Electronic Journal of Statistics
Vol. 19 (2025) 1370–1424
ISSN: 1935-7524
https://doi.org/10.1214/25-EJS2363

Auxiliary MCMC samplers for parallelisable inference in high-dimensional latent dynamical systems

Adrien Corenflos

Department of Statistics, University of Warwick

Department of Electrical Engineering and Automation, Aalto University, e-mail: adrien.corenflos@warwick.ac.uk

and

Simo Särkkä

Department of Electrical Engineering and Automation, Aalto University, e-mail: simo.sarkka@aalto.fi

Abstract: Sampling from the full posterior distribution of high-dimensional non-linear, non-Gaussian latent dynamical models presents significant computational challenges. While Particle Gibbs (also known as conditional sequential Monte Carlo) is considered the gold standard for this task, it quickly degrades in performance as the latent space dimensionality increases. Conversely, globally Gaussian-approximated methods like extended Kalman filtering, though more robust, are seldom used for posterior sampling due to their inherent bias. We introduce novel auxiliary sampling approaches that address these limitations. By incorporating artificial observations of the system as auxiliary variables in our MCMC kernels, we develop both efficient exact Kalman-based samplers and enhanced Particle Gibbs algorithms that maintain performance in high-dimensional latent spaces. Some of our methods support parallelization along the time dimension, achieving logarithmic scaling when implemented on GPUs. Empirical evaluations demonstrate superior statistical and computational performance compared to existing approaches for high-dimensional latent dynamical systems.

Keywords and phrases: Feynman–Kac models, state-space models, particle MCMC, Kalman filtering, parameter estimation.

Received July 2023.

Contents

1 Introduction 1371
1.1 Gaussian approximated state-space models 1373
1.2 Sequential Monte Carlo 1374
1.3 Motivation and contributions 1374

2 Auxiliary Kalman samplers 1376
2.1 Auxiliary gradient-based samplers 1376
2.2 Auxiliary Kalman samplers 1378
2.3 New auxiliary samplers for models with non-Gaussian dynamics 1381
2.4 Sampling and evaluating the posterior of LGSSMs 1384
2.4.1 Forward-filtering backward-sampling for LGSSMs 1384
2.4.2 Parallel-in-time sampling of LGSSMs 1386
3 Auxiliary particle Gibbs samplers 1387
3.1 SMC and particle Gibbs algorithms 1388
3.2 Particle Gibbs for Feynman–Kac models with auxiliary observations 1390
3.3 Adapted proposals in particle Gibbs with auxiliary observations 1392
3.3.1 Differentiable models 1393
3.3.2 Approximately Gaussian transitions 1394
3.3.3 Hybrid proposal models 1395
3.4 Extension to pseudo-marginal methods 1395
4 Experimental evaluation 1396
4.1 Multivariate stochastic volatility model 1398
4.2 Spatio-temporal model 1400
4.3 Parameter estimation in a continuous-discrete diffusion smoothing problem 1402
4.4 Failure modes 1404
5 Discussion 1406
A Sampling and evaluating LGSSM pathwise smoothing distributions . 1408
A.1 Sequential implementations 1408
A.2 Parallel implementations 1410
A.2.1 Prefix-sums and the parallel Kalman filter 1410
A.2.2 Parallel Rauch–Tung–Striebel sampler 1412
A.2.3 Divide-and-conquer strategies 1413
B Generalised statistical linear regression 1415
C Backward sampling and parallel-in-time particle Gibbs 1416
C.1 Backward sampling 1417
C.2 Parallel-in-time particle Gibbs 1417
D Sequential results for the spatio-temporal experiment of Section 4.2 . 1418
Individual contributions 1419
Acknowledgments 1419
Funding 1419
References 1420

1. Introduction

State-space models [SSMs, see, e.g., 22, 68, 13], otherwise known as hidden Markov models, are a class of dynamic statistical models routinely employed to model phenomena in bio-medicine, epidemiology, chemistry, or economics. For a given finite horizon $T > 0$, they are fully described by the joint distribution over

their latent states and the observations, which, when it exists, can be identified with its density

$$p(x_{0:T}, y_{0:T}) \coloneqq p_0(x_0) \left\{ \prod_{t=0}^{T} h_t(y_t \mid x_t) \right\} \left\{ \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}) \right\}. \tag{1}$$

In this formulation, p_0 represents the initial distribution of the state x_0, while p_t and h_t represent the (conditional) transition and emission distributions 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 sampling, or computing expectations with respect to the conditional distribution $p(x_t \mid y_{0:t})$, where $y_{0:t} = \{y_i; i = 0, 1, \ldots, t\}$; marginal smoothing is concerned with the same problems for the quantity $p(x_t \mid y_{0:T})$, $t \leq T$; and pathwise smoothing is concerned with sampling or computing expectations with respect to the quantity $p(x_{0:T} \mid y_{0:T})$.

In many cases, the "true" generative model, consisting of the initial distribution p_0, the transition distributions p_t, and emission distributions h_t, is unknown, and one needs to estimate it from the observed data. A typical way is to assume parametric forms for $p_0(x_0 \mid \theta)$, $p_t(x_t \mid x_{t-1}, \theta)$, and $h_t(y_t \mid x_t, \theta)$, as well as a prior distribution $p(\theta)$ for the parameters, resulting in a joint distribution

$$p(x_{0:T}, y_{0:T}, \theta) \coloneqq p_0(x_0 \mid \theta) \left\{ \prod_{t=0}^{T} h_t(y_t \mid x_t, \theta) \right\} \left\{ \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}, \theta) \right\} p(\theta). \tag{2}$$

Under these notations, the parameter estimation problem then consists of computing either deterministic or probabilistic estimates of the posterior distribution over the parameters $p(\theta \mid y_{0:T})$. In this work, we will focus on computing probabilistic estimates for the pathwise smoothing distribution $p(x_{0:T} \mid y_{0:T})$ and the joint state-parameter posterior distribution $p(\theta, x_{0:T} \mid y_{0:T})$ (which marginally recovers $p(\theta \mid y_{0:T})$).

Throughout the rest of the article, 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 conditional distributions.

In this article, we consider a slight generalisation of (1), as given by the larger class of models

$$\pi(x_{0:T}) \propto g(x_0, x_1, \ldots, x_T)\, p_0(x_0) \left\{ \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}) \right\}. \tag{3}$$

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 (1) 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., 22]

$$\pi(x_{0:T}) \propto g_0(x_0)\, p_0(x_0) \left\{ \prod_{t=1}^{T} g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1}) \right\}, \tag{4}$$

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 [13, Ch. 5].

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 filters 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., 68, 4]. 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., 42], or on sigma-point linearisations, first introduced in [44, 77].

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 [5, 29, 74]. 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 [73] and [68, Ch. 10, 13, and 14].

Finally, it has been recently shown [67, 79, 80] that (extended/sigma-point) 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 [79, 80], where the operation needs to be repeated until eventual convergence of the smoothing solution. Markov chain Monte Carlo algorithms, which the present article is concerned with, are one such class of iterated methods.

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. A bias 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 [37] which we review next.

1.2. Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods [see 13] are alternatives to Gaussian-approximated posteriors which represent the filtering and smoothing distributions using Monte Carlo samples. They proceed by propagating the trajectory sequentially via an importance 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 [48]. 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 [35], and their resulting convergence improvements have been studied, for example in [25], and under a more general framework, in [20].

Importantly, because particle filtering provides an unbiased likelihood estimate, it can be used to perform asymptotically exact parameter and state estimation in state-space models. A particularly useful class of methods leveraging this property are the particle Markov chain Monte Carlo (pMCMC) methods [3, 1], which are based on constructing MCMC schemes either as a Metropolis–Rosenbluth–Teller–Hastings (MRTH) algorithm [56, 39], or a Gibbs-like sampler [31]. We refer to these as pseudo-marginal and particle Gibbs (pGibbs), respectively.

The aforementioned 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 [78, 53] 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 [21]. 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. 75]. Doing so, however, still fails as the dimension becomes larger.

Recently, [27] and [55, Ch. 4] independently proposed two related particle Gibbs algorithms that alleviate this issue by a generic localisation trick rather than approximation methods. [27] 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 [16] 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 [67, 79, 80].

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. 75], 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 SSM distributions but suffer from a 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 [27, Proposition 2.2] that is inherited from particle filtering in general.

In view of this, we develop general methods to perform 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 using techniques inherited from [67, 79, 80] in Gaussian-approximated case and from [16] in the particle Gibbs case.

These two approaches are respectively based on (i) [71] who design auxiliary MCMC gradient-based inference in high-dimensional latent Gaussian models, which we review in Section 2.1; (ii) [27] who 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 3.2 — consist in augmenting the target distribution π with auxiliary variables: using our SSM notation, $\pi(x_{0:T}, u_{0:T}) = \pi(x_{0:T}) \prod_{t=0}^{T} \mathcal{N}(u_t; x_t, \frac{\delta_t}{2}\Sigma_t)$ 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 [71] or a pGibbs kernel for a modified model [27].

Algorithm 1: Auxiliary MCMC

Result: An updated trajectory $x_{0:T}^{k+1}$
1 **Function** *AUX-MCMC*$(x_{0:T}^{k})$
2 Sample $u_{0:T}^{k} \sim \prod_{t=0}^{T} \mathcal{N}(u_t; x_t^k, \frac{\delta_t}{2}\Sigma_t)$
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** $x_{0:T}^{k+1}$

This perspective motivates our contributions outlined below.

1. In Section 2, we show that, in the case of generalised Feynman–Kac mod-

els (3) with Gaussian dynamics, the auxiliary proposals of [71] recover the posterior distribution of an auxiliary LGSSM. We leverage this to reduce their time and space complexity to $\mathcal{O}(T)$ rather than $\mathcal{O}(T^2)$. We then extend this to non-Gaussian prior dynamics using local Gaussian approximants. Furthermore, in Section 2.4, we introduce parallel-in-time samplers for the pathwise smoothing distribution of LGSSMs based on a prefix-sum implementation akin to [67], resulting in an overall $\mathcal{O}(\log T)$ MCMC algorithm on parallel hardware.

2. In Section 3, we describe how [27] is an instance of the auxiliary sampler. This novel perspective allows us to introduce novel, guided, auxiliary particle Gibbs methods by explicitly incorporating prior and gradient information in the form of locally optimal proposals for the auxiliary target model. Doing so improves on [27] in the highly-informative observation regime, but also reduces some of its drawbacks in the weakly-informative regime, essentially providing a more robust version of the method. Additionally, we discuss how this new perspective on [27] allows for the development of statistically efficient, gradient-informed parallel-in-time particle Gibbs samplers that can be efficiently implemented on GPUs.
3. In Section 4, we apply the proposed methods to perform inference on a multidimensional stochastic volatility model [27], a high-dimensional spatio-temporal model with fat-tailed observations taken from [19], and on a joint state-parameter inference problem for a non-linear stochastic differential equation [57]. Special attention is paid to understanding the statistical as well as computational trade-offs of our methods, in particular in terms of how the sequential and parallel counterparts of the methods (when they exist) compare. Finally, in Section 4.4, we highlight the respective failure modes, potential pitfalls, and limitations of the proposed methods, also providing a heuristic for explaining their performance in the other experiments.

2. Auxiliary Kalman samplers

In this section, we first review the auxiliary samplers of [71] 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 to reduce the time and memory complexity to linear in the number of time steps rather than quadratic. Finally, we discuss how linearisation methods can be used to extend the method to non-linear dynamics.

2.1. Auxiliary gradient-based samplers

Auxiliary gradient-based methods were introduced in [71] as a way to construct prior-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

[1] As well as, under a trivial change of variables, for models with non-zero prior mean.

shown to outperform classical pre-conditioned (prior-informed) and gradient-based (likelihood-informed) samplers, such as pre-conditioned Crank–Nicholson [18] or manifold MCMC methods [34] for latent Gaussian models. This impressive performance is both due to their better representation of the covariance of the posterior distribution [71, Section 3.4], and their computational advantage compared to classical methods, resulting in an improved effective sample size [ESS, see, e.g., 30, Ch. 11] per unit of time even when the effective sample size itself was lesser [71, Table 2].

Auxiliary gradient-based samplers rely on augmenting the target π with an auxiliary variable u:

$$\pi(x,u) \propto \exp(f(x))\,\mathcal{N}(x;0,C)\,\mathcal{N}\left(u;x,\frac{\delta}{2}I\right), \tag{5}$$

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

$$\begin{aligned} q(y \mid x,u) &\propto \exp\left(\nabla f(x)^\top y\right)\mathcal{N}(y;0,C)\,\mathcal{N}\left(u;y,\frac{\delta}{2}I\right) \\ &= \mathcal{N}\left(y;\frac{2}{\delta}A\left(u+\frac{\delta}{2}\nabla f(x)\right),A\right), \end{aligned} \tag{6}$$

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 [58]:

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 (6) with respect to $\mathcal{N}(u;x,\frac{\delta}{2}I)$:

$$q(y \mid x) = \mathcal{N}\left(y;\frac{2}{\delta}A\left(x+\frac{\delta}{2}\nabla f(x)\right),\frac{2}{\delta}A^2+A\right). \tag{7}$$

This marginalised version skips the intermediate sampling step of the auxiliary variable, and is provably better – both empirically and in terms of Peskun ordering [63, 70, 51] – than its auxiliary version. As a result, the marginalised version can use step sizes δ roughly twice as large [see Tables 1, 2, and 3 in 71] 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 $\delta > 0$, the matrices A and C share the same eigenspace [71, Section 3.3]. This ensures that, after an initial spectral decomposition of C, calibrating the value of δ (to achieve a target acceptance rate) can be done at a negligible cost compared to the actual sampling process itself. 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 $\pi(x,\theta) \propto \exp(f(x))\,\mathcal{N}(x;0,C_\theta)\,p(\theta)$, 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, costing $\mathcal{O}(d_x^3)$ operations in general. However, this can be mitigated thanks to the following observation [71]: under a reparametrisation of u, which corresponds to considering the augmented target

$$\pi(x,u) \propto \exp(f(x))\,\mathcal{N}(x;0,C)\,\mathcal{N}\left(u; x + \frac{\delta}{2}\nabla f(x), \frac{\delta}{2} I\right), \tag{8}$$

rather than (5), the proposal distribution $q(y \mid x,u)$ can be made independent of the current state of the chain x. This makes joint updates of x and θ in parametric models possible, 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 for non-parametric models.

In the remainder of this article, despite its statistical efficiency, we do not consider the marginalised proposal (7), and we consider the auxiliary sampler as defined in (5). This is because, as explained in the following section, (7) does not preserve the Markovian structure of our target models, making it computationally less efficient than the auxiliary samplers, which do. Similarly, we do not consider the empirically inferior reparametrised version (8) because its main advantage, namely that the resulting proposal, conditionally on the auxiliary variable, is independent of the current state of the chain [see 71, Section 3.3] does not extend directly to non-Gaussian priors, which we consider in the rest of this article. Nonetheless, for Gaussian prior dynamics, our methodology is directly compatible with (8) and can be used almost *mutatis mutandis* within our framework.[2]

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[3]:

$$\pi(x_{0:T}) \propto g(x_0,\dots,x_T)\,\mathcal{N}(x_0;m_0,P_0)\prod_{t=1}^{T}\mathcal{N}(x_t; F_{t-1}\,x_{t-1} + b_{t-1}, Q_{t-1}). \tag{9}$$

However, *directly* treating these as latent Gaussian models with the methods of [71] would incur a computational complexity of $\mathcal{O}(T^2 d_x^2)$, with an initial

[2]This parametrisation was also leveraged extensively in the follow-up work to the present article [17] treating of auxiliary pGibbs methods.

[3]This was in fact explicitly used in [13, Ch. 15], where the authors successfully apply [71] 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.

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. Nonetheless, as what pointed out by a reviewer, it is possible to reduce the computational complexity to linear in T by leveraging direct sparse Cholesky decompositions [see also 24, for such an approach], which are largely available on sequential hardware [see, e.g., 12]. Such sparse methods are however not as readily available on parallel hardware such as GPUs or TPUs [see, nonetheless 65, for a mixed CPU-GPU implementation achieving some speed-up, typically $\times 3$], and more plausible alternatives such as conjugate gradient methods [40] are not direct and require additional tuning. Instead of leveraging general sparse linear algebra techniques, it is possible, in the case of a model like (9), to directly formulate the auxiliary sampler as an LGSSM.

