart 1 of Proposition 19 with the same definition of F_k.

We proceed by controlling the terms in (46). Note that $D^n_{\text{Particle-aMALA},\text{Particle-aGRAD},k} = D^n_{\text{Particle-MALA},\text{Particle-mGRAD},k}$. Hence, by the arguments from the proof Part 1 of Proposition 19,

$$\sup_{\mathbf{x}\in F_k} D^n_{\text{Particle-aMALA},\text{Particle-aGRAD},k}(\mathbf{x}) \in \mathrm{O}(\lambda_k^{-(1-\varepsilon)/2}).$$

Additionally, using similar arguments as in the proofs for the ‘marginal’ algorithm, we can verify that

$$\sup_{\mathbf{x}\in F_k} E^n_{\text{Particle-aMALA},\text{Particle-aGRAD},k}(\mathbf{x}) \in \mathrm{O}(\lambda_k^{-(1-\varepsilon)}).$$

This completes the proof. □

Appendix F. Step-size adaptation

All our algorithms involve the calibration of several step sizes δ_t, one for each time step. To calibrate these, we implement a routine that recursively increases or decreases δ_t if the running average of the acceptance rate α_t (i.e., the relative frequency with which $\mathbf{x}_t$ is updated) is respectively above or below a pre-specified target acceptance rate (in our experiments, we picked this to be $\alpha^* = 75\,\%$). The only exception to this lies in the twisted algorithms of Section 4.4 which we calibrate using a single step-size δ (so that $\delta = \delta_1 = \ldots = \delta_T$), and for which the target relates to the overall acceptance rate averaged across time steps. The reason for this difference stems from the fact that the twisting causes the acceptance rate at time s additionally depend on *future* auxiliary variables $\mathbf{u}_t$, and therefore the future step-size parameters δ_t (for $t > s$), thereby making the behaviour per time-step harder to control. In practice, our calibration of the twisted Particle-aGRAD is therefore more similar to that of aGRAD than that of our other algorithms. The adaptation procedure is summarised in the following algorithm.

Algorithm 37 (step-size adaptation)

```
1. Initialise the trajectory $\mathbf{x}_{1:T}[0]$, the initial step sizes $\delta_t[0]$ (for $t \in [T]$), the initial learning rate $\rho[0] = \frac{1}{2}$.
2. Initialise the history of accepted time steps $A \coloneqq (A_{w,t}) \in \{0,1\}^{W \times T}$, with $0$ everywhere.
3. For $k = 1, \ldots, K$,
    (a) sample $\mathbf{x}_{1:T}[k] \sim P(\,\cdot\,|\mathbf{x}_{1:T}[k-1])$, where $P$ denotes the Markov kernel induced by one of the algorithms discussed in this work with step sizes $\delta_{1:T}$ set equal to $\delta_{1:T}[k-1]$,
    (b) roll the array $A$ by one: set $A_{2:\min\{W,k\},t} \coloneqq A_{1:\min\{W-1,k-1\},t}$, and $A_{1,t} = \mathbb{I}\{\mathbf{x}_t[k] = \mathbf{x}_t[k-1]\}$, for $t \in [T]$,
    (c) compute $\alpha_t = \frac{1}{\min\{W,k\}} \sum_{w=1}^{\min\{W,k\}} A_{w,t}$, for $t \in [T]$,
    (d) if $|\alpha_t - \alpha^*| < \sigma$ then keep $\delta_t[k] = \delta_t[k-1]$ unchanged;
        otherwise, set
            $$\delta_t[k] \coloneqq \delta_t[k-1] + \max\{k^\gamma \rho, \rho_{\min}\}(\alpha_t - \alpha)/\alpha^*.$$
```

In our experiments, we took $\sigma = 5\,\%$, $K = 10\,000$, $\delta_t[0] = 10^{-2}$, $W = 100$, $\rho = \frac{1}{2}$, $\rho_{\min} = 10^{-3}$, $\gamma = -\frac{1}{2}$.

Appendix G. Additional experimental results

In this section, we provide additional simulation results for the multivariate stochastic volatility model experiments from Section 5.