Remark 2.1. *Whithin the context of our work, this approach presents several advantages: (i) classical sequential [68] and parallel filtering and smoothing algorithms [67] can be applied almost* mutatis mutandis*, see Section 2.4, (ii) the formulation makes it easy to then extend the method to non-linear dynamics using linearisation techniques developed in the signal processing literature as discussed in Section 2.3, and (iii) the links with sequential Monte Carlo methods are more apparent, as we will see in Section 3.*

In order to formulate (6) as a LGSSM, we emulate [71] and consider the augmented target distribution

$$\pi(x_{0:T}, u_{0:T}) \propto \pi(x_{0:T}) \prod_{t=0}^{T} \mathcal{N}\left(u_t; x_t, \frac{\delta}{2}\Sigma_t\right), \tag{10}$$

where $\delta > 0$ and, for all $t = 0, \ldots, T$, 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 (5).

Let us define γ via $\exp(\gamma(x_0, x_1, \ldots, x_T)) := g(x_0, x_1, \ldots, x_T)$, 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

$$\begin{aligned} q(z_{0:T} \mid u_{0:T}, x_{0:T}) \propto \mathcal{N}(z_0; m_0, P_0) \left\{ \prod_{t=1}^{T} \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1}) \right\} \\ \left\{ \prod_{t=0}^{T} \mathcal{N}\left(u_t + \frac{\delta}{2}\Sigma_t v_t; z_t, \frac{\delta}{2}\Sigma_t\right) \right\}, \end{aligned} \tag{11}$$

which corresponds to the pathwise smoothing distribution of an LGSSM with unchanged dynamics compared to (9), and observations given by $u_t + \frac{\delta}{2}\Sigma_t v_t$ for an observation model $\mathcal{N}\left(\cdot; z_t, \frac{\delta}{2}\Sigma_t\right)$, $t = 0, 1, \ldots, T$. Sampling from this distribution, and evaluating its likelihood can be done using Kalman filtering and smoothing techniques in $\mathcal{O}(T)$ steps [see, e.g. 68, Ch. 6 and Ch. 12], [26,

Section 3.2], [13, Section 3.2], and Appendix A for more details. In fact, this representation is key to reducing 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 2.4.

To summarise, sampling from $\pi(x_{0:T}, u_{0:T})$ is then done via Hastings-within-Gibbs [58]: (i) sample $u^k_{0:T} \mid x^k_{0:T} \sim \prod_{t=0}^T \mathcal{N}(u_t; x^k_t, \frac{\delta}{2}\Sigma_t)$, (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 (iii) accept the move with the corresponding acceptance probability. We insist that this proposal is statistically equivalent to the auxiliary method of [71] for a choice of constant $\Sigma_t = I$, but exhibits better computational complexity than their implementation due to the LGSSM structure. Marginalising it over $u_{0:T}$, recovering (7), however, would destroy the proposal Markovian structure, removing this advantage completely. Similarly, in general, a second order approximations of γ would result in fully dependent observations, so that the proposal distribution would not correspond to a LGSSM anymore.

Nonetheless, 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

$$\gamma(z_{0:T}) \approx \gamma(x_{0:T}) + \langle v_{0:T}, z_{0:T} - x_{0:T} \rangle + \frac{1}{2} \sum_{t=0}^T (z_t - x_t)^\top \Lambda_t (z_t - x_t), \tag{12}$$

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

$$\begin{aligned} &q(z_{0:T} \mid u_{0:T}, x_{0:T}) \\ &\quad \propto \mathcal{N}(z_0; m_0, P_0) \left\{ \prod_{t=1}^T \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1}) \right\} \left\{ \prod_{t=0}^T \mathcal{N}\left(\omega_t; z_t, \Omega_t\right) \right\}, \end{aligned} \tag{13}$$

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 and will recover the exact auxiliary target when the original model $\pi(x_{0:T})$ is Gaussian.

Finally, when the dynamics are not Gaussian, it is often possible to transform the model at hand into an equivalent representation of $\pi(x_{0:T})$ with Gaussian dynamics by setting

$$\begin{aligned} &p_0(x_0) \leftarrow \mathcal{N}(x_0; m_0, P_0), \quad p_t(x_t \mid x_{t-1}) \leftarrow \mathcal{N}(x_t; F_{t-1} x_{t-1} + b_{t-1}, Q_{t-1}), \\ &g(x_{0:T}) \leftarrow g(x_{0:T}) \frac{p_0(x_0)}{\mathcal{N}(x_0; m_0, P_0)} \prod_{t=1}^T \frac{p_t(x_t \mid x_{t-1})}{\mathcal{N}(x_t; F_{t-1} x_{t-1} + b_{t-1}, Q_{t-1})}, \end{aligned} \tag{14}$$

for a choice of m_0, P_0, F_{t-1}, b_{t-1}, and Q_{t-1}, enabling the use of the auxiliary sampler (11). While this is sometimes a natural thing to do [see, e.g., 47, for an

application swapping a reflected Brownian motion prior for a standard Brownian motion one], 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.

Remark 2.2. *As pointed out by a reviewer, when using a second-order approximation of the potential, the resulting proposal distribution is agnostic to the choice of the Gaussian prior. This is a direct consequence of the fact that the second-order approximation of the potential is a quadratic function, and that the resulting proposal distribution is a Gaussian distribution. As a consequence, while there is sometimes no natural choice for introducing a Gaussian prior in the model when using first-order linearisation, all such choices are equivalent when using a second-order approximation and will only affect the computational aspects of the algorithm. However, it is not plausible that the second-order approximation would result in a Markovian structure for the latent variables, making this approach highly inefficient in practice. Nonetheless, it may be possible to derive practical Hessian approximations that preserve Markovianity by construction, hopefully offering another approach to designing efficient auxiliary samplers for non-linear non-Gaussian models. We leave this as an open question for future research.*

2.3. New auxiliary samplers for models with non-Gaussian dynamics

In Section 2.2, we have made an explicit link between the auxiliary samplers of [71] 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, akin to extended Kalman linearisation [see, e.g., 68, Ch. 7].

Let us assume that our target distribution is given by

$$\pi(x_{0:T}) \propto p_0(x_0) \left\{ \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}) \right\} g(x_{0:T}), \tag{15}$$

where the latent dynamics model $p_0(x_0) \left\{ \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}) \right\}$ is not necessarily Gaussian anymore. Similarly as in Section 2.2, we can form the augmented target distribution

$$\pi(x_{0:T}, u_{0:T}) := p_0(x_0) \left\{ \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}) \right\} g(x_{0:T}) \left\{ \prod_{t=0}^{T} \mathcal{N}(u_t; x_t, \frac{\delta}{2}\Sigma_t) \right\} \tag{16}$$

where $\delta > 0$, and for all t, Σ_t is a positive definite matrix.

In order to form a proposal distribution $q(z_{0:T} \mid u_{0:T}, x_{0:T})$ for $\pi(x_{0:T} \mid u_{0:T})$, we can first linearise the potential function

$$g(x_{0:T}) \approx \exp\left\{\gamma(x_{0:T}) + \sum_{t=0}^{T} \left(v_t^\top (z_t - x_t)\right)\right\} \tag{17}$$

around $x_{0:T}$, for $v_t = \nabla_{x_t}\gamma(x_{0:T})$ as in Section 2.2, forming the intermediary (intractable in general) proposal distribution

$$\begin{aligned}\tilde{q}(z_{0:T} \mid u_{0:T}, v_{0:T}, x_{0:T}) \coloneqq & p_0(z_0)\left\{\prod_{t=1}^{T} p_t(z_t \mid z_{t-1})\right\} g(x_{0:T}) \\ & \left\{\prod_{t=0}^{T} \mathcal{N}\left(u_t + \frac{\delta}{2}\Sigma_t v_t; z_t, \frac{\delta}{2}\Sigma_t\right)\right\},\end{aligned} \tag{18}$$

where, contrary to (11) we make the dependency on $v_{0:T}$ explicit despite the redundancy with $x_{0:T}$ at this specific stage.

This can then be further approximated by forming a linear Gaussian approximation to the dynamics model $p_0(z_0)\left\{\prod_{t=1}^{T} p_t(z_t \mid z_{t-1})\right\}$, whereby we can approximate $p_0(z_0) \approx \mathcal{N}(z_0; m_0, P_0)$, via its first two moments, and $p_t(x_t \mid x_{t-1}) \approx \mathcal{N}(z_t; F_{t-1}z_{t-1} + b_{t-1}, Q_{t-1})$ for $t = 1, \ldots, T$.

In principle, the latter approximation can, for example, be obtained by minimising the Kullback–Leibler [KL, 49] divergence between the true and the approximated transition model

$$\mathrm{KL}(\mathcal{N}(z_t; F_{t-1}z_{t-1} + b_{t-1}, Q_{t-1}) \parallel p_t(z_t \mid z_{t-1})) \tag{19}$$

as a function of F_{t-1}, b_{t-1}, and Q_{t-1}. However, the optimal solution to this problem will in general depend on the value of z_{t-1} and is therefore not a well-defined problem. Instead, we can minimise the *expected* KL divergence with respect to a reference random variable distributed as $\mathcal{N}(x_{t-1}, \Gamma_{t-1})$, centred on the current state x_{t-1} and with a user-chosen covariance matrix Γ_{t-1}. This leads to the generalised statistical linear regression (GSLR) framework of [74] which we review in Appendix B. In practice, the solution to the KL minimisation problem (19) recovers classical state-space model linearisation techniques [for a review of these, we refer to 68] such as the extended Kalman filter, which we detail in Example 2.1, but also allows for more sophisticated approximations.

Example 2.1. *Suppose that the latent dynamics model has additive noise, that is, it is given by $X_t = f(X_{t-1}) + \epsilon_{t-1}$, where f is a smooth function and ϵ_{t-1} is a centred Gaussian noise term with covariance Q_{t-1}. Clearly, $p_t(z_t \mid z_{t-1})$ is then* conditionally *Gaussian, with mean $f(z_{t-1})$ and covariance Q_{t-1}. For a given z_{t-1}, we then compute the KL divergence (19) as*

$$\begin{aligned}&\mathrm{KL}(\mathcal{N}(z_t; F_{t-1}z_{t-1} + b_{t-1}, Q_{t-1}) \parallel \mathcal{N}(z_t; f(z_{t-1}), Q_{t-1})) \\ &= \mathrm{Cte} + \frac{1}{2}\left(F_{t-1}z_{t-1} + b_{t-1} - f(z_{t-1})\right)^\top Q_{t-1}\left(F_{t-1}z_{t-1} + b_{t-1} - f(z_{t-1})\right)\end{aligned} \tag{20}$$

where Cte *is a constant that does not depend on* F_{t-1}, b_{t-1}, *or* Q_{t-1}. *A first order linearisation of* f *around* x_{t-1} *of the right-hand side of* (20) *then gives the approximation* $F_{t-1} \approx \nabla f(x_{t-1})$ *and* $b_{t-1} \approx f(x_{t-1}) - F_{t-1}x_{t-1}$, *independent of the choice of* Γ_{t-1}.

These linear approximations, together with the known (or approximated) first two moments m_0 and P_0 of $p_0(x_0)$, can then be used to form a proposal distribution defined as an auxiliary LGSSM smoothing distribution with density

$$q(z_{0:T} \mid u_{0:T}, v_{0:T}, x_{0:T}) \propto \mathcal{N}(z_0; m_0, P_0) \left\{ \prod_{t=0}^{T} \mathcal{N}\left(u_t + \frac{\delta}{2}\Sigma_t v_t; z_t, \frac{\delta}{2}\Sigma_t\right) \right\} \left\{ \prod_{t=1}^{T} \mathcal{N}(z_t; F_{t-1}z_{t-1} + b_{t-1}, Q_{t-1}) \right\}. \tag{21}$$

This proposal distribution is then included as part of a Metropolis–Rosenbluth–Teller–Hastings (MRTH) acceptance-rejection step. The resulting sampler corresponds to Algorithm 2. Evaluating the augmented density (10) appearing in

Algorithm 2: General Auxiliary Kalman sampler

Result: An updated trajectory $z_{0:T}$
1 **Function** *AuxKalmanSampler*$(x_{0:T})$
  `// Generate the auxiliary observations`
2 **for** $t = 0, 1, \ldots, T$ **sample** $u_t \mid x_t \sim \mathcal{N}(\cdot; x_t, \frac{\delta}{2}\Sigma_t)$
  `// Form the proposal` $q(\cdot \mid u_{0:T}, v_{0:T}, x_{0:T})$ `in` (21)
3 **for** $t = 0 \ldots, T$ **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 Set $v_t = \nabla_{x_t} \gamma(x_{0:T})$
7 Sample $x^*_{0:T} \sim q(\cdot \mid u_{0:T}, v_{0:T}, x_{0:T})$ and compute $L^* = \frac{q(x^*_{0:T}, u_{0:T}, v_{0:T} \mid x_{0:T})}{q(u_{0:T}, v_{0:T} \mid x_{0:T})}$
  `// MRTH step`
8 Form the reversed proposal $q^*(x_{0:T} \mid u_{0:T}, v^*_{0:T}, x^*_{0:T})$ following steps 5 and 6 around $x^*_{0:T}$ and compute $L = \frac{q^*(x_{0:T}, v^*_{0:T}, u_{0:T} \mid x^*_{0:T})}{q^*(v^*_{0:T}, u_{0:T} \mid x^*_{0:T})}$
9 With probability $\min\left(1, \frac{p(x^*_{0:T}, u_{0:T})L}{p(x_{0:T}, u_{0:T})L^*}\right)$, set $z_{0:T} = x^*_{0:T}$
10 Otherwise, set $z_{0:T} = x_{0:T}$
11 **return** $z_{0:T}$

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 $x^*_{0:T} \sim q(\cdot \mid u_{0:T}, v_{0:T}, x_{0:T})$ of the LGSSM at hand, and compute the corresponding smoothing density $q(x^*_{0:T}, v_{0:T}, u_{0:T} \mid x_{0:T})/q(v_{0:T}, u_{0:T} \mid x_{0:T})$. We come back to this point in Section 2.4.

We end this section by noting that, while we assumed that we had linearised the potential g prior to finding an approximation to the dynamics, the two tasks

can be tackled simultaneously. This is particularly useful when the potential is obtained as a product of observation models $h(y_t \mid x_t)$, as in the case of state-space models, for which we are able to compute approximations

$$h(y_t \mid z_t) \approx \mathcal{N}(y_t; H_t z_t + c_t, R_t), \tag{22}$$

around x_t for all t. In this case, we can apply exactly the same linearisation procedure to the observation model as we did to the dynamics model, form the proposal distribution (21) by combining the two linear approximations into a proposal model

$$q(z_{0:T} \mid u_{0:T}, y_{0:T}, x_{0:T}) \propto \mathcal{N}(z_0; m_0, P_0) \left\{ \prod_{t=1}^{T} \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1}) \right\} \left\{ \prod_{t=0}^{T} \mathcal{N}\left(u_t; z_t, \frac{\delta}{2}\Sigma_t\right) \right\} \left\{ \prod_{t=0}^{T} \mathcal{N}(y_t; H_t z_t + c_t, R_t) \right\}, \tag{23}$$

and then proceed almost identically to Algorithm 2. Forming such approximations is described in more detail in Appendix B.

2.4. Sampling and evaluating the posterior of LGSSMs

In the previous sections, we described a new auxiliary-variable-based MCMC algorithm for Markovian models, which, after a choice of linearisation, amounts to sampling from a linear Gaussian state-space model $q(z_{0:T} \mid u_{0:T}, x_{0:T})$ depending on the current state of the chain $x_{0:T}$ and the auxiliary variables $u_{0:T}$ and then accepting the move with the corresponding acceptance probability within a MRTH step. To use it within Algorithm 1, we therefore only need to understand how to sample from the proposal distribution $q(z_{0:T} \mid u_{0:T}, x_{0:T})$ and evaluate it. Thankfully, the resulting distribution is the posterior distribution of an LGSSM, for which efficient sampling and evaluation methods exist [see, e.g., 4, 68, for a comprehensive treatment of the topic]. In this section, we quickly review the classical forward-filtering backward-sampling algorithm for LGSSMs, which, when implemented on sequential hardware, has a time and complexity of $\mathcal{O}(T)$. We then explain how this can be improved to $\mathcal{O}(\log T)$ using either prefix-sum algorithms [see, e.g., 9] or divide-and-conquer strategies. More details on the different methods, including implementation details, are provided in Appendix A.

2.4.1. Forward-filtering backward-sampling for LGSSMs

The forward-filtering backward-sampling [11, 28, FFBS,] algorithm is a classical method to sample from the posterior distribution of an LGSSM. Given a state-space model

$$p(x_{0:T}, y_{0:T}) = p_0(x_0) \prod_{t=1}^{T} p_t(x_t \mid x_{t-1}) h_t(y_t \mid x_t), \tag{24}$$

as in (1), we can compute the filtering densities $p(x_t \mid y_{0:t})$ recursively as

$$\begin{aligned} p(x_t \mid y_{0:t}) &\propto h_t(y_t \mid x_t) \int p_t(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{0:t-1}) \mathrm{d}x_{t-1}, \quad t = 1, \ldots, T, \\ p(x_0 \mid y_0) &\propto h_0(y_0 \mid x_0) p_0(x_0). \end{aligned} \tag{25}$$

When the initial distribution, transition and observation models are linear Gaussian, the quantities in (25) are Gaussian too and can be computed in closed form. Moreover, this recursion, as a by-product, also computes the marginal likelihood of the observations $p(y_{0:T})$ via

$$p(y_{0:T}) = p(y_0) \prod_{t=1}^{T} p(y_t \mid y_{0:t-1}) = p(y_0) \prod_{t=1}^{T} \int h_t(y_t \mid x_t) p(x_t \mid y_{0:t-1}) \mathrm{d}x_t \tag{26}$$

where each term $p(y_t \mid y_{0:t-1})$ (resp. $p(y_0)$) is the normalisation constant of the filtering density $p(x_t \mid y_{0:t})$ with respect to $p(x_t \mid y_{0:t-1})$ (resp. $p(x_0 \mid y_0)$ with respect to $p_0(x_0)$).

Once all the filtering densities have been computed, the backward sampling step consists in sampling from the conditional distribution $p(x_{0:T} \mid y_{0:T})$ recursively as

$$\begin{aligned} x_T &\sim p(x_T \mid y_{0:T}), \\ x_t &\sim p(x_t \mid x_{t+1}, y_{0:T}), \quad t = T-1, \ldots, 0, \end{aligned} \tag{27}$$

noting that

$$p(x_t \mid x_{t+1}, y_{0:T}) = p(x_t \mid x_{t+1}, y_{0:t}) \propto p(x_{t+1} \mid x_t) p(x_t \mid y_{0:t}), \tag{28}$$

which, under the same hypothesis as above, is Gaussian and can be computed in closed form. Given that

$$\begin{aligned} q(z_{0:T} \mid u_{0:T}, v_{0:T}, x_{0:T}) \propto \mathcal{N}(z_0; m_0, P_0) &\left\{ \prod_{t=0}^{T} \mathcal{N}\left(u_t + \frac{\delta}{2}\Sigma_t v_t; z_t, \frac{\delta}{2}\Sigma_t\right) \right\} \\ &\left\{ \prod_{t=1}^{T} \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1}) \right\}, \end{aligned} \tag{29}$$

introduced in (21), is the posterior distribution of an LGSSM with observations $y_t = u_t + \frac{\delta}{2}\Sigma_t v_t$, we can therefore sample from it using the decomposition above, and evaluate the marginal likelihood $q(u_{0:T}, v_{0:T} \mid x_{0:T})$ using (26). This offers a solution to also compute $q(x^*_{0:T} \mid u_{0:T}, v_{0:T}, x_{0:T}) = q(x^*_{0:T}, u_{0:T}, v_{0:T} \mid x_{0:T}) / q(u_{0:T}, v_{0:T} \mid x_{0:T})$ appearing in Algorithm 2. The same applies to all other instances of the method we presented above.

2.4.2. Parallel-in-time sampling of LGSSMs

Due to its recursive structure, the method described in Section 2.4.1 has a time complexity of $\mathcal{O}(T)$, which can be prohibitive for large values of T. While this complexity is optimal on sequential hardware, where the computation of the filtering densities has to be done sequentially, it can be improved to $\mathcal{O}(\log T)$ on parallel hardware, such as GPUs or TPUs. In this section, we describe two methods to achieve this: a prefix-sum approach and a divide-and-conquer approach, which can be used to sample from the proposal distribution $q(z_{0:T} \mid u_{0:T}, x_{0:T})$ in Algorithm 1 in $\mathcal{O}(\log T)$ time. Both algorithms rely on first computing the filtering densities $p(x_t \mid y_{0:t})$ in $\mathcal{O}(\log T)$, which, when the model is linear Gaussian, can be done using [67].

In order to simplify the description of these methods, we assume that the backward distribution $p(x_t \mid x_{t+1}, y_{0:T}) = \mathcal{N}(x_t; E_t x_{t+1} + f_t, L_t)$ has a linear Gaussian form, which is the case for the LGSSM model (24), and that the coefficients E_t, f_t, and L_t have been precomputed and are available for sampling in $\mathcal{O}(1)$ time. Further details on the prefix-sum and divide-and-conquer approaches, including a review of the filtering densities computation of [67], and the formulation of the backward distribution parameters E_t, f_t, and L_t, are provided in Appendix A.

Prefix-sum approach to parallel FFBS Prefix-sum algorithms are a class of parallel algorithms that compute the cumulative "sum" of an array of elements in logarithmic time under sufficient parallelism. Formally, given a sequence of elements $x_0, x_1, \ldots, x_{T-1}$ and an operator $\oplus$, the prefix sum of the sequence is the sequence $y_0, y_1, \ldots, y_{T-1}$ such that $y_t = \bigoplus_{i=0}^{t} x_i$. Provided that the operator $\oplus$ is associative, the prefix sum can be computed in $\mathcal{O}(\log T)$ time using $\mathcal{O}(T)$ processors.

Suppose now that we have access to X_{t+1}. Then, to sample from $X_t \mid \{X_{t+1}, y_{0:T}\}$, we can sample from the Gaussian noise $X_t \sim \mathcal{N}(0, L_t)$ and compute $X_t \leftarrow E_t X_{t+1} + f_t + X_t$. However, this operation, as seen as an operation on X_t, X_{t+1} only, is not associative, and we cannot apply the prefix sum directly. On the other hand, the same method can be seen as an operation on the quadruplet (E_t, f_t, L_t, X_{t+1}) via the operator $\circ$ defined as

$$\begin{aligned}&(E_{t-1}, f_{t-1}, L_{t-1}, X_t) \circ (E_t, f_t, L_t, X_{t+1}) \\ &= (E_{t-1}E_t,\ E_{t-1}f_{t-1} + f_t,\ E_{t-1}L_t E_{t-1}^\top + L_{t-1},\ X_t + E_t X_{t+1} + f_t).\end{aligned} \tag{30}$$

Because it collects the composition of the transition matrices E_t, the offset vectors f_t, and the transition covariance L_t in a single operation, the operator $\circ$ is associative, and we can apply the prefix sum to the sequence (E_t, f_t, L_t, X_{t+1}) to sample from $X_{0:T} \mid y_{0:T}$ in $\mathcal{O}(\log T)$ time. A formal statement of this result, as well as a more efficient implementation of the algorithm, relying on propagating only E_t and X_t, are provided in Appendix A.2.2.

Divide-and-conquer approach to parallel FFBS Another approach to parallelise the FFBS algorithm is to use a divide-and-conquer strategy, which

consists in computing bridging distributions between time steps in a hierarchical manner. Formally, we can recursively sample the distribution $p(x_{0:T} \mid y_{0:T})$ by

1. first sampling from $p(x_T \mid y_{0:T})$ and $p(x_0 \mid x_T, y_{0:T})$,
2. then $p(x_{\lfloor T/2 \rfloor} \mid x_0, x_T, y_{0:T})$,
3. then $p(x_{\lfloor T/4 \rfloor} \mid x_0, x_{\lfloor T/2 \rfloor}, y_{0:T})$ and $p(x_{\lfloor T/4 \rfloor} \mid x_{\lfloor T/2 \rfloor}, x_T, y_{0:T})$,
4. and so on,

until we exhaust all the time steps.

Because, at each level k in the recursion (apart from the first one), we sample from 2^k distributions in parallel, the total number of non-parallel steps is $\mathcal{O}(\log T)$, and the divide-and-conquer approach therefore has a time complexity of $\mathcal{O}(\log T)$ on parallel hardware. Of course, if implemented naively, this approach would require computing the bridging distributions $p(x_s \mid x_l, x_u, y_{0:T})$ for all $l < s < u$ at each level of the recursion, which would be computationally prohibitive. Instead, it is possible to compute these via an initial reversed recursion, whereby $p(x_t \mid x_{t+1}, y_{0:T})$ is initialised as $\mathcal{N}(x_t; E_t x_{t+1} + f_t, L_t)$, which then allows to compute $p(x_t \mid x_{t+2^0}, x_{t-2^0}, y_{0:T})$, then $p(x_t \mid x_{t+2^1}, x_{t-2^1}, y_{0:T})$, and so on, for all times t that appear at level $\lfloor \log T \rfloor - k$ of the recursion. This method, as well as a description of how to efficiently compute the bridging distributions $p(x_s \mid x_l, x_u, y_{0:T})$ arising in the recursion above, are provided in Appendix A.2.3.

3. Auxiliary particle Gibbs samplers

We have so far been concerned with designing global MCMC proposals that leveraged local LGSSM approximations of the target distribution. These proposals, while expected to work particularly well when the prior is almost Gaussian and the potential relatively non-informative, nonetheless constitute "global acceptance methods", and, as such, present at least two limitations:

1. Because they accept or reject a full trajectory at once, a single unfortunate proposed time-step can lead to a rejection of the whole trajectory, even if the rest of the trajectory is correct. In other terms, the method is not robust to heterogenously informative observations.
2. Even though the method is efficient when the prior is informative, it will still collapse when the number of time steps goes to infinity.

Remark 3.1. *While intuitive, the second limitation can be formalised in the case when the model is fully separable, that is, when* $\pi(x_{0:T}) = \prod_{t=0}^{T} p(x_t) g(x_t)$ *for some Gaussian prior* p *and likelihood function* g*. In this case, the acceptance probability of the Kalman-based MCMC kernel will be the product of the acceptance probabilities of the individual time steps and therefore, the acceptance probability of the full trajectory will go to zero as* T *goes to infinity unless* δ *decreases to zero as well. Understanding the exact rate at which* δ *should in general decrease to zero is a difficult problem, one that is not addressed in this article. Nonetheless, when* $p \equiv 1$ *is an improper prior, it can be seen that [71]*

recovers the MALA algorithm [6] and therefore, one can expect that the results of [66] correspond to a worst-case scenario for the method, i.e., δ should then decrease to zero at a rate of $\mathcal{O}(1/T^{1/3})$ at most.

For the above reasons, in this section, we turn ourselves to the successful class of particle MCMC algorithms, in particular particle Gibbs algorithms [1], which have been shown to be very robust to increasingly many time steps T [50, 46], and show how the same auxiliary observation trick can be leveraged to design efficient particle MCMC samplers for Feynman–Kac models. Intuitively, these will be more robust to the highly informative observations as they essentially form *local* MCMC moves [27, Section 2.2]. For instance, for the degenerate case of fully separable models, factorising in time as in Remark 3.1, "trajectories" will be accepted independently of each other, and the algorithm will be less affected by the presence of a single bad time-step.

In the remainder of this section, we first quickly recall the basic particle Gibbs algorithm, and a recent high-dimensional extension due to [27]. We then show how [27] can be understood as an instance of a more general method, relying on a similar auxiliary observation trick as the one used in Section 2. This novel perspective then allows us to introduce novel auxiliary particle Gibbs methods, extending [27] to incorporate prior and gradient information in the form of "locally optimal proposals" [also called guided proposals in 13, Ch. 16], and discuss when these can be parallelised efficiently on GPUs along the time dimension, similarly to the methods of Section 2.4.

3.1. SMC and particle Gibbs algorithms

Particle Gibbs algorithms are Gibbs-like MCMC samplers that target the posterior distribution of Feynman–Kac models [1, 54, 53]. 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 under the weak hypothesis that the potential functions are bounded above [see 50, 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 the density $p_t(x_t \mid x_{t-1})$ as a function of x_t and x_{t-1}, we can modify the representation of the Feynman–Kac model as

$$\pi(x_{0:T}) \propto \tilde{g}_0(x_0)\, \tilde{p}_0(x_0) \left\{\prod_{t=1}^{T} \tilde{g}_t(x_t, x_{t-1})\, \tilde{p}_t(x_t \mid x_{t-1})\right\} \tag{31}$$

provided that the identity $\tilde{g}_t = \frac{g_t \cdot p_t}{\tilde{p}_t}$ holds for all $t = 0, 1, \ldots, T$, in which case p_t and g_t can be replaced by $\tilde{p}_t$ and $\tilde{g}_t$ in Algorithm 3 while keeping the same

Algorithm 3: Conditional SMC

Result: An updated trajectory $z_{0:T}$

1 **Function** $cSMC(x_{0:T},\ N)$

`// Forward propagation`

2 **for** $n = 1, 2, \ldots, N-1$ **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** $t = 1, \ldots, T$ **do**

6 **for** $n = 1, \ldots, N-1$ **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 $w_t^n = g_t(X_t^n, X_{t-1}^{A_t^n})$

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 $z_T = X_T^{B_T}$

11 **for** $t = T-1, \ldots, 0$ **do**

12 Set $B_t = A_{t+1}^{B_{t+1}}$, $z_t = X_t^{B_t}$

13 **return** $z_{0:T}$

posterior target $\pi(x_{0:T})$ invariant. This key property will be used extensively in the remainder of this section.

Additionally, when the density p_t can be evaluated, we can also rejuvenate the selection of the genealogy (step 9 in Algorithm 3), allowing for lower degeneracy in the early time steps. The most notable two such methods are the backward and ancestor sampling methods [78, 53, respectively]. The former [78], in particular, has been the subject of much interest in the literature, and has recently been shown to improve the mixing of the algorithm from $\mathcal{O}(T)$ to $\mathcal{O}(\log T)$ in the number of time steps [46]. In other terms, when implemented with the backward sampling method, the resulting Markov chain will require $\mathcal{O}(\log T)$ iterations to achieve stationarity, while the 'naive' version of Algorithm 3 will require $\mathcal{O}(T)$ iterations. The full implementation is given in Algorithm 11 in Appendix C.1. Another method, useful in our context, is that of [16, Section 3], which implements a parallel-in-time conditional SMC, particularly amenable to when the proposal/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 samplers in this article: for example the sampler presented next in Algorithm 4, see Section 3.2 for more details.

While widely used in practice, particle Gibbs algorithms suffer from degeneracy inherent to importance sampling methods when the dimension of the latent space increases. In order to counteract this issue, several methods have been proposed, such as using spatial blocking [69] or divide-and-conquer strategies [19], but these are not always applicable as they require a specific structure in the model and can be complicated to implement and tune. Recently, [27] proposed a localised cSMC algorithm, recognising that the degeneracy of the particle filter came from the fact that the proposals used therein (line 8 of Algorithm 3) did not depend on the current state of the Markov chain $x_{0:T}$; a property that can be understood as it generalising the Metropolis–Hastings algorithm for inde-

pendent proposals. From this observation, they proposed to modify the cSMC algorithm to emulate the random walk Metropolis–Hastings algorithm, by using a proposal that depends on the current state of the chain. We summarise this approach in Algorithm 4 (RW-cSMC).