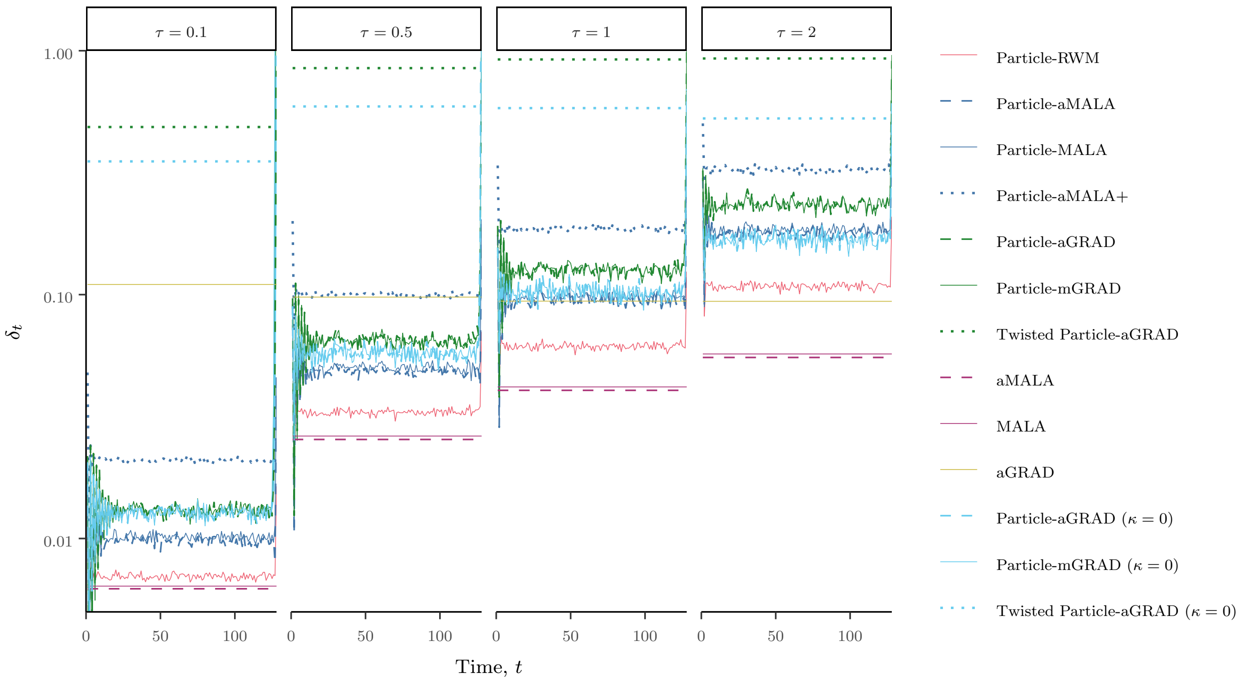


Figure 6: Adaptation of the step-size parameters δ_t, averaged across all four chains and all five simulated data sets (per value of τ) in the multivariate stochastic volatility model.

G.1 Calibrated step sizes and acceptance rates

Recall that the step sizes δ_t were calibrated to achieve an acceptance rate of 75 %. Here, the 'acceptance rate' at time t refers to the relative frequency with which the state $\mathbf{x}_t$ is updated. The calibrated step sizes are shown in Figure 6; and the corresponding acceptance rates are shown in Figure 7.

The results are averaged over the four chains and five simulated data sets. We do not report CSMC as it does not require calibration. All methods consistently resulted in acceptance rates close to the target 75 %. Only the twisted Particle-aGRAD algorithm showed more instability as the informativeness of the prior decreased: this is because, contrary to the methods, only a single step-size is used for all time steps, so calibrating for the informativeness of individual observations is not feasible. This seems to hint to the fact that the twisted Particle-aGRAD, *under our proposed calibration,* is less robust than alternatives to heterogeneous levels of informativeness.

G.2 Breakdown of CSMC, aMALA and MALA

In this section, we illustrate the breakdown of CSMC, aMALA and MALA.

Firstly, Figure 8 illustrates that the estimates of the marginal posterior means of $x_{t,15}$ (the 15th component of the state at time t) produced by CSMC, aMALA and MALA differ substantially from those produced by all the other algorithms. We emphasise that the

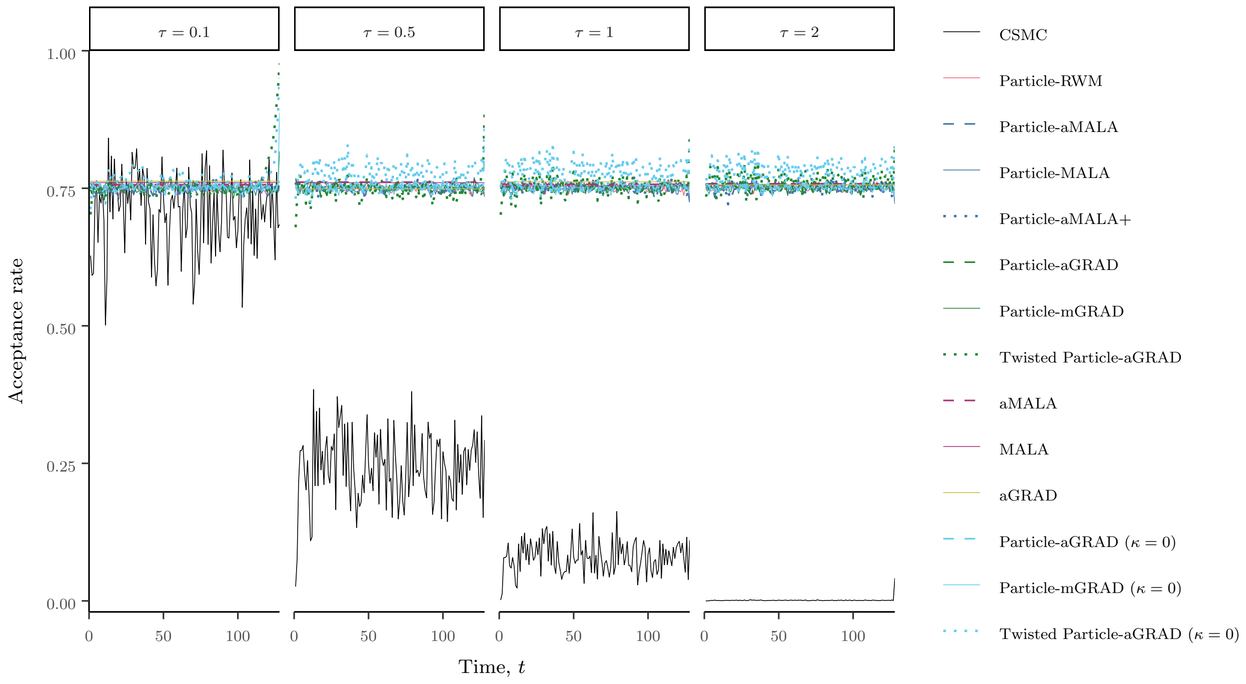


Figure 7: Acceptance rates (i.e., relative frequencies with which states are updated), averaged across all four chains and all five simulated data sets (per value of τ) in the multivariate stochastic volatility model.

15th component was arbitrarily chosen as an example and is representative of the other components.

Secondly, Figure 9 illustrates that the energy traces of CSMC, aMALA and MALA differ substantially from those of all the other algorithms. Here, the energy is defined as $\log \pi_T(\mathbf{x}_{1:T}[i])$, where $\mathbf{x}_{1:T}[i]$ is the sample from the ith iteration after burn-in. Such energy traces serve as a visual illustration of both stationarity and mixing speed: if the energy trace of a sampler differs too much from the others, or is not consistent across the independent Markov chains we used, the sampler is unlikely to perform correctly.

G.3 Effective sample sizes

In this section, in Figures 10–12 report the minimum, median and maximum ESS and ESS per second (averaged across all four chains and all five simulated data sets) individually for each time step $t = 1, \dots, T$.

G.4 Autocorrelation

Figure 13 shows the autocorrelation (corrected using Vehtari et al., 2021) of the energy from Figure 9. This serves as a visual confirmation of the statistical performance of the different algorithms considered under several prior dispersion regimes: as expected, the twisted Particle-aGRAD dominates all other alternatives, while Particle-aMALA+ domi-