Algorithm 4: Random-walk Conditional SMC

Result: An updated trajectory $x_{0:T}$
Function *RW-cSMC*$(x_{0:T}, N, \lambda_{0:T} \in \mathbb{R}^{T+1})$
  `// Forward propagation`
  for $t = 0, \dots, T$ **do**
    for $n = 1, \dots, N-1$ **do**
      Sample $U_t \sim \mathcal{N}(x_t, \frac{\delta_t}{2} I)$ then $X_t^n \sim \mathcal{N}(U_t, \frac{\delta_t}{2} I)$
    Set $X_t^{N+1} = x_t$
  for $n = 1, \dots, N$ **do**
    Set $w_0^n = g_0(X_0^n) p_0(X_0^n)$
  for $t = 1, \dots, T$ **do**
    for $n = 1, \dots, N-1$ **do**
      Sample A_t^n with $\mathbb{P}(A_t^n = k) \propto w_{t-1}^k$
      Set $w_t^n = g_t(X_t^n, X_{t-1}^{A_t^n}) p_t(X_t^n \mid X_{t-1}^{A_t^n})$
    Set $X_t^N = x_t$, $w_t^N = g_t(x_t, x_{t-1}) p_t(x_t \mid x_{t-1})$
  `// The genealogy selection is left unchanged compared to Algorithm 3`

This algorithm takes its name from the fact that it generalises the Gaussian random-walk Metropolis–Hastings (RWMH) algorithm to more than a single time step. Indeed, given the current state of the chain x_t at time t, the proposed particles $X_t^{1:N}$ are all marginally distributed as $\mathcal{N}(x_t, \delta_t I)$, and the acceptance probability of the proposal is given by the ratio of the g_t of the current and proposed states, which is symmetric in the current and proposed states. Importantly, RW-cSMC exhibits a similar asymptotic scaling as RWMH in terms of the dimension of the state-space, and a similar scaling as cSMC in terms of the time dimension, making it a good candidate for high-dimensional state-space models. See [27] for quantitative details and different instances of the algorithm. In the following section, we offer another interpretation of Algorithm 4 in terms of an auxiliary variable sampler, better suited to extensions.

3.2. *Particle Gibbs for Feynman–Kac models with auxiliary observations*

For the class of Feynman–Kac models (4), we can emulate the construction of Section 2 to form the following auxiliary target

$$\pi(x_{0:T}, u_{0:T}) \propto g_0(x_0)\, p_0(x_0) \left\{ \prod_{t=1}^{T} g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1}) \right\} \left\{ \prod_{t=0}^{T} \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2} \Sigma_t\right) \right\}, \tag{32}$$

corresponding to a model with an augmented potential function $g_t(x_t, x_{t-1}) \times \mathcal{N}(u_t; x_t, \frac{\delta_t}{2}\Sigma_t)$ at each time step t. In order to sample from $\pi(x_{0:T}, u_{0:T})$, it is, therefore, enough to implement an abstract algorithm given by Algorithm 5.

Algorithm 5: Auxiliary cSMC

Result: An updated trajectory $x_{0:T}^{k+1}$

1 **Function** AUX-CSMC$(x_{0:T}^k, u_{0:T}^k)$

2 Sample $u_{0:T}^{k+1} \sim \prod_{t=0}^{T} \mathcal{N}(u_t; x_t^k, \frac{\delta_t}{2}\Sigma_t)$

3 Sample $x_{0:T}^{k+1} \sim K(\cdot \mid x_{0:T}^k)$ `// from a` $\pi(x_{0:T} \mid u_{0:T}^{k+1})$`-invariant cSMC kernel`

4 **return** $x_{0:T}^{k+1}, u_{0:T}^{k+1}$

Clearly, in Algorithm 5, 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 $\pi(u_{0:T}) \equiv 1$ for the auxiliary variables.

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 $g_t(x_t, x_{t-1})\mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right)$ at each time step t. This new potential function becomes very informative as δ_t gets smaller, which is known to induce high variance weights in particle filtering and smoothing algorithms [see, e.g. 13, Section 10.3.1] akin to increasing the dimension. However, rather than seeing $\mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right)$ as describing an auxiliary observation, we can leverage the symmetry of Gaussian distributions to look at it as the generative model $\mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}\Sigma_t\right)$ instead.

Namely, we can swap the roles of $p_t(x_t \mid x_{t-1})$ and $\mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right)$ in the auxiliary Feynman–Kac model (32) to obtain the following modified model

$$\pi(x_{0:T} \mid u_{0:T}) \propto \tilde{p}_0(x_0 \mid u_0) \left\{\prod_{t=1}^{T} \tilde{p}_t(x_t \mid x_{t-1}, u_t)\right\} \tilde{g}_0(x_0) \left\{\prod_{t=1}^{T} \tilde{g}_t(x_t, x_{t-1})\right\}, \tag{33}$$

for the modified dynamics $\tilde{p}_t(x_t \mid x_{t-1}, u_t) = \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}\Sigma_t\right)$ and potential functions $\tilde{g}_t = g_t \cdot p_t$, $t = 0, 1, \ldots, T$.

Remark 3.2. *This procedure amounts to moving the auxiliary likelihood* $\mathcal{N}(u_t; x_t, \frac{\delta_t}{2}\Sigma_t)$ *from the potential function to the dynamics, and the "true" dynamics* $p_t(x_t \mid x_{t-1})$ *to the potential function, which is a common principle we leverage in all the methods we propose in this section.*

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 5 via Algorithm 3. Moreover, because the

auxiliary prior model is separable across time, the method of [16] applies directly[4], and a parallel-in-time particle Gibbs can be implemented to reduce the computational complexity to $\mathcal{O}(\log T)$ on parallel hardware, the construction of which we describe in Appendix C. We also note that, contrarily to [16, 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 the same independent proposals.

In hindsight, it is easy to see that, when $\Sigma_t = I$, $t = 0, \dots, T$, this method is *exactly* the same one as the one proposed in [27, Algorithm 3 and extensions] who instead phrase it as a form of conditional SMC with exchangeable proposals. Informally, rather than independently proposing the particles $x_t^{1:N}$ from $p_t(x_t \mid x_{t-1})$, they use a correlated proposal $\tilde{p}_t(x_t^1, \dots, x_t^N)$ which induces an exchangeable dependency across particles, that is, $\tilde{p}_t(x_t^1, \dots, x_t^N) = \tilde{p}_t(x_t^{\sigma(1)}, \dots, x_t^{\sigma(N)})$ for any permutation σ. As done in [27], and first introduced in the context of classical MCMC in [72], in the case of Gaussian variables, taking a conditional sample $p(\dots, x_t^{k-1}, x_t^{k+1}, \dots \mid x_t^k)$ 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 the proposal and weighting mechanism of Algorithm 5 for the modified Feynman–Kac model (33) and justifies the following proposition.

Proposition 3.1. *The method of [27], given in Algorithm 4, implements Algorithm 5 with* $u_t^{k+1} \sim \mathcal{N}\left(u_t; x_t^k, \frac{\delta_t}{2} I\right)$ *and proposal distributions* $\tilde{p}(x_t \mid x_{t-1}) \sim \mathcal{N}\left(\cdot; u_t^{k+1}, \frac{\delta_t}{2} I\right)$ *for different choices of kernels* K*: embedded HMM [60], conditional SMC [1], conditional SMC with forced move [14], and conditional SMC with backward sampling [78, see also Algorithm 3].*

In other terms, the results of [27] apply too, and for a given choice of a standard conditional SMC – with and without backward sampling – Algorithm 5 for the proposals $\tilde{p}(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; u_t^{k+1}, \frac{\delta_t}{2} I\right)$ avoids the curse of dimensionality. Formally, under a scaling $\delta = \mathcal{O}(1/d_x)$, it is stable for increasingly large d_x (as well as T) [for details and assumptions, see Proposition 3.4 in 27]. This new perspective on Algorithm 4 is rich in consequences: the entirety of the literature on particle Gibbs can be applied to step 3 of Algorithm 5, and we can expect that the curse of dimensionality can be controlled in this case too, provided that the auxiliary variables are used to design the proposal.

3.3. Adapted proposals in particle Gibbs with auxiliary observations

In the previous section, we have described an algorithm that recovers [27, 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

[4]While in [16] 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.

that we can incorporate additional 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 [27]. 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.

3.3.1. 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

$$\exp(\gamma(x_{0:T})) \approx \exp\left(\gamma(u_{0:T}) + \sum_{t=0}^{T} \frac{\partial \gamma(u_{0:T})}{\partial u_t} \cdot (x_t - u_t)\right) =: \prod_{t=0}^{T} \hat{g}_t(x_t \mid u_{0:T}). \tag{34}$$

Now, as in Section 2, we can form the proposal distributions

$$\begin{aligned} \tilde{p}_t(x_t \mid x_{t-1}, u_{0:T}) &\propto \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}\Sigma_t\right) \hat{g}_t(x_t \mid u_{0:T}) \\ &\propto \mathcal{N}\left(x_t; u_t + \frac{\delta_t}{2}\Sigma_t \frac{\partial \gamma(u_{0:T})}{\partial u_t}, \frac{\delta_t}{2}\Sigma_t\right) \end{aligned} \tag{35}$$

and similarly for $\tilde{p}_0$. Omitting the dependency on $u_{0:T}$ on the right handside for simplicity, we can therefore reformulate the auxiliary Feynman–Kac model as

$$\begin{aligned} \pi(x_{0:T} \mid u_{0:T}) &\propto p_0(x_0) \left\{\prod_{t=1}^{T} p_t(x_t \mid x_{t-1})\right\} \\ &\quad \times g_0(x_0) \left\{\prod_{t=1}^{T} g_t(x_t, x_{t-1})\right\} \left\{\prod_{t=0}^{T} \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right)\right\} \\ &\propto \tilde{p}_0(x_0) \left\{\prod_{t=1}^{T} \tilde{p}_t(x_t \mid x_{t-1})\right\} \tilde{g}_0(x_0) \left\{\prod_{t=1}^{T} \tilde{g}_t(x_t, x_{t-1})\right\}, \end{aligned} \tag{36}$$

for

$$\tilde{g}_t(x_t, x_{t-1}) = g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1}) \frac{\mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}\Sigma_t\right)}{\mathcal{N}\left(x_t; u_t + \frac{\delta_t}{2}\Sigma_t \frac{\partial \gamma(u_{0:T})}{\partial u_t}, \frac{\delta_t}{2}\Sigma_t\right)} \tag{37}$$

and similarly for $\tilde{g}_0$.

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 (33) to then recover a gradient-informed equivalent representation of $\pi(x_{0:T} \mid u_{0:T})$ that will still be local, as [27], but will have proposal distributions that are approximately "locally-optimal" [in the sense of, e.g., 13, Ch. 10] for the auxiliary target. Interestingly, these new proposal distributions are also fully separable in time, so that they can immediately be used in the parallel-in-time particle Gibbs algorithm of [16].

3.3.2. Approximately Gaussian transitions

Consider now the case when the prior process is conditionally Gaussian (this extends, as in Section 2.2, to the more general case when the prior is not conditionally Gaussian but its conditional means and covariances are tractable). We can easily design a model [this is called a guided proposal in 13, Section 10.3.2] locally adapted to the auxiliary observation u_t as

$$\begin{aligned}\tilde{p}_t(x_t \mid x_{t-1}, u_t) &\propto \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right) \mathcal{N}\left(x_t; m^X_{t-1}(x_{t-1}), C^X_{t-1}(x_{t-1})\right) \\ &\propto \mathcal{N}\left(x_t; \mu_t, \Lambda_t\right),\end{aligned} \tag{38}$$

for $\mu_t = m^X_{t-1}(x_{t-1}) + K_{t-1}[u_t - m^X_{t-1}(x_{t-1})]$ and $\Lambda_t = C^X_{t-1}(x_{t-1}) - K_{t-1}C^X_{t-1}(x_{t-1})$, where $K_{t-1} = C^X_{t-1}(x_{t-1})\left[C^X_{t-1}(x_{t-1}) + \frac{\delta_t}{2}\Sigma_t\right]^{-1}$. A similar form is available for $\tilde{p}_0$. 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

$$\pi(x_{0:T} \mid u_{0:T}) \propto \tilde{p}_0(x_0)\left\{\prod_{t=1}^{T} \tilde{p}_t(x_t \mid x_{t-1})\right\} \tilde{g}_0(x_0)\left\{\prod_{t=1}^{T} \tilde{g}_t(x_t, x_{t-1})\right\}, \tag{39}$$

where $\tilde{p}_t$ is given by (38), and

$$\tilde{g}_t(x_t, x_{t-1}) = \frac{g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1})}{\tilde{p}_t(x_t \mid x_{t-1})} \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right). \tag{40}$$

The resulting auxiliary Feynman–Kac model (39) can then be sampled from using Algorithm 3 where the particles are sampled from the proposal $\tilde{p}_t$ of (38) and the weights are computed using the potential functions $\tilde{g}_t$ of (40).

Using such a proposal 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, as they explicitly incorporate these inside the proposal model. We also note that the construction proposed in (38) and (39) extends to other methods developed to leverage approximate Gaussian conjugacy relationships in state-space models, for instance, they are directly compatible with Laplace approximations of the potential [see, e.g. 13, Section 10.5.3] or Rao–Blackwellisation [59].

3.3.3. Hybrid proposal models

It is worth highlighting that the two approaches presented above 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, thereby obtaining hybrid adapted proposals that may work better than their individual components taken in isolation.

With the notations above, this would, for example, correspond to

$$\begin{aligned}
&\tilde{p}_t(x_t \mid x_{t-1}, u_{0:T}) \\
&\quad \propto \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right) \mathcal{N}\left(x_t; m_{t-1}^X(x_{t-1}), C_{t-1}^X(x_{t-1})\right) \hat{g}_t(x_t \mid u_{0:T}) \\
&\quad \propto \mathcal{N}\left(u_t + \frac{\delta_t}{2}\Sigma_t \frac{\partial \gamma(u_{0:T})}{\partial x_t}; x_t, \frac{\delta_t}{2}\Sigma_t\right) \mathcal{N}\left(x_t; m_{t-1}^X(x_{t-1}), C_{t-1}^X(x_{t-1})\right),
\end{aligned} \tag{41}$$

if the linearisation point of γ was taken to be $u_{0:T}$. This can then be simplified explicitly as in (38) to obtain gradient-informed, guided proposals. Similarly as in Sections 3.3.1 and 3.3.2, we can then formulate the modified potential functions $\tilde{g}_t = g_t \cdot p_t / \tilde{p}_t$ and $\tilde{g}_0 = g_0 \cdot p_0 / \tilde{p}_0$ to obtain a new representation of the auxiliary Feynman–Kac model which can be sampled from using Algorithm 3.

Finally, 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.

3.4. Extension to pseudo-marginal methods

While the particle Gibbs approach to sampling from (32) is perhaps the most natural, it is also possible to instead consider a pseudo-marginal approach [3] as given by the particle marginal Metropolis—Hastings (PMMH) sampler of [1]. Consider a proposal distribution $q(u'_{0:T} \mid u_{0:T})$, for example, $\prod_0^T \mathcal{N}(u'_t; u_t, \frac{\delta_t}{2}\Sigma_t)$. 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 6.

This method is related to the method of [23]. 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 it results in correlated likelihood ratios $\frac{\hat{\mathcal{Z}}'_N}{\hat{\mathcal{Z}}^k_N}$ 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 5, we can hope to also benefit from a reduced variance of the likelihood estimates

Algorithm 6: Auxiliary pseudo-marginal sampler

Result: An updated trajectory $x_{0:T}^{k+1}$

Function *AUX-PM*$(x_{0:T}^k, u_{0:T}^k, \hat{\mathcal{Z}}_N^k)$

  Sample $u'_{0:T} \sim q(\cdot \mid u_{0:T}^k)$

  Sample $x'_{0:T}$ and $\hat{\mathcal{Z}}'_N$ using a particle filter targeting $\pi(x_{0:T} \mid u'_{0:T})$

  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 $x_{0:T}^k$

  return $x_{0:T}^{k+1}$

ratio in Algorithm 6. 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 [23], 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 [23, see the comments in Theorem 3 and Section 5.3], Algorithm 6 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 [50], is however not clear, and we leave the study of this question open for future work.

4. Experimental evaluation

In this section, we aim to empirically evaluate the statistical and computational behaviours of our proposed methods. To this end, we consider four sets of examples. In all cases we compare to state-of-the-art methods, that is, either the original method of [27] or [57].

- The first model is a multivariate stochastic volatility model known to be challenging for Gaussian approximations and used as a benchmark in, for example, [38, 27]. This model has latent Gaussian dynamics, and an observation model which are both differentiable with respect to the latent state, so that all the methods of Section 2.2 and Section 3 apply. We consider the same parametrisation as in [27], 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 [19] 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 third model performs joint parameter and state estimation for a discretely observed stochastic differential equation. This model was used in

[57] 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.

- While the three first examples highlight the benefits of our approach, the final example is a very simple, but illustrative toy-example, aimed at isolating their respective failure modes which we already alluded to in Sections 2 and 3.

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 50% across all time steps. This is more conservative than the recommendation of [27], 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 [71, 27], we consider $\delta_t \Sigma_t = \delta_t I$, with a single $\delta_t = \delta$ 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. Finally, we note that the choice of $N = 25$ particles for the cSMC samplers is somewhat arbitrary, and driven mostly by computational (memory) resources. The cSMC algorithm in Algorithm 3 together with the modification in Algorithm 11 is known to be robust to increasing T, even for a fixed $N \geq 2$ [50]; as such, the choice of N is not critical for the convergence of the algorithm. Nonetheless, its convergence rate improves polynomially with N [46, Theorem 1], and larger N values would improve the mixing properties of the algorithm. In simple terms, while it is beneficial to take N as large as possible, there is no minimum N required for the algorithm to converge.

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 an Nvidia® GeForce RTX 3090 GPU with 24 GB memory. All experiments were implemented in Python [76] using the JAX library [10] 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`.

4.1. Multivariate stochastic volatility model

We consider the same multivariate stochastic volatility example as in [27, Section E.3.]. This model is classically used as a benchmark for high dimensional SMC-related methods [see also 38]. 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

$$g(x_{0:T}) = \prod_{t=0}^{T} h(y_t \mid x_t), \text{ where } \quad h(y_t \mid x_t) = \prod_{d=1}^{d_x} \mathcal{N}(y_t(d); 0, \exp(x_t(d))). \quad (42)$$

As per [27], 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 $N = 25$.

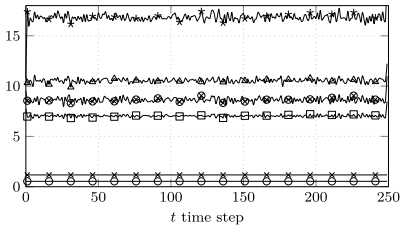
The different methods we compare here are the following: (i) auxiliary Kalman sampler with first order linearisation (11) (both on CPU and GPU), (ii) with second order linearisation (13) (both on CPU and GPU), (iii) auxiliary cSMC sampler with backward sampling for the proposals $\mathcal{N}(\cdot; u_t, \frac{\delta_t}{2} I)$ corresponding to [27] (on CPU), (iv) auxiliary cSMC sampler with parallel-in-time [16] 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 (35) (on CPU), (vi) auxiliary cSMC sampler with parallel-in-time sampling for the gradient-informed proposals (35) (on GPU), and (vii) the guided auxiliary cSMC sampler with backward sampling for both the proposals (38) and (41) (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 expected squared jump distance [ESJD, 62][5] for each sampler, defined as, for each time step t, the empirical value of

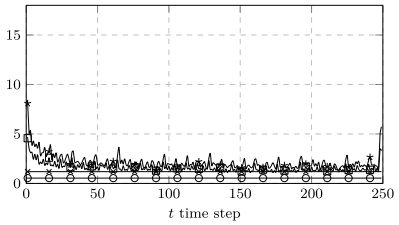
$$\frac{1}{L} \sum_{l=1}^{L-1} \sum_{i,j=1}^{d} \left[X_{t+1}^{A+l+1}(i,j) - X_{t+1}^{A+l}(i,j)\right]^2. \quad (43)$$

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) ESJD is reported in Figure 1a for the sequential versions

[5]The ESJD is, in first approximation, proportional to the effective sample size [ESS, see, e.g., 33] which measures the "equivalent" number of independent samples that would have resulted in an estimator with the same variance. The reason why we use the ESJD and not the ESS directly here is because the latter requires storing long sequences of sample trajectories, which is memory intensive and artificially decreased the performance of the GPU based methods, as GPUs have less memory available.



(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.



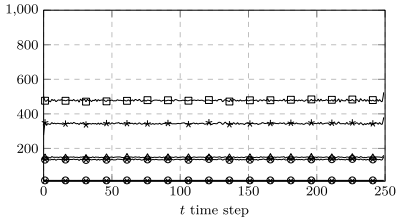
(b) GPU: auxiliary first order Kalman sampler —○—, second order Kalman sampler —×—, cSMC sampler —□—, and gradient-informed cSMC —✶— sampler. The latter two are hard to distinguish, but the gradient-informed cSMC is generally above the non-informed.

FIG 1. *Average (across 10 different experiments) expected squared jump distance per iteration for all the samplers considered on the stochastic volatility model of Section 4.1.*

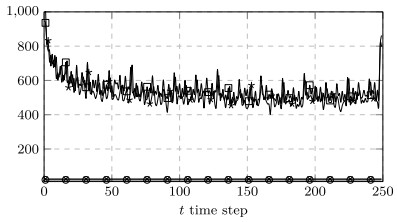
of the algorithm, and in Figure 1b for the parallel counterparts (noting that there is, as expected, no statistical difference between the sequential and parallel implementations of the Kalman samplers). As highlighted by Figure 1a, the gradient-informed auxiliary cSMC statistically dominates the alternatives 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 ESJD per second, rather than per iteration in Figures 2a and 2b. In this case, on the CPU, the method of [27] dominates the other ones. This is because it offers reasonable statistical efficiency ($\sim 70\%$ the ESJD 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. 8] and prompted the development of a square-root version of the parallel Kalman filtering and smoothing algorithms in [80]. Here, we 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 [43]. The issue did not arise for the sequen-



(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) expected squared jump distance per second for all the samplers considered on the stochastic volatility model.*

tial 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.

4.2. Spatio-temporal model

We now consider the spatio-temporal model of [19, 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

$$\begin{aligned} X_t(i,j) &= X_{t-1}(i,j)+U_t(i,j), \quad i,j=1,\dots,d,\\ Y_t(i,j) &= X_t(i,j)+V_t(i,j), \quad i,j=1,\dots,d, \end{aligned} \tag{44}$$

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.

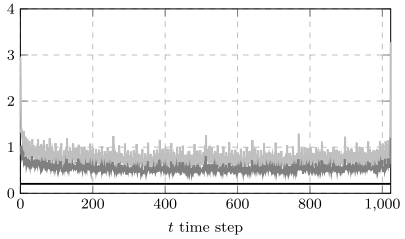
In [19], the parameters are chosen to be $\sigma_X=1$, $\nu=10$, $\tau=-1/4$, $r_y=1$, and $D[(i,j),i',j')]=|i-i'|+|j-j'|$, so that an observation is mostly corrupted by its direct neighbours. We keep all the parameters unchanged, with the exception that, in order to make the problem more difficult, 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 19]. 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 (21) 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. In other terms, instead of needing to compute conditional Gaussian distributions of dimension $d \times d$, and therefore needing to solve systems of size $d \times d$ (ocurring a cost $\mathcal{O}((d^2)^3)$ on CPU), we only need to solve one-dimensional systems, that is, divide and multiply 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 (under no parallelisation, we can expect is to be roughly $N + 1$ times more expensive, where $N + 1$ is the total number of particles).

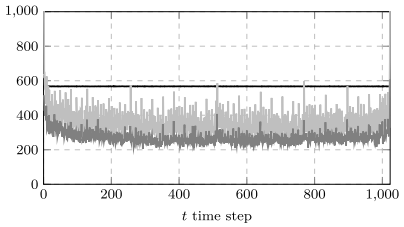
The experiment design is as follows: we simulate 20 datasets from (44). For each of these, we set the initial trajectory of the MCMC chain to be the result of a single trajectory formed from the backward sampling [36] of a bootstrap filter algorithm with 1 000 particles (this gives bad smoothing statistics 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 but do so in Appendix D. Instead, we focus on (i) the parallel-in-time version of [27] given by using [16] on step 3 of Algorithm 5, (ii) the parallel-in-time version of the gradient auxiliary proposal (35) of Section 3.3, which we refer to as gradient-informed, and finally (iii) the auxiliary Kalman sampler (11) of Section 2.2, with first-order linearisation only, noting that the second order would remove the computational benefits of having a separable prior.

As per [71], 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 ESJD is shown, averaged over all experiments, in Figure 3a, while the time-scaled ESJD, namely ESJD 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 ESJD than the basic PIT auxiliary cSMC which in turn has a better ESJD than the auxiliary Kalman sampler. The ordering of these methods however changes if one takes into ac-



(a) 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) 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) expected squared jump distance per iteration and second for all the samplers considered on the spatio-temporal model* (44).

count 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 counterpart.

In practice, the auxiliary Kalman, conditional SMC, and gradient-enhanced conditional SMC samplers took respectively on 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.

4.3. Parameter estimation in a continuous-discrete diffusion smoothing problem

In this section, we consider the same experiment as in [57, 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

$$\beta_\theta(x) = \begin{pmatrix} \theta_1(x_2 - x_1) \\ \theta_2 x_1 - x_2 - x_1 x_3 \\ x_1 x_2 - \theta_3 x_3 \end{pmatrix}. \tag{45}$$

The state is then observed at regular intervals (every $t_0 = 0.01, t_1 = 0.02, \ldots, t_K = 2$) through its second and third component only, giving an observation model $Y_k \sim \mathcal{N}(Hx(t_k), 5I_2)$, for $H = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and where I_2 denotes the identity matrix in $\mathbb{R}^2$. In order to provide comparable results to [57], 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 [57]. As per their experiment, we sample from the joint distribution

$$\pi(x(t'_0), \dots, x(t'_L), \theta \mid y_{0:K}), \tag{46}$$

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 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 [57, 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

$$\mathbb{E}[x(t'_l) \mid x(t'_{l-1})] = \beta_{\theta^k}(x(t'_{l-1}))(t'_l - t'_{l-1}) \tag{47}$$

around $x^k(t'_{l-1})$ using the method of Section 2.2 with extended linearisation, obtaining approximations

$$p(x(t'_l) \mid x(t'_{l-1})) \approx \mathcal{N}(x(t'_l); F_{l-1}x(t'_{l-1}) + b_{l-1}, Q_{l-1}). \tag{48}$$

For $l = 0, \dots, L$, we then sample $u_l \sim \mathcal{N}(x(t'_l), \frac{\delta}{2} I_3)$, where I_3 denotes the identity matrix in $\mathbb{R}^3$, and then form the proposal

$$\begin{aligned} &q\left((z(t'_l))_{l=0}^L \mid u_{0:L}, (x^k(t'_l))_{l=0}^L, y_{0:L}\right) \\ &\quad \propto \mathcal{N}\left(z(t'_0); m_0, P_0\right) \left\{\prod_{l=1}^{L} \mathcal{N}\left(z(t'_l); F_{l-1}z(t'_{l-1}) + b_{l-1}, Q_{l-1}\right)\right\} \\ &\quad \left\{\prod_{k=0}^{K} \mathcal{N}\left(y_k; Hz(t_k), 5I_2\right)\right\} \left\{\prod_{l=0}^{L} \mathcal{N}\left(u_l; z(t'_l), \frac{\delta}{2} I_3\right)\right\} \end{aligned} \tag{49}$$

targeting the augmented model

$$\begin{aligned} \pi\left((z(t'_l))_{l=0}^L \mid u_{0:L}, y_{0:K}\right) \propto \mathcal{N}\left(z(t'_0); m_0, P_0\right) &\left\{\prod_{l=1}^{L} p\left(z(t'_l) \mid z(t'_{l-1})\right)\right\} \\ &\left\{\prod_{k=0}^{K} \mathcal{N}\left(y_k; Hz(t_k), 5I_2\right)\right\} \left\{\prod_{l=0}^{L} \mathcal{N}\left(u_l; z(t'_l), \frac{\delta}{2} I_3\right)\right\}. \end{aligned} \tag{50}$$

We run 2 500 adaptation steps, during which we modify δ to target an average acceptance rate of 23.4% (as per [57]). 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

TABLE 1
Effective sample size (ESS) for the auxiliary sampler, computed using chains of length 10^5. *The results for [57] 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
[57]	10480.3	22890.5	24070.2	4592.4	15379.5	10917.7

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 [57] for both the parameters and the smoothing marginals (while the posterior distributions were similar). We report this in Table 1.

In practice, our sampler took 3 149 seconds (52 minutes) to run on the GPU, and 9 424 seconds (2h30mn) on the CPU. [57], on the other hand, resulted in much faster run times (approximately 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 [7], we use the JAX library [10] 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 (49) 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.

4.4. *Failure modes*

In this section, we highlight the different failure modes of both the local (cSMC-based) and global (auxiliary Kalman-based) methods. To do so, we consider a much simpler model than the ones presented in the previous sections, which is aimed at interpolating between the different regimes in which the methods outperform (or not) each other.

The model is given as a two time-step one-dimensional linear Gaussian state-space model, with a single (unlikely) observation at the second time step. The latent dynamics are given by an $\mathcal{N}(0,1)$ stationary autoregressive process, that is $x_t \sim \mathcal{N}(\rho x_{t-1}, 1-\rho^2)$, $x_0 \sim \mathcal{N}(0,1)$, and the observation model $y \sim \mathcal{N}(x_1, r^2)$. In other terms, the higher the value of ρ, the more 'sticky' the dynamics are and the more the model is likely to be in a regime where the global Kalman samplers are expected to outperform the local cSMC ones. On the other hand, the smaller the value of ρ, the more separable the dynamics are and the more the model is likely to be in a regime where the local cSMC samplers are expected to outperform the global Kalman ones. Lowering the observation noise r corresponds

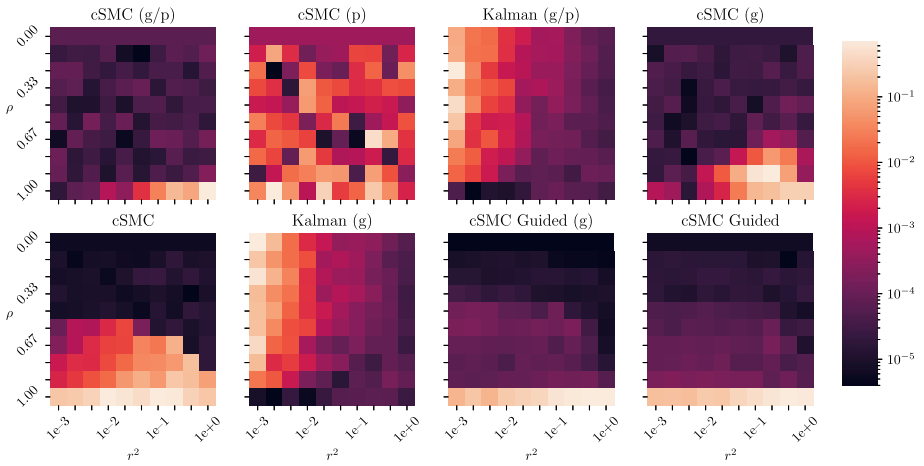


FIG 4. *Heatmap of average square error of the estimated mean of the first time step x_0, scaled by the true standard deviation for the different samplers as a function of the observation noise r^2 and the autocorrelation of the latent dynamics ρ. Here "cSMC" stands for [27], with (g) indicating gradient information (Section 3.3.1), and (p) indicating parallelisation in time of [16]. "Kalman" stands for the first-order auxiliary Kalman samplers of Section 2, and "Guided cSMC" stands for the methods of Sections 3.3.2 and 3.3.3 (with (g) indicating gradient information for the latter).*

to modeling a case where an observation is very unlikely (or equivalently highly informative) relative to the rest of the observations. This is a case where the global Kalman samplers are expected to underperform as their scale parameter δ will shrink to mostly account for this single time step, while the rest of the time series may have required a much higher δ to achieve good mixing.

We set $y = 5$ to be a rare observation, and make ρ vary between 0 and 0.999 and r^2 between 0.001 and 1. For each combination of ρ and r^2, we run all the samplers started at stationarity (note that we can do so because the true model is linear Gaussian), run 5 000 adaptation steps, and then 8 times 20 000 iterations to compute the empirical mean of the first time step x_0. The experiment is then repeated 10 times for each ρ and r^2 to account for the randomness in the initialisation of the samplers, after which we compute the mean squared error of the mean approximation computed as

$$\text{MSE} = \frac{1}{10}\sum_{i=1}^{10}\left(\frac{\hat{x}_{0,i} - m_0}{s_0}\right)^2, \tag{51}$$

where $\hat{x}_{0,i}$ is the mean estimator of the 0-th time step of the latent state for the i-th out of ten experiments, and m_0, s_0 are the true posterior mean and standard deviation of the 0-th time step. The results are reported in Figure 4.