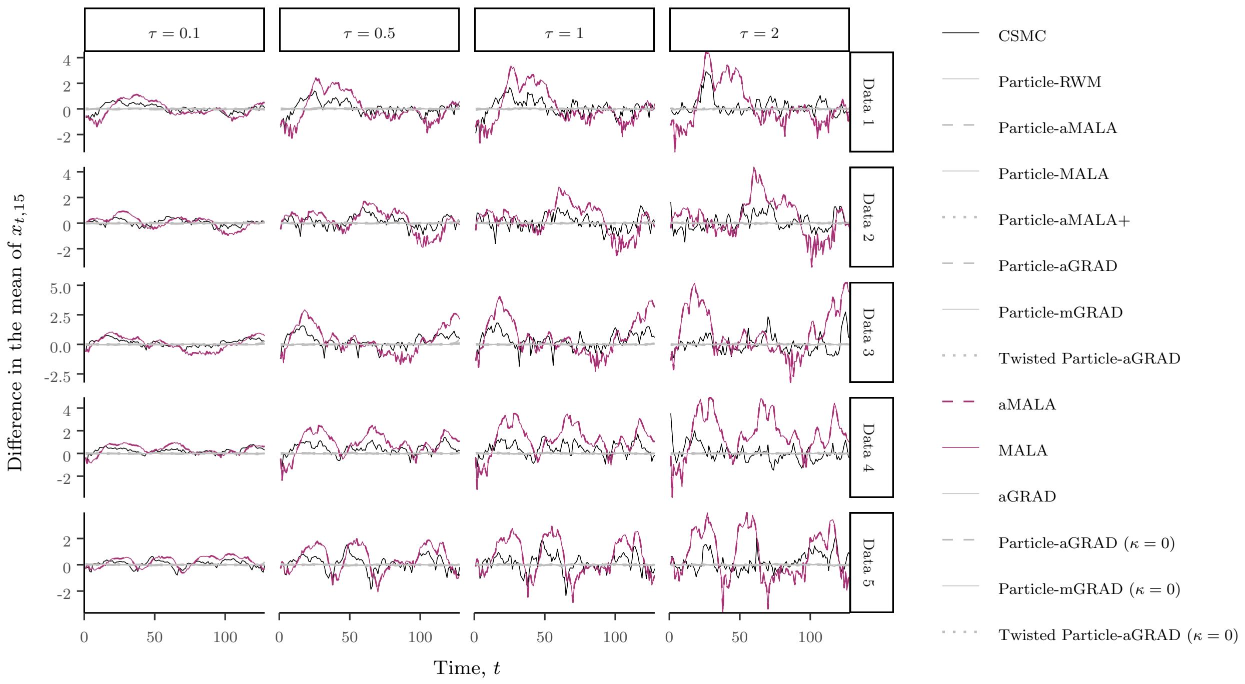


Figure 8: Estimated posterior mean of $x_{t,15}$ minus the estimated posterior mean of $x_{t,15}$ under the aGRAD algorithm, averaged across all four chains for each of the five simulated data sets (per value of τ) in the multivariate stochastic volatility model. The figure shows that the estimated posterior means of CSMC, aMALA and MALA differ substantially from those of all the other algorithms.

nates other alternatives, including aGRAD as soon as the prior variance is large enough, followed by Particle-aGRAD/Particle-aGRAD, and then by Particle-aMALA/Particle-MALA, with Particle-RWM being the least efficient.

References

Christophe Andrieu and Matti Vihola. Establishing some order amongst exact approximations of MCMCs. *Annals of Applied Probability*, 26(5):2661–2696, 2016.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2009.00736.x. With discussion.

Christophe Andrieu, Anthony Lee, and Matti Vihola. Uniform ergodicity of the iterated conditional SMC and geometric ergodicity of particle Gibbs samplers. *Bernoulli*, 24(2): 842–872, 2018. doi: 10.3150/15-BEJ785.

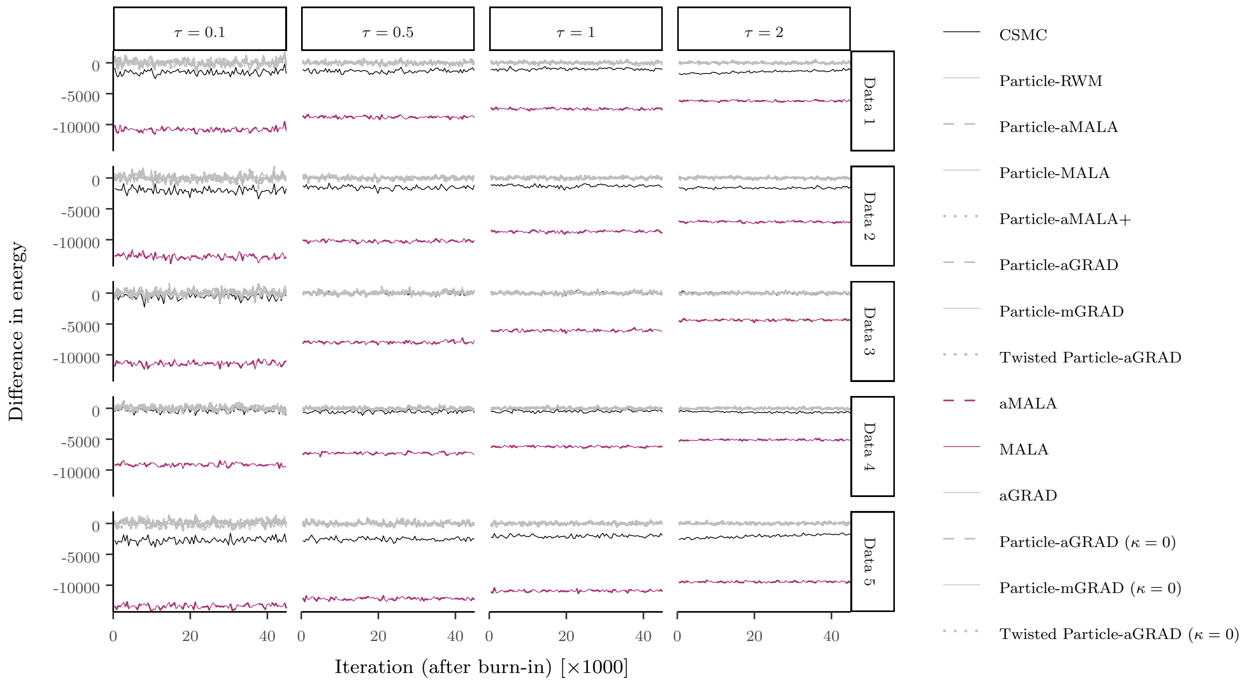


Figure 9: Energy (i.e., $\log \pi_T(\mathbf{x}_{1:T}[i]) + \text{const}$, where $\mathbf{x}_{1:T}[i]$ is the sample from the ith iteration after burn-in) minus the energy under the aGRAD algorithm, averaged across all four chains for each of the five simulated data sets (per value of τ) in the multivariate stochastic volatility model. The figure shows that the energy traces of CSMC, aMALA and MALA differ substantially from those of all the other algorithms.