As expected, the Kalman samplers collapse when the observation noise r^2 is very low and the autocorrelation of the latent dynamics ρ is low too. This is because the Kalman samplers adapt their step-size δ to account for the most

informative time step (here the second one, x_1), and not the rest of the time series. This results in a very slow mixing of the first time step x_0.

On the other hand, the cSMC samplers are much more robust to the low correlation setting, as they can adapt their step-sizes δ_t per time step, and thus mix "locally as well" for all time steps. However, they collapse when the correlation is very high, as proposals at time 0 do not account for the information at time 1 aside from the information given by the auxiliary variable u_0. This is less of an issue for the guided cSMC samplers, as they do account for the correlation between the latent states, but they still collapse when the correlation is extremely high.

Finally, the method of [27], i.e. Algorithm 4, and its parallel implementation via [16] discussed in Section 3 are not robust to varying degrees of correlation or informativeness. Gradient information, however, mitigates this issue, and the gradient-informed cSMC of [27] and its parallel counterpart are competitive with the guided cSMC and the Kalman samplers in nearly all regimes, despite the fact that they do not *explicitly* account for the correlation between the latent states.

5. 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 distribution of the trajectory (conditionally on the artificial observations) 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 [71] 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 [27] 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. Formally, 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 severely 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 [32].
2. Models with very non-Gaussian latent dynamics or observations, such as those exhibiting multiplicative noise or presenting boundary constraints akin to discontinuities.

Other, softer, issues comprise the following: (i) 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), and, in models where no specific structure alleviates these computations, they can quickly become computationally overwhelming as the dimension of the latent space increases; (ii) the method is based on a global acceptance step, which means that its performance will naturally degrade as the number of time steps increases, and will be sensitive to a single bad time step, making it somewhat brittle to heterogeneously informative observations.

Replacing the LGSSM proposal of Section 2.2 by a local conditional SMC update as per Section 3 allowed us to trade the single expensive accept-reject step for a series of cheaper local ones. This solved the brittleness issue, because time steps are considered more independently, and the calibration of the method can happen more locally. Additionally, the conditional SMC instance of the method naturally inherits the scalability in time of the underlying cSMC algorithm, and, contrary to the auxiliary Kalman sampler, does not require specific treatment to handle increasing numbers of time steps [27, 50, 46]. However, the usual issues with cSMC remain: several trajectories need to be simulated, and the fully adapted auxiliary cSMCs of Section 3.3 cannot be parallelised-in-time, which we showed to be a significant bottleneck in the case of the spatio-temporal model of Section 4.2. They also do not solve the problem of intractable densities, or multimodality.

The reformulation of [27] 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. We believe that this can still be improved upon many-fold in a number of settings and a natural first step would be to combine these with methods developed to tackle degeneracy in particle Gibbs [e.g. 52] or very long time series [47].

In addition to these, we mention that, since the first version of this article, a follow-up work, [17], has built upon the guided cSMC perspective to unify conditional SMC and Metropolis adjusted Langevin algorithms [MALA, 6] as well as the prior-informed samplers of [71] and other related methods. While the methods of [17] are not parallelisable, contrary to most of the methods proposed here, they overcome some limitations highlighted in Section 4.4, in particular the collapse of cSMC in the highly-informative prior regime.

A final remark is concerned with the implementation of the prefix-sum algorithm [9] in the JAX library [10]. 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 [76] rather than natively using the CUDA [61] GPU backend. This is in contrast to other control flow primitives such as "for loops" and "if-else" branching, and a native implementation of the algorithm, fully GPU-focused would improve the time-performance of the Kalman samplers.

Appendix A: Sampling and evaluating LGSSM pathwise smoothing distributions

In this Section, we describe how the smoothing distribution forming the proposal of the auxiliary Kalman samplers in Section 2.2 can be sampled and evaluated efficiently. We first review the "classical" sequential Kalman filter and backward sampling algorithms, which are used to sample from the smoothing distribution of a linear Gaussian state-space model (LGSSM) in $\mathcal{O}(T)$ steps on sequential hardware. We then show how these algorithms can be parallelised to run in $\mathcal{O}(\log T)$ steps on parallel hardware, either by using prefix-sum algorithms or by divide-and-conquer strategies.

In this section only, and in contrast with the notations of the main text, we consider that we are given a LGSSM of the form

$$\begin{aligned} p_t(x_t \mid x_{t-1}) &= \mathcal{N}(x_t; F_{t-1}x_{t-1} + b_{t-1}, Q_{t-1}), \quad p_0(x_0) = \mathcal{N}(x_0; m_0, P_0), \\ p_t(y_t \mid x_t) &= \mathcal{N}(y_t; H_t x_t + c_t, R_t), \end{aligned} \tag{52}$$

and we want to sample from the smoothing distribution as well as evaluate its likelihood

$$p(x_{0:T} \mid y_{0:T}) = \frac{p(x_{0:T}, y_{0:T})}{p(y_{0:T})} = \frac{p(x_{0:T}, y_{0:T})}{\int p(x_{0:T}, y_{0:T}) \mathrm{d}x_{0:T}}. \tag{53}$$

Noting that, without loss of generality, we can assume that $c_t = 0$, we will omit it from the notation in the remainder of this section.

A.1. Sequential implementations

The Kalman filter [45] and Rauch–Tung–Striebel smoother [64] are well-known algorithms to compute the *marginal* filtering and smoothing distributions of a LGSSM in $\mathcal{O}(T)$ steps [see, e.g., 68, for a review]. The Kalman filter computes the filtering distribution $p(x_t \mid y_{0:t}) = \mathcal{N}(x_t; m_t^f, P_t^f)$ recursively for $t = 1, \ldots, T$ as

$$\begin{aligned} m_t^f &= m_t^p + K_t(y_t - H_t m_t^p), \\ P_t^f &= P_t^p - K_t H_t P_t^p, \end{aligned} \tag{54}$$

where $K_t = P_t^p H_t^\top (H_t P_t^p H_t^\top + R_t)^{-1}$ is the Kalman gain and

$$m_t^p = F_{t-1} m_{t-1}^f + b_{t-1}, \quad P_t^p = F_{t-1} P_{t-1}^f F_{t-1}^\top + Q_{t-1} \tag{55}$$

are the predicted mean and covariance of the filtering distribution. The initialisation is done with $m_0^p = m_0$, $P_0^p = P_0$ and

$$m_0^f = m_0 + K_0(y_0 - H_0 m_0), \quad P_0^f = P_0 - K_0 H_0 P_0, \tag{56}$$

for $K_0 = P_0 H_0^\top (H_0 P_0 H_0^\top + R_0)^{-1}$. Importantly, the marginal likelihood of the observations $y_{0:T}$ can be computed recursively as

$$p(y_{0:T}) = \prod_{t=0}^{T} \mathcal{N}(y_t; H_t m_t^f, H_t P_t^f H_t^\top + R_t). \tag{57}$$

Put together, this results in Algorithm 7 for the Kalman filter.

Algorithm 7: Kalman filter

Result: The filtering means and covariances $m_{0:T}^f$, $P_{0:T}^f$ and the likelihood $p(y_{0:T})$
Data: The observations $y_{0:T}$ and the LGSSM parameters
Function KalmanFilter($y_{0:T}$, m_0, P_0, $F_{0:T-1}$, $b_{0:T-1}$, $Q_{0:T-1}$, $H_{0:T}$, $R_{0:T}$)
  Initialise $m_0^p = m_0$, $P_0^p = P_0$
  Set $K_0 = P_0 H_0^\top (H_0 P_0 H_0^\top + R_0)^{-1}$
  Initialise $m_0^f = K_0 y_0 + (I - K_0 H_0) m_0$, $P_0^f = P_0 - K_0 H_0 P_0$
  Initialise $p(y_{0:T}) = \mathcal{N}(y_0; H_0 m_0^f, H_0 P_0^f H_0^\top + R_0)$
  for $t = 1, \ldots, T$ **do**
    Compute $m_t^p = F_{t-1} m_{t-1}^f + b_{t-1}$, $P_t^p = F_{t-1} P_{t-1}^f F_{t-1}^\top + Q_{t-1}$
    Compute $K_t = P_t^p H_t^\top (H_t P_t^p H_t^\top + R_t)^{-1}$
    Compute $m_t^f = m_t^p + K_t(y_t - H_t m_t^p)$, $P_t^f = P_t^p - K_t H_t P_t^p$
    Update $p(y_{0:T}) \leftarrow p(y_{0:T}) \times \mathcal{N}(y_t; H_t m_t^p, H_t P_t^p H_t^\top + R_t)$
  return $m_{0:T}^f$, $P_{0:T}^f$, $p(y_{0:T})$

Once the marginal likelihood $p(y_{0:T})$ has been computed, it is then easy to evaluate the smoothing distribution $p(x_{0:T} \mid y_{0:T})$ using the identity

$$p(x_{0:T} \mid y_{0:T}) = \frac{p(x_{0:T}, y_{0:T})}{p(y_{0:T})} \tag{58}$$

noting that the numerator can be computed as the product

$$\begin{aligned} &p(x_{0:T}, y_{0:T}) \\ &\quad = p_0(x_0) \left\{ \prod_{t=1}^{T} p(x_t \mid x_{t-1}) \right\} \prod_{t=0}^{T} p(y_t \mid x_t) \\ &\quad = \mathcal{N}(x_0; m_0, P_0) \left\{ \prod_{t=1}^{T} \mathcal{N}(x_t; F_{t-1} x_{t-1} + b_{t-1}, Q_{t-1}) \right\} \prod_{t=0}^{T} \mathcal{N}(y_t; H_t x_t, R_t), \end{aligned} \tag{59}$$

or more numerically stably with the sum of the logarithms of the terms in (59) and similarly for the denominator $p(y_{0:T})$.

The backward sampler then proceeds recursively to obtain a sample from the smoothing distribution $p(x_{0:T} \mid y_{0:T})$ in $\mathcal{O}(T)$ steps via the recursive identity

$$p(x_{t-1} \mid y_{0:T}, x_{t+1:T}) = p(x_{t-1} \mid y_{0:t}, x_{t+1}), \tag{60}$$

which, for LGSSMs is given as a conditional Gaussian distribution

$$p(x_{t-1} \mid y_{0:t}, x_{t+1}) = \mathcal{N}\left(x_t; m_t^f + G_t(x_{t+1} - m_{t+1}^p), P_t^f - G_t P_{t+1}^p G_t^\top\right), \quad t < T, \tag{61}$$

and with initialisation $p(x_T \mid y_{0:T}) = \mathcal{N}(x_T; m_T^f, P_T^f)$. The backward sampler is summarised in Algorithm 8 which we refer to as the Rauch–Tung–Striebel *sampler* due to its similarity with the Rauch–Tung–Striebel smoother [64]. Other approaches to sampling from the smoothing distribution of a LGSSM exist [see, e.g., 26], but do not necessarily improve the computational complexity and are not detailed here.

Algorithm 8: Rauch–Tung–Striebel sampler

Result: A sample from the smoothing distribution $p(x_{0:T} \mid y_{0:T})$

Function SequentialSampler($m_{0:T}^f$, $P_{0:T}^f$, $F_{0:T-1}$, $b_{0:T-1}$, $Q_{0:T-1}$, $H_{0:T}$, $R_{0:T}$, $y_{0:T}$)

  Initialise $x_T \sim \mathcal{N}(m_T^f, P_T^f)$

  for $t = T-1, \ldots, 0$ **do**

    Compute $G_t = P_t^f F_t^\top \left(F_t P_t^f F_t^\top + Q_t\right)^{-1}$

    Sample $x_t \sim \mathcal{N}\left(m_t^f + G_t(x_{t+1} - m_{t+1}^p), P_t^f - G_t P_{t+1}^p G_t^\top\right)$

  return $x_{0:T}$

A.2. Parallel implementations

We now turn to two different strategies to parallelise the Kalman filter and Rauch–Tung–Striebel sampler algorithms on parallel hardware. The first strategy is to use prefix-sum algorithms [9] to parallelise the backward sampler, while the second strategy is to use divide-and-conquer strategies. Because both these rely on having pre-computed the filtering means and covariances $m_{0:T}^f$, $P_{0:T}^f$, we first describe how to parallelise the Kalman filter algorithm in $\mathcal{O}(\log T)$ steps following the methods of [67].

A.2.1. Prefix-sums and the parallel Kalman filter

Prefix-sum algorithms [9] are a class of parallel algorithms that can be used to compute the *cumulative* composition $e_1 \circ \ldots \circ e_t$, $t = 1, \ldots, T$ of a sequence of

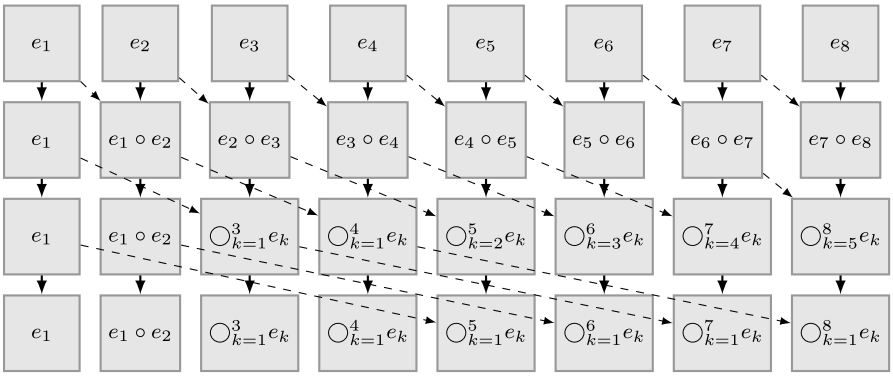


FIG 5. *Illustration of the Hillis–Steele prefix-sum algorithm. The algorithm performs* $\lfloor \log_2 T \rfloor$ *iterations, each of which using operations which are embarrassingly parallel.*

T elements in $\mathcal{O}(\log T)$ steps on parallel hardware. It relies on the associative property of the operator $\circ$, whereby we have

$$(e_1 \circ e_2) \circ e_3 = e_1 \circ (e_2 \circ e_3). \tag{62}$$

A typical example is when the e_t's are scalars and the operator $\circ$ is the addition, in which case the prefix-sum of the sequence e_t is the cumulative sum $s_t = e_1 + \cdots + e_t$, $t = 1, \ldots, T$ of the sequence. Several parallel implementations of prefix-sums are available, with different memory/parallelisation properties. In Algorithm 9 we illustrate the simplest such algorithm, known as the Hillis–Steele scan [41]. A visual representation of the algorithm is also given in Figure 5. As can be seen, the algorithm performs $\lfloor \log_2 T \rfloor$ iterations, each of which requires (at most) T operations but which are embarrassingly parallel, and thus the full algorithm runs in $\mathcal{O}(\log T)$ steps on parallel hardware provided enough parallel resources are available. In practice, more efficient implementations exist, such as the work-efficient scan [9] but we do not detail them here.

Algorithm 9: Hillis-Steele algorithm

Result: Prefix-sums $e_1 \circ \ldots \circ e_t$, for $t = 1, \ldots, T$.
Function PREFIXSUM$(e_1, \ldots, e_T)$
  for $d \leftarrow 0$ **to** $\lfloor \log_2 T \rfloor$ **do**
    for $t \leftarrow T - 1$ **to** 0 **do** `// in parallel`
      if $t - 2^d \geq 0$ **then**
        $e_t \leftarrow e_{t-2^d} \circ e_t$
  return $e_1, \ldots, e_T$

In order to apply the prefix-sum algorithm to the Kalman filter, we now need to express the filtering means and covariances $m^f_{0:T}$, $P^f_{0:T}$ as the result of a prefix-sum operation for elements e_t and operator $\circ$ to be defined. This is done

in [67] by remarking that the Bayesian filtering recursion (for both the state and the marginal likelihood) can be written as

$$\begin{aligned} p(x_t \mid y_{0:t}) &= \frac{\int p(y_t \mid x_{t-1}) p(x_t \mid y_t, x_{t-1}) p(x_{t-1} \mid y_{0:t-1}) \mathrm{d}x_{t-1}}{\int p(y_t \mid x_{t-1}) p(x_{t-1} \mid y_{0:t-1}) \mathrm{d}x_{t-1}}, \\ p(y_{0:t}) &= p(y_{0:t-1}) \int p(y_t \mid x_{t-1}) p(x_{t-1} \mid y_{0:t-1}) \mathrm{d}x_{t-1}, \end{aligned} \tag{63}$$

noting that $p(y_t \mid y_{0:t-1}) = \int p(y_t \mid x_{t-1}) p(x_{t-1} \mid y_{0:t-1}) \mathrm{d}x_{t-1}$. As a consequence, the elements e_t can be identified as the pairs of (conditional) distributions $[p(x_t \mid y_t, x_{t-1}), p(y_t \mid x_{t-1})]^\top$ appearing in (63), and the operator $\circ$ as the integration corresponding to

$$\begin{bmatrix} p(x_t \mid y_t, x_{t-1}) \\ p(y_t \mid x_{t-1}) \end{bmatrix} \circ \begin{bmatrix} p(x_{t-1} \mid y_{0:t-1}) \\ p(y_{0:t-1}) \end{bmatrix} = \begin{bmatrix} p(x_t \mid y_{0:t}) \\ p(y_{0:t}) \end{bmatrix}. \tag{64}$$

Thankfully, both the element pairs and the operator can be computed[6] for Gaussian LGSSMs [for details on their expressions, see 67], and the prefix-sum algorithm can be applied to compute the filtering means and covariances $m^f_{0:T}$, $P^f_{0:T}$ and $p(y_{0:t})$ in $\mathcal{O}(\log T)$ steps on parallel hardware.

A.2.2. Parallel Rauch–Tung–Striebel sampler

Now that we have the filtering means and covariances $m^f_{0:T}$, $P^f_{0:T}$, we can modify Algorithm 8 to use the prefix-sum algorithm to sample from the smoothing distribution $p(x_{0:T} \mid y_{0:T})$ in $\mathcal{O}(\log T)$ parallel steps. Indeed, we know [28, Proposition 1] that

$$\begin{aligned} p(x_T \mid y_{0:T}) &= \mathcal{N}(x_T; m^f_T, P^f_T), \\ p(x_t \mid x_{t+1}, y_{0:t}) &= \mathcal{N}\left(x_t; m^f_t + G_t[x_{t+1} - F_t m^f_t - b_t], \Sigma_t\right), \quad t < T, \end{aligned} \tag{65}$$

where $G_t = P^f_t F_t^\top \left(F_t P^f_t F_t^\top + Q_t\right)^{-1}$ and $\Sigma_t = P^f_t - G_t(F_t P^f_t F_t^\top + Q_t) G_t^\top$ for all $t < T$.

We can furthermore rearrange the terms to express $\hat{X}_T \sim \mathcal{N}(m^f_T, P^f_T)$ and $\hat{X}_t \sim p(x_t \mid \hat{X}_{t+1}, y_{0:t})$ recursively as $\hat{X}_t = G_t \hat{X}_{t+1} + U_t$, where the U_t's are independently distributed as Gaussians $\mathcal{N}(m^f_t - G_t(F_t m^f_t + b_t), \Sigma_t)$ for all $t < T$. We also let $G_T = 0$, so that we can then define $U_T \sim \mathcal{N}(m^f_T, P^f_T)$ to be a sample of the final marginal smoothing distribution. Because the means and covariances of the U_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 $p(x_{0:T} \mid y_{0:T})$ we then need to apply the recursion to the pre-sampled sequence

[6] In practice, the marginal likelihood $p(y_{0:t})$ is obtained up to a multiplicative constant that may depend on the parameters of the LGSSM, and one therefore needs to perform a second step to compute it using (57).

U_t, $t = 0, \ldots, T$. However, the recursive dependency in (65) 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 [9]. Thankfully, this is readily done by considering the elements $e_t = \begin{bmatrix} G_t & U_t \end{bmatrix}^\top$ and the operator $\circ$ defined as follows

$$(G_{ij}, U_{ij}) = (G_i, U_i) \circ (G_j, U_j), \quad \text{where } G_{ij} = G_i G_j, \text{ and } U_{ij} = G_i U_j + U_i. \tag{66}$$

Proposition A.1. *The backward prefix-sum of operator $\circ$ applied to the sequence (G_t, U_t), $t = 0, \ldots, T$, recovers the pathwise smoothing distribution $p(x_{0:T} \mid y_{0:T})$, that is, if $(\tilde{G}_t, \tilde{U}_t) = (G_t, U_t) \circ \ldots \circ (G_T, U_T)$, then $(\tilde{U}_0, \ldots, \tilde{U}_T)$ is distributed according to $p(x_{0:T} \mid y_{0:T})$.*

Proof. The operator $\circ$ defined in (30) is clearly associative. We prove that its result corresponds to sampling from the pathwise smoothing distribution by reversed induction: suppose that $(\tilde{U}_t, \ldots, \tilde{U}_T)$ is distributed according to $p(x_{t:T} \mid y_{0:T})$, then $\tilde{U}_{t-1} = G_{t-1}\tilde{U}_t + U_{t-1}$, which is distributed according to $p(x_{t-1} \mid \tilde{U}_t, y_{0:t-1})$ as discussed before, so that $(\tilde{U}_{t-1}, \ldots, \tilde{U}_T)$ is distributed according to $p(x_{t-1:T} \mid y_{0:T})$. 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 [67] to compute the filtering means and covariances m_t^f, P_t^f, $t = 0, \ldots, T$, then form all the elements G_t and sample U_t fully in parallel, and finally, apply the prefix-sum primitive [9] to $(G_t, U_t)_{t=0}^T$ with the associative operator $\circ$. The parallel implementation of the Rauch–Tung–Striebel sampler is then given in Algorithm 10.

Algorithm 10: Parallel Rauch–Tung–Striebel sampler

Result: A sample from the smoothing distribution $p(x_{0:T} \mid y_{0:T})$

Function *ParallelSampler*($m_{0:T}^f$, $P_{0:T}^f$, $F_{0:T-1}$, $b_{0:T-1}$, $Q_{0:T-1}$, $H_{0:T}$, $R_{0:T}$, $y_{0:T}$)

  Initialise $U_T \sim \mathcal{N}(m_T^f, P_T^f)$

  for $t = T-1, \ldots, 0$ **do** `// in parallel`

    Compute $G_t = P_t^f F_t^\top \left(F_t P_t^f F_t^\top + Q_t\right)^{-1}$

    Sample $U_t \sim \mathcal{N}\left(m_t^f - G_t(F_t m_t^f + b_t), P_t^f - G_t(F_t P_t^f F_t^\top + Q_t) G_t^\top\right)$

  Apply the prefix-sum algorithm to $(G_t, U_t)_{t=T}^0$ with the operator $\circ$

  return $U_{0:T}$

A.2.3. Divide-and-conquer strategies

An alternative strategy to parallelise the Kalman filter and Rauch–Tung–Striebel sampler algorithms is to use divide-and-conquer strategies. Again, we assume that the filtering means and covariances $m_{0:T}^f$, $P_{0:T}^f$ have been computed using the parallel-in-time Kalman filter of [67] or similar methods.

We now present a divide-and-conquer alternative to Section A.2.2 for PIT sampling from the pathwise smoothing distribution of LGSSMs. The method is based on recursively finding tractable Gaussian expressions for the "bridging" $p(x_l \mid y_{0:T}, x_k, x_m)$, $0 \leq k < l < m \leq T$ of the smoothing distribution. This will allow us to derive a tree-based divide-and-conquer sampling mechanism for the pathwise smoothing distribution $p(x_{0:T} \mid y_{0:T})$.

Suppose we are given the LGSSM (52), then given three indices $0 \leq k < l < m \leq T$. We have

$$p(x_l \mid y_{0:T}, x_k, x_m) = \frac{p(x_k, x_l \mid y_{0:T}, x_m)}{p(x_k \mid y_{0:T}, x_m)} \tag{67}$$

with, furthermore,

$$p(x_k, x_l \mid y_{0:T}, x_m) = p(x_k \mid y_{0:T}, x_l) p(x_l \mid y_{0:T}, x_m) \tag{68}$$

thanks the to Markovian structure of the model. Now let $p(x_k \mid y_{0:T}, x_l)$ and $p(x_l \mid y_{0:T}, x_m)$ be given by

$$\begin{aligned} p(x_k \mid y_{0:T}, x_l) &= \mathcal{N}(x_k; E_{k:l} x_l + g_{k:l}, L_{k:l}) \\ p(x_l \mid y_{0:T}, x_m) &= \mathcal{N}(x_k; E_{l:m} x_m + g_{l:m}, L_{l:m}) \end{aligned} \tag{69}$$

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

$$\begin{aligned} &p(x_k, x_l \mid y_{0:T}, x_m) \\ &= \mathcal{N}\left(\begin{bmatrix} x_l \\ x_k \end{bmatrix}; \begin{bmatrix} E_{l:m} x_m + g_{l:m} \\ E_{k:l} E_{l:m} x_m + E_{k:l} g_{l:m} + g_{k:l} \end{bmatrix}, \begin{bmatrix} L_{l:m} & L_{l:m} E_{k:l}^\top \\ E_{k:l} L_{l:m} & E_{k:l} L_{l:m} E_{k:l}^\top + L_{k:l} \end{bmatrix} \right) \end{aligned} \tag{70}$$

giving both the marginal distribution of x_k

$$\begin{aligned} p(x_k \mid y_{0:T}, x_m) &= \mathcal{N}(x_k; E_{k:l} E_{l:m} x_m + E_{k:l} g_{l:m} + g_{k:l}, E_{k:l} L_{l:m} E_{k:l}^\top + L_{k:l}) \\ &= \mathcal{N}(x_k; E_{k:m} x_m + g_{k:m}, L_{k:m}), \end{aligned} \tag{71}$$

where

$$E_{k:m} = E_{k:l} E_{l:m}, \quad g_{k:m} = E_{k:l} g_{l:m} + g_{k:l}, \quad L_{k:m} = E_{k:l} L_{l:m} E_{k:l}^\top + L_{k:l}, \tag{72}$$

and (after simplification for (72)) the conditional distribution of x_l

$$p(x_l \mid y_{0:T}, x_k, x_m) = \mathcal{N}(x_l; G_{k:l:m} x_k + \Gamma_{k:l:m} x_m + w_{k:l:m}, V_{k:l:m}), \tag{73}$$

for

$$\begin{aligned} G_{k:l:m} &= L_{l:m} E_{k:l}^\top L_{k:m}^{-1}, & w_{k:l:m} &= g_{l:m} - G_{k:l:m} g_{k:m}, \\ \Gamma_{k:l:m} &= E_{l:m} - G_{k:l:m} E_{k:m}, & V_{k:l:m} &= L_{l:m} - G_{k:l:m} L_{k:m} G_{k:l:m}^\top. \end{aligned} \tag{74}$$

This construction provides a recursive tree structure for sampling from $p(x_{0:T} \mid y_{0:T})$ which can be initialised by

$$p(x_t \mid y_{0:T}, x_{t+1}) = \mathcal{N}(x_t; E_{t:t+1}x_{t+1} + g_{t:t+1}, L_{t:t+1}), \tag{75}$$

with

$$\begin{aligned} E_{t:t+1} &= P_t^f F_t^\top (F_t P_t^f F_t^\top + Q_t)^{-1}, \\ g_{t:t+1} &= m_t^f - E_{t:t+1}(F_t m_t^f + b_t), \\ L_{t:t+1} &= P_t^f - E_{t:t+1} F_t P_t^f, \end{aligned} \tag{76}$$

and $p(x_T \mid y_{0:T}) = \mathcal{N}(x_T; m_T^f, P_T^f)$. Finally, noting that

$$p(x_0 \mid y_{0:T}, x_T) = \mathcal{N}(x_0; E_{0:T} m_T^f + g_{0:T}, L_{0:T}), \tag{77}$$

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 A.2.2, to use the parallel-in-time Kalman filtering method of [67] to compute the filtering means and covariances m_t^f, P_t^f, $t = 0, \ldots, T$. 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 $t = 0, 1, \ldots, T-1$, then $E_{t-1:t+1}, g_{t-1:t+1}, L_{t-1:t+1}$, for $t = 1, 3, 5, \ldots, 2\lfloor (T-1)/2 \rfloor + 1$, etc. Once this has been done, we can then sample from $p(x_T \mid y_{0:T})$, then from $p(x_0 \mid y_{0:T}, x_T)$, 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.

Appendix B: Generalised statistical linear regression

We now describe how to linearise state-space models arising in Section 2 using the generalised statistical linear regression (GSLR) framework of [29, 74], which requires the existence of the first two conditional moments $\mathbb{E}[X_t \mid X_{t-1}]$ and $\mathbb{V}[X_t \mid X_{t-1}]$ of the transition model p_t. This approach comprises, as a special case, the extended and unscented linearisation methods of [42, 44]. For the sake of completeness, we also describe how to handle the potential $g(x_{0:T})$, when it is given as a product of observation models $h_t(y_t \mid x_t)$, in the same framework.

Following [74], we suppose that the first two conditional moments

$$m^X(x_{t-1}) \coloneqq \int x_t p_t(x_t \mid x_{t-1}) \mathrm{d}x_t, \tag{78}$$

$$V^X(x_{t-1}) \coloneqq \int (x_t - m^X(x_{t-1}))(x_t - m^X(x_{t-1}))^\top p_t(x_t \mid x_{t-1}) \mathrm{d}x_t, \tag{79}$$

and

$$m^Y(x_t) \coloneqq \int y_t h_t(y_t \mid x_t) \mathrm{d}y_t, \tag{80}$$

$$V^Y(x_t) \coloneqq \int (y_t - m^Y(x_t))(y_t - m^Y(x_t))^\top h_t(y_t \mid x_t) \mathrm{d}y_t, \tag{81}$$

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. As described in Section 2.1, 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 $x_{0:T} \in \mathbb{R}^{T \times d_x}$ 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 [74] 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:

$$\begin{aligned} p_t(z_t \mid z_{t-1}) &\approx \mathcal{N}(z_t; F_{t-1} z_{t-1} + b_{t-1}, Q_{t-1}), \\ h_t(y_t \mid z_t) &\approx \mathcal{N}(y_t; H_t z_t + c_t, R_t), \end{aligned} \tag{82}$$

with,

$$\begin{aligned} F_{t-1} &= C^X_{t-1} \Gamma^{-1}_{t-1}, & H_t &= C^Y_t \Gamma^{-1}_t, \\ b_{t-1} &= \mu^X_{t-1} - F_{t-1} x_{t-1}, & c_t &= \mu^Y_t - H_t x_t, \\ Q_{t-1} &= S^X_{t-1} - F_{t-1} \Gamma_{t-1} F^\top_{t-1}, & R_t &= S^Y_t - H_t \Gamma_t H^\top_t, \end{aligned} \tag{83}$$

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 (83) are in turn given by the general formulae

$$\begin{aligned} C^X_{t-1} &= \mathbb{C}\left[m^X(\zeta_{t-1}), \zeta_{t-1}\right], & C^Y_t &= \mathbb{C}\left[m^Y(\zeta_t), \zeta_t\right], \\ \mu^X_{t-1} &= \mathbb{E}\left[m^X(\zeta_{t-1})\right], & \mu^Y_t &= \mathbb{E}\left[m^Y(\zeta_t)\right], \\ S^X_{t-1} &= \mathbb{E}\left[V^X(\zeta_{t-1})\right] + \mathbb{V}\left[m^X(\zeta_{t-1})\right], & S^Y_t &= \mathbb{E}\left[V^Y(\zeta_t)\right] + \mathbb{V}\left[m^Y(\zeta_t)\right]. \end{aligned} \tag{84}$$

Clearly, the quantities in (84) 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., 68, Ch. 5].

Appendix C: Backward sampling and parallel-in-time particle Gibbs

For the sake of completeness, in this Section, we present details of the backward sampling method of [78], to be used instead of the genealogy selection step in Algorithm 3 for improved mixing. Additionally, we present the parallel-in-time

particle Gibbs algorithm [16, Section 3], specialised to the method of [27, see also Algorithm 4], in its auxiliary form presented in Section 3.2. The description extends to the gradient-informed proposals of Section 3.3.1 almost *verbatim* by corresponding a different auxiliary proposal mechanism.