J. E. Besag. Contribution to the discussion on ‘Representations of knowledge in complex systems’ by Grenander, U and Miller, M. I.. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 56(4):549–581, 1994.

D. M. Ceperley and M. Dewing. The penalty method for random walks with uncertain energies. *The Journal of Chemical Physics*, 110(20):9812–9820, 1999.

Nicolas Chopin and Sumeetpal S Singh. On particle Gibbs sampling. *arXiv e-prints*, 2013.

Adrien Corenflos and Simo Särkkä. Auxiliary MCMC and particle Gibbs samplers for parallelisable inference in latent dynamical systems. *arXiv preprint arXiv:2303.00301*, 2023.

Adrien Corenflos, Nicolas Chopin, and Simo Särkkä. De-sequentialized Monte Carlo: A parallel-in-time particle smoother. *Journal of Machine Learning Research*, 23(283):1–39, 2022.

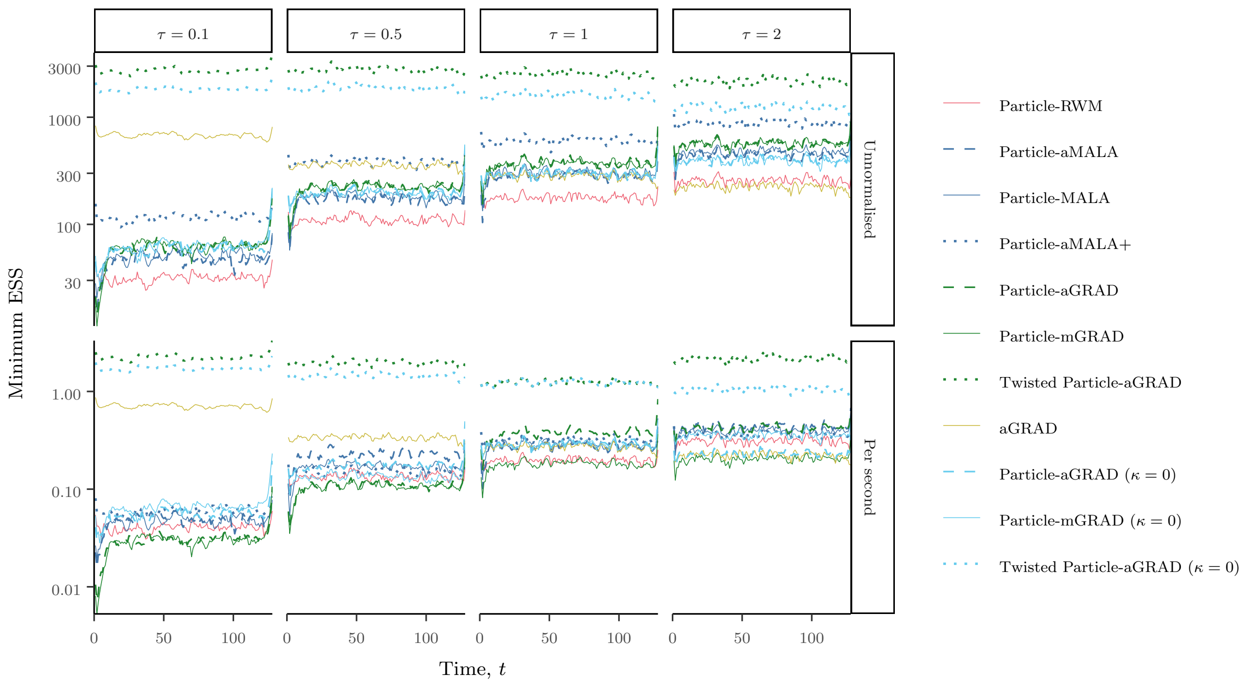


Figure 10: Minimum ESS and ESS per second averaged across all four chains and all five simulated data sets (per value of τ) in the multivariate stochastic volatility model.

S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC methods for functions: Modifying old algorithms to make them faster. *Statistical Science*, 28(3):424–446, 2013. doi: 10.1214/13-STS421. URL `https://doi.org/10.1214/13-STS421`.

Paul Fearnhead and Loukia Meligkotsidou. Augmentation schemes for particle MCMC. *Statistics and Computing*, 26:1293–1306, 2016.

Axel Finke. *On Extended State-Space Constructions for Monte Carlo Methods*. PhD thesis, Department of Statistics, University of Warwick, UK, 2015.

Axel Finke and Alexandre H Thiery. Conditional sequential Monte Carlo in high dimensions. *The Annals of Statistics*, 51(2):437–463, 2023.