C.1. Backward sampling

As discussed in Section 3.2, the genealogy selection step of Algorithm 3, lines 9 and beyond, can be replaced by a backward sampling step [78] to improve mixing. This modified version of Algorithm 3 is given in Algorithm 11 and is only implementable provided the quantity $p_t(x_t \mid x_{t-1})g_t(x_t, x_{t-1})$ can be evaluated pointwise. While other techniques exist when this is not the case [20], we focus on this method given all our examples verify this assumption. The algorithm is given in Algorithm 11. The algorithm can further be augmented to use different

Algorithm 11: Conditional SMC

Result: An updated trajectory $z_{0:T}$

Function $\mathit{cSMC}(x_{0:T}, N)$

  `// Forward propagation: same as Algorithm 3`

  `// Genealogy selection`

  Sample B_T with $\mathbb{P}(B_T^n = k) \propto w_T^k$ and set $z_T = X_T^{B_T}$

  for $t = T-1, \dots, 0$ **do**

    for $n = 1, \dots, N$ **do**

      Compute $\bar{w}^n = w_t^n g_{t+1}(z_{t+1}, X_t^n) p(z_{t+1} \mid X_t^n)$

    Sample B_t with $\mathbb{P}(B_t^n = k) \propto \bar{w}^k$ and set $z_t = X_t^{B_t}$

  return $z_{0:T}$

acceptance probabilities for the backward sampling step, as in [15], but we do not consider this here. Contrary to simple genealogy tracing, as implemented in Algorithm 3, backward sampling obtains mixing properties that do not degrade with the number of time steps T, even for a fixed number of particles $N \geq 2$ [2, 50, 46].

C.2. Parallel-in-time particle Gibbs

Consider the auxiliary model (32)

$$\begin{aligned}
&\pi(x_{0:T}, u_{0:T}) \\
&\quad \propto g_0(x_0)\, p_0(x_0) \left\{ \prod_{t=1}^{T} g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1}) \right\} \left\{ \prod_{t=0}^{T} \mathcal{N}\left(u_t; x_t, \frac{\delta_t}{2}\Sigma_t\right) \right\}, \\
&\quad \coloneqq \Gamma_0(x_0) \left\{ \prod_{t=1}^{T} \Gamma_t(x_t, x_{t-1}) \right\} \left\{ \prod_{t=0}^{T} \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}\Sigma_t\right) \right\},
\end{aligned} \tag{85}$$

for $\Gamma_t(x_t, x_{t-1}) = g_t(x_t, x_{t-1})\, p_t(x_t \mid x_{t-1})$ $t > 1$ and $\Gamma_0(x_0) = g_0(x_0)\, p_0(x_0)$. [16] then proceeds from the enabling recursion on "partial" smoothing distributions

$$\begin{aligned}\pi_{a:b}(x_{a:b} \mid u_{a:b}) &\coloneqq \frac{1}{L_{a:b}} \mathcal{N}(x_a; u_a, \frac{\delta_a}{2}\Sigma_a) \prod_{t=a+1}^{b} \Gamma_t(x_t, x_{t-1}) \mathcal{N}\left(x_t; u_t, \frac{\delta_t}{2}\Sigma_t\right) \\ &= \frac{L_{a:c-1} L_{c:b}}{L_{a:b}} \Gamma_c(x_c, x_{c-1}) \pi_{a:c-1}(x_{a:c-1} \mid u_{a:c-1}) \pi_{c:b}(x_{c:b} \mid u_{c:b}),\end{aligned} \tag{86}$$

and we have $\pi_{0:T}$ as the target distribution as well as $\pi_{a:a}(x_a \mid u_a) = \mathcal{N}(x_a; u_a, \frac{\delta_a}{2}\Sigma_a)$ for all $a = 1, \ldots, T$, and $\pi_{0:0}(x_0 \mid u_0) = \mathcal{N}(x_0; u_0, \frac{\delta_0}{2}\Sigma_0)\Gamma_0(x_0)$.[7]

The recursion (86) is then used to implement the parallel-in-time particle Gibbs algorithm. Indeed, if $\frac{1}{N}\sum_{n=1}^{N} \delta_{X^n_{a:c-1}}$ and $\frac{1}{N}\sum_{n=1}^{N} \delta_{X^n_{c:b}}$ are two independent Monte Carlo approximations of $\pi_{a:c-1}$ and $\pi_{c:b}$, respectively, then the 'stitched' empirical distribution

$$\sum_{m,n=1}^{N} W^{mn} \delta_{[X^m_{a:c-1}, X^n_{c:b}]}, \tag{87}$$

where

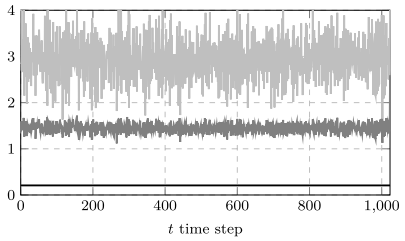
$$W^{mn} = \frac{\Gamma_c(X^n_c, X^m_{c-1})}{\sum_{i,j=1}^{N} \Gamma_c(X^j_c, X^i_{c-1})}, \tag{88}$$

is an approximation of $\pi_{a:b}$. We can then resample N trajectories out of the N^2 following the weights W^{mn} to then obtain an N-sized sample from $\pi_{a:b}$. The conditional version of this approach is then implemented similarly as for standard conditional SMC (Algorithm 3), by ensuring that one of the trajectories remains the current state of the Markov chain at each time step, until the last 'stitching' step where the genealogy is selected. For more details on the implementation of this algorithm, we refer the reader to [16, Section 3].

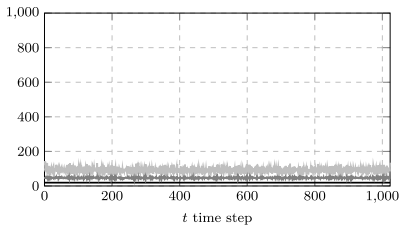
Appendix D: Sequential results for the spatio-temporal experiment of Section 4.2

We now report the sequential counterpart of the experiment run in Section 4.2. It is worth noting that the sequential and parallel implementations of the two Kalman samplers are fully equivalent and only differ in their actual implementation. Consequently, the expected squared jump distance for both should be (and is indeed) the same up to some variance coming from differences in generating the random variables for the sampling procedure. This is not the case for the cSMC implementations, and while their properties should be similar (from using both the same proposal mechanism), they are not expected to behave

[7] Or equivalently $\pi_{0:0}(x_0 \mid u_0) = \mathcal{N}(x_0; u_0, \frac{\delta_0}{2}\Sigma_0)$ and the weight Γ_0 is added to Γ_1: i.e., $\Gamma_1 \leftarrow \Gamma_0 \times \Gamma_1$.



(a) Expected squared jump distance for the sequential versions of 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) 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 6. *Average (across 20 different experiments) expected squared jump distance per iteration and second for all the sequential samplers considered on the spatio-temporal model* (44).

exactly similarly. The ESJD and ESJD per second are reported in Figure 6a and Figure 6b, respectively, where we have kept the same y-axis scale as in the parallel case for ease of comparison. As discussed already in Section 4.2, the non-sequential version are comparatively so much slower (up to 5 times as slow) in this instance than the parallel ones, that their comparative statistical performances are fully erased by their computational drawbacks.

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 method 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 [71]. Some 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. Finally, the authors would like to thank two anonymous reviewers for their constructive feedback.

Funding

Both authors gratefully acknowledge funding from the Academy of Finland, project 321891 (ADAFUME), and project 321900 (PARADIST). Adrien Coren-

flos also acknowledges the financial support provided by UKRI for OCEAN (a 2023-2029 ERC Synergy grant co-sponsored by UKRI).

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. MR2758115

[2] Andrieu, C., Lee, A. and Vihola, M. (2018). Uniform ergodicity of the iterated conditional SMC and geometric ergodicity of particle Gibbs samplers. *Bernoulli* **24** 842–872. MR3706778

[3] Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics* **37** 697–725. MR2502648

[4] Barfoot, T. D. (2017). *State estimation for robotics*. Cambridge University Press.

[5] Bell, B. M. (1994). The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization* **4** 626–636. MR1287820

[6] Besag, J. (1994). Comments on "Representations of knowledge in complex systems" by U. Grenander and M. I. Miller. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **56** 4. MR1293234

[7] Bezanson, J., Edelman, A., Karpinski, S. and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review* **59** 65–98. https://doi.org/10.1137/141000671 MR3605826

[8] Bierman, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation*. Academic Press. MR0453090

[9] Blelloch, G. E. (1989). Scans as primitive parallel operations. *IEEE Transactions on Computers* **38** 1526–1538.

[10] 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.

[11] Carter, C. and Kohn, R. (1994). On Gibbs sampling for state space models. *Biometrika* **81** 541–553. MR1311096

[12] Chen, Y., Davis, T. A., Hager, W. W. and Rajamanickam, S. (2008). Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* **35**. https://doi.org/10.1145/1391989.1391995 MR2738209

[13] Chopin, N. and Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo*. Springer. MR4215639

[14] Chopin, N. and Singh, S. S. (2015). On particle Gibbs sampling. *Bernoulli* **21** 1855–1883. MR3352064

[15] Chopin, N. and Singh, S. S. (2015). On particle Gibbs sampling (preprint v1). *arXiv preprint arXiv:1304.1887*. MR3352064

[16] 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. MR4577722

[17] Corenflos, A. and Finke, A. (2024). Particle-MALA and Particle-mGRAD: Gradient-based MCMC methods for high-dimensional state-space models. *arXiv preprint arXiv:2401.14868*.

[18] 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. MR3135540

[19] Crucinio, F. R. and Johansen, A. M. (2023). A divide and conquer sequential Monte Carlo approach to high dimensional filtering. *Statistica Sinica*. MR4764623

[20] Dau, H.-D. and Chopin, N. (2023). On backward smoothing algorithms. *The Annals of Statistics* **51** 2145–2169. MR4678799

[21] 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.

[22] Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer New York, New York, NY. MR2044973

[23] 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. MR3874301

[24] Dellaportas, P., Titsias, M. K., Petrova, K. and Plataniotis, A. (2023). Scalable inference for a full multivariate stochastic volatility model. *Journal of Econometrics* **232** 501–520. https://doi.org/10.1016/j.jeconom.2021.09.013 MR4539497

[25] 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. MR2895411

[26] Doucet, A. (2010). A Note on Effcient Conditional Simulation of Gaussian Distributions Technical Report, University of British Columbia.

[27] Finke, A. and Thiery, A. H. (2023). Conditional sequential Monte Carlo in high dimensions. *The Annals of Statistics* **51** 437–463. https://doi.org/10.1214/22-AOS2252 MR4600988

[28] Frühwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *Journal of Time Series Analysis* **15** 183–202. MR1263889

[29] García-Fernández, Á. F., Svensson, L. and Särkkä, S. (2017). Iterated posterior linearization smoother. *IEEE Transactions on Automatic Control* **62** 2056–2063. MR3636361

[30] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A. and Rubin, D. B. (2013). *Bayesian Data Analysis*. CRC Press. https://doi.org/10.1201/b16018 MR3235677

[31] 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.

[32] Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *Interface Proceedings*.
[33] Geyer, C. J. (1992). Practical Markov Chain Monte Carlo. *Statistical Science* **7** 473–483.
[34] 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. MR2814492
[35] 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. MR2054295
[36] 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. MR2054295
[37] 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.
[38] Guarniero, P., Johansen, A. M. and Lee, A. (2017). The Iterated Auxiliary Particle Filter. *Journal of the American Statistical Association* **112** 1636–1647. MR3750887
[39] Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* **57(1)** 97–109. MR3363437
[40] Hestenes, M. R., Stiefel, E. et al. (1952). *Methods of conjugate gradients for solving linear systems* **49**. NBS Washington, DC. MR0060307
[41] Hillis, W. D. and Steele Jr, G. L. (1986). Data parallel algorithms. *Communications of the ACM* **29** 1170–1183.
[42] Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Academic Press.
[43] 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.
[44] Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* **92** 401–422.
[45] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering* **82** 35–45. MR3931993
[46] Karjalainen, J., Lee, A., Singh, S. S. and Vihola, M. (2024). Mixing time of the conditional backward sampling particle filter. *arXiv preprint arXiv:2312.17572*. MR4152635
[47] Karppinen, S., Singh, S. S. and Vihola, M. (2024). Conditional particle filters with bridge backward sampling. *Journal of Computational and Graphical Statistics* **33** 364–378. MR4754798
[48] Kitagawa, G. (1996). Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics* **5** 1–25. MR1380850

[49] Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics* **22** 79–86. https://doi.org/10.1214/aoms/1177729694 MR0039968

[50] Lee, A., Singh, S. S. and Vihola, M. (2020). Coupled conditional backward sampling particle filter. *The Annals of Statistics* **48** 3066–3089. MR4152635

[51] Leisen, F. and Mira, A. (2008). An extension of Peskun and Tierney orderings to continuous time Markov chains. *Statistica Sinica* 1641–1651. MR2469328

[52] Lindsten, F., Bunch, P., Singh, S. S. and Schön, T. B. (2015). Particle ancestor sampling for near-degenerate or intractable state transition models. *arXiv preprint arXiv:1505.06356*.

[53] Lindsten, F., Jordan, M. I. and Schön, T. B. (2014). Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research* **15** 2145–2184. MR3231604

[54] Lindsten, F., Schön, T. and Jordan, M. (2012). Ancestor sampling for particle Gibbs. *Advances in Neural Information Processing Systems* **25**.

[55] Malory, S. J. (2021). *Bayesian Inference for Stochastic Processes.* Lancaster University (United Kingdom). MR4336613

[56] 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.

[57] Mider, M., Schauer, M. and Van der Meulen, F. (2021). Continuous-discrete smoothing of diffusions. *Electronic Journal of Statistics* **15** 4295–4342. MR4312205

[58] Müller, P. (1993). Alternatives to the Gibbs sampling scheme Technical Report, Institute of Statistics and Decision Sciences, Duke Univ.

[59] 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. MR1847807

[60] Neal, R. M. (2003). Markov Chain Sampling for Non-linear State Space Models Using Embedded Hidden Markov Models.

[61] NVIDIA, Vingelmann, P. and Fitzek, F. H. P. (2022). CUDA, release: 11.8.x.

[62] Pasarica, C. and Gelman, A. (2010). Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica* 343–364. MR2640698

[63] Peskun, P. H. (1973). Optimum Monte-Carlo sampling using Markov chains. *Biometrika* **60** 607–612. MR0362823

[64] Rauch, H. E., Tung, F. and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA journal* **3** 1445–1450. MR0181489

[65] Rennich, S. C., Stosic, D. and Davis, T. A. (2016). Accelerating sparse Cholesky factorization on GPUs. *Parallel Computing* **59** 140–150. Theory and Practice of Irregular Applications. https://doi.org/10.1016/j.parco.2016.06.004 MR3566860

[66] Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli* **2** 341–363. MR1440273

[67] Särkkä, S. and García-Fernández, Á. F. (2021). Temporal parallelization of Bayesian smoothers. *IEEE Transactions on Automatic Control* **66** 299–306. MR4210413

[68] Särkkä, S. and Svensson, L. (2023). *Bayesian filtering and smoothing* **17**. Cambridge university press. MR4692992

[69] Singh, S. S., Lindsten, F. and Moulines, E. (2017). Blocking strategies and stability of particle Gibbs samplers. *Biometrika* **104** 953–969. MR3737314

[70] Tierney, L. (1998). A note on Metropolis-Hastings kernels for general state spaces. *Annals of applied probability* 1–9. MR1620401

[71] 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. MR3849342

[72] Tjelmeland, H. (2004). Using all Metropolis–Hastings proposals to estimate mean values Technical Report, NTNU.

[73] Tronarp, F. (2020). Iterative and Geometric Methods for State Estimation in Non-linear Models, PhD thesis, Aalto University.

[74] 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.

[75] Van Der Merwe, R., Doucet, A., De Freitas, N. and Wan, E. (2000). The unscented particle filter. *Advances in neural information processing systems* **13**.

[76] Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[77] 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.

[78] Whiteley, N. (2010). Discussion on particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B* **72** 306–307.

[79] 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.

[80] Yaghoobi, F., Corenflos, A., Hassan, S. and Särkkä, S. (2025 (to appear)). Parallel square-root statistical linear regression for inference in nonlinear state-space models. *SIAM Journal on Scientific Computing*.