Axel Finke, Arnaud Doucet, and Adam M. Johansen. On embedded hidden Markov models and particle Markov chain Monte Carlo methods. *arXiv e-prints*, art. arXiv:1610.08962, October 2016. doi: https://doi.org/10.48550/arXiv.1610.08962.

Pieralberto Guarniero, Adam M Johansen, and Anthony Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647, 2017. doi: https://doi.org/10.1080/01621459.2016.1222291.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. doi: 10.1093/biomet/57.1.97.

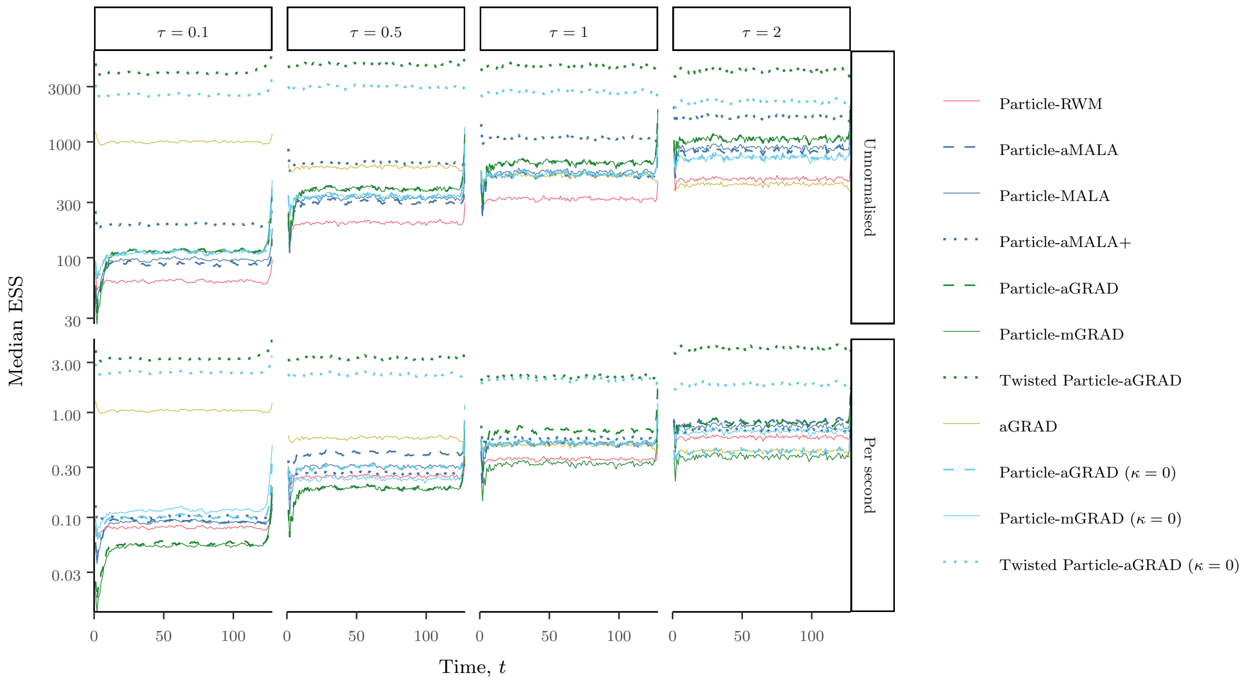


Figure 11: Medium ESS and ESS per second averaged across all four chains and all five simulated data sets (per value of τ) in the multivariate stochastic volatility model.

Harold V Henderson and Shayle R Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(1):53–60, 1981.

Jeremy Heng, Adrian N. Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential Monte Carlo. *The Annals of Statistics*, 48(5):2904 – 2929, 2020. doi: 10.1214/19-AOS1914. URL https://doi.org/10.1214/19-AOS1914.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45, 1960.

Joona Karjalainen, Anthony Lee, Sumeetpal S. Singh, and Matti Vihola. Mixing time of the conditional backward sampling particle filter. *arXiv e-prints*, 2023.

Santeri Karppinen and Matti Vihola. Conditional particle filters with diffuse initial distributions. *Statistics and Computing*, 31:1–14, 2021.

Santeri Karppinen, Sumeetpal S. Singh, and Matti Vihola. Conditional particle filters with bridge backward sampling. *Journal of Computational and Graphical Statistics*, 0(0):1–15, 2023. doi: 10.1080/10618600.2023.2231514.

Anthony Lee, Sumeetpal S Singh, and Matti Vihola. Coupled conditional backward sampling particle filter. *Annals of Statistics*, 48(5):3066–3089, 2020. doi: 10.1214/19-AOS1922.

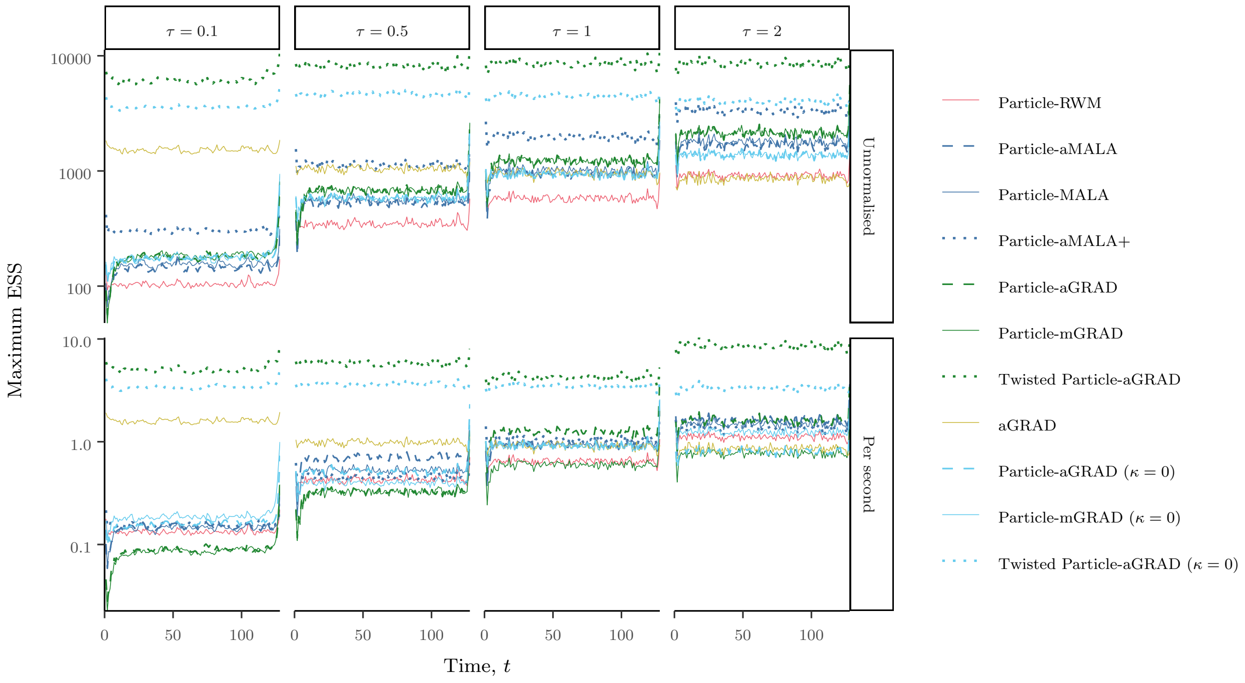


Figure 12: Maximum ESS and ESS per second averaged across all four chains and all five simulated data sets (per value of τ) in the multivariate stochastic volatility model.

Fredrik Lindsten, Michael I. Jordan, and Thomas B. Schön. Ancestor sampling for particle Gibbs. In *Proceedings of the 2012 Conference on Neural Information Processing Systems*, Lake Tahoe, NV, 2012.

Fredrik Lindsten, Randal Douc, and Eric Moulines. Uniform ergodicity of the particle Gibbs sampler. *Scandinavian Journal of Statistics*, 42(3):775–797, 2015. doi: https://doi.org/10.1111/sjos.12136.

Fredrik Lindsten, Adam M Johansen, Christian A Naesseth, Bonnie Kirkpatrick, Thomas B Schön, John AD 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. Peskun's theorem and a modified discrete-state Gibbs sampler. *Biometrika*, 83 (3):681–682, 1996.

Samuel Livingstone and Giacomo Zanella. The Barker proposal: Combining robustness and efficiency in gradient-based MCMC. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(2):496–523, 01 2022. ISSN 1369-7412. doi: 10.1111/rssb.12482.

Sean Malory. *Bayesian inference for stochastic processes.* PhD thesis, Lancaster University, 2021.

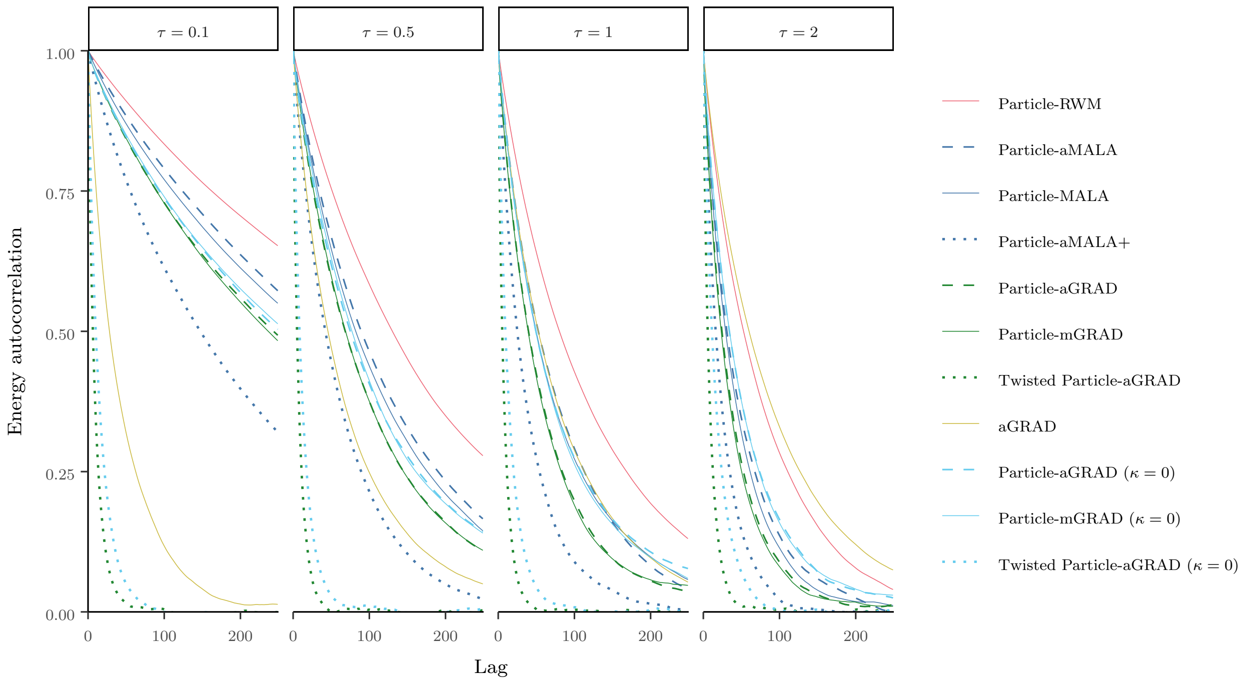


Figure 13: Autocorrelation of the energy from Figure 9 in the multivariate stochastic volatility model.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi: 10.1063/1.1699114.

Lawrence M Murray, Emlyn M Jones, and John Parslow. On disturbance state-space models and the particle marginal Metropolis–Hastings sampler. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):494–521, 2013.

G. K. Nicholls, C. Fox, and A. Muir Watt. Coupled MCMC with a randomized acceptance probability. *arXiv e-prints*, May 2012.

Benjamin Rhodes and Michael Gutmann. Enhanced gradient-based MCMC in discrete spaces. *arXiv e-prints*, 2022.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling for various Metropolis–Hastings algorithms. *Statistical Science*, 16(4):351–367, 2001.

Gareth O Roberts, Andrew Gelman, and Walter R Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1): 110–120, 1997. doi: 10.1214/aoap/1034625254.

Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge University Press, 2023.

Chris Sherlock and Alexandre H Thiery. A discrete bouncy particle sampler. *Biometrika*, 109(2):335–349, 2022.

Alexander Y Shestopaloff and Radford M Neal. Sampling latent states for high-dimensional non-linear state space models with the embedded HMM method. *Bayesian Analysis*, 13 (3):797–822, 2018. doi: 10.1214/17-BA1077.

Sumeetpal S Singh, Fredrik Lindsten, and Eric Moulines. Blocking strategies and stability of particle Gibbs samplers. *Biometrika*, 104(4):953–969, 2017. doi: https://doi.org/10.1093/biomet/asx051.

Michalis K Titsias. Contribution to the discussion on 'Riemann manifold Langevin and Hamiltonian Monte Carlo methods' by Girolami, M., and Calderhead, b. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(2):123–214, 2011.

Michalis K Titsias and Omiros Papaspiliopoulos. Auxiliary gradient-based sampling algorithms. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80 (4):749–767, 2018.

Håkon Tjelmeland. Using all Metropolis–Hastings proposals to estimate mean values. preprint 4/2004, Norwegian University of Science and Technology, Trondheim, Norway, 2004.

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC (with discussion). *Bayesian Analysis*, 16(2):667–718, 2021. doi: 10.1214/20-BA1221. URL https://doi.org/10.1214/20-BA1221.

Jure Vogrinc and Wilfrid S Kendall. Counterexamples for optimal scaling of Metropolis–Hastings chains with rough target densities. *The Annals of Applied Probability*, 31(2): 972–1019, 2021.

Nick Whiteley. Contribution to the discussion on 'Particle Markov chain Monte Carlo methods' by Andrieu, C., Doucet, A., and Holenstein, R. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):306–307, 2010.

Nick Whiteley and Anthony Lee. Twisted particle filters. *The Annals of Statistics*, 42(1):115–141, 2014. doi: 10.1214/13-AOS1167. URL https://doi.org/10.1214/13-AOS1167.

Giacomo Zanella. Informed proposals for local MCMC in discrete spaces. *Journal of the American Statistical Association*, 115(530):852–865, 2020. doi: 10.1080/01621459.2019.1585255.