Efficient Bayesian Inference for Switching State-Space Models using Particle Markov Chain Monte Carlo Methods

Nick Whiteley, Christophe Andrieu
Department of Mathematics,
University of Bristol,
University Walk,
Bristol BS8 1TW, UK.
Email: {Nick.Whiteley,C.Andrieu}@bris.ac.uk

Arnaud Doucet
Department of Statistics,
University of British Columbia,
Vancouver V6T 1Z2, BC, Canada.
Email: Arnaud@stat.ubc.ca

June 15, 2010

Abstract

Switching state-space models (SSSM) are a popular class of time series models that have found many applications in statistics, econometrics and advanced signal processing. Bayesian inference for these models typically relies on Markov chain Monte Carlo (MCMC) techniques. However, even sophisticated MCMC methods dedicated to SSSM can prove quite inefficient as they update potentially strongly correlated variables one-at-a-time. Particle Markov chain Monte Carlo (PMCMC) methods are a recently developed class of MCMC algorithms which use particle filters to build efficient proposal distributions in high-dimensions [1]. The existing PMCMC methods of [1] are applicable to SSSM, but are restricted to employing standard particle filtering techniques. Yet, in the context of SSSM, much more efficient particle techniques have been developed [22, 23, 24]. In this paper, we extend the PMCMC framework to enable the use of these efficient particle methods within MCMC. We demonstrate the resulting generic methodology on a variety of examples including a multiple change-points model for well-log data and a model for U.S./U.K. exchange rate data. These new PMCMC algorithms are shown to outperform experimentally state-of-the-art MCMC techniques for a fixed computational complexity. Additionally they can be easily parallelized [39] which allows further substantial gains.

Keywords: Bayesian inference, Markov chain Monte Carlo, optimal resampling, particle filters, sequential Monte Carlo, switching state-space models.

1 Introduction

Linear Gaussian Switching State-Space Models (SSSM) are a class of time series models in which the parameters of a linear Gaussian model switch according to a discrete latent process. They are ubiquitous in statistics [7, 32], econometrics [37, 35] and advanced signal processing [3, 12] as they allow us to describe in a compact and interpretable way regime switching time series. SSSM have been successfully used to describe, among others, multiple change-point models [23, 34], nonparametric regression models with outliers [9] and Markov switching autoregressions [5, 32, 37].

Performing Bayesian inference for SSSM requires the use of Markov chain Monte Carlo (MCMC) techniques. The design of efficient sampling techniques for this class of models has been a subject of active research for over 15 years, dating back at least as far as [8, 41]. A recent overview of MCMC in this context can be found in [7, 32]. The main practical difficulty lies in simulating from the conditional distribution of the trajectory of the discrete-valued latent process. The cost of computing this distribution grows exponentially in the length of the observation record and therefore obtaining an exact sample from it is impractical for all but tiny data sets. A standard strategy is instead to update the components of the discrete latent process one-at-a-time [9, 33, 34]. However, it is well-known that such an approach can significantly slow down the convergence of MCMC algorithms. An alternative is to sample approximately from the joint distribution of the latent discrete trajectory using particle filters: non-iterative techniques based on a combination of importance sampling and resampling techniques, see [17, 40] for a review of the literature. Empirical evidence suggests that particle filters are able to provide samples whose distribution is close to the target distribution of interest and this evidence is backed up by the rigourous quantitative bounds established in [14, chapter 8]. This motivates using particle filters as proposal distributions within MCMC.

This idea is very natural, but its realization is far from trivial as the distribution of a sample generated by a particle filter does not admit a closed-form expression hence preventing us from directly using the standard Metropolis-Hastings (MH) algorithm. In a recent paper [1], it has been shown that it is possible to bypass this problem. The authors have proposed a whole class of MCMC algorithms named Particle MCMC (PMCMC) relying on proposals built using particle filters. These algorithms have been demonstrated in the context of non-linear non-Gaussian state-space models and are directly applicable to SSSM; see also [29] for applications in financial econometrics. However, the standard particle methods employed in [1] do not fully exploit the discrete nature of the latent process in SSSM. This was recognized early by Paul Fearnhead who proposed an alternative generic algorithm, which we refer to as the Discrete Particle Filter (DPF) [22]. The DPF bypasses the importance sampling step of standard particle techniques and can be interpreted as using a clever random pruning mechanism to select support points from the exponentially growing sequence of discrete latent state spaces. The DPF methodology has been demonstrated successfully in a variety of applications [7, 22, 23, 24]. It has been shown to significantly outperform alternative approaches such as the Rao-Blackwellized particle filters developed in [10, 19] for a fixed computational complexity.

The main contribution of this article is to extend the PMCMC methodology to allow us to use the efficient DPF as a proposal distribution for this important class of statistical models. The practical efficiency of the proposed methods relies on a new backward sampling procedure. We show that on a variety of applications this new generic methodology outperforms state-of-the-art MCMC algorithms for a fixed computational complexity. Moreover, as in the case of standard particle filters [39], the DPF can be parallelized easily. This suggests that even greater computational gains can be achieved.

The rest of the paper is organised as follows. In Section 2 we present the general class of SSSM considered in this paper and give some illustrative examples. In Section 3, we discuss the intractability of exact inference in SSSM and present the DPF algorithm [22, 23, 24]. Our presentation is slightly non-standard and explicitly introduces the random support sets generated by the algorithm. This allows us to describe the DPF precisely and compactly in a probabilistic way which proves useful to establish the validity of the PMCMC algorithms. We also review standard MCMC techniques used in this context. In Section 4 we introduce original PMCMC algorithms relying on the DPF to perform inference in SSSM and discuss some theoretical results. In Section 5, we discuss generic practical issues and demonstrate the efficiency of the proposed methods in the context of three examples. Finally in Section 6 we discuss several extensions of this work.

2 Switching state-space models

2.1 Model

From herein, we use the standard convention whereby capital letters are used for random variables while lower case letters are used for their values. Hereafter for any generic process $\{z_n\}$ we will denote $z_{i:j} := (z_i, z_{i+1}, \ldots, z_j)$. The identity matrix of size p is denoted I_p and the matrix of zeros of size $p \times q$ by $0_{p\times q}$.

Consider the following SSSM, also known in the literature as a conditionally linear Gaussian state-space model or a jump linear system. The latent state process $\{X_n\}_{n\geq 1}$ is such that X_n takes values in a **finite** set $\mathcal{X}$. It is characterized by its initial distribution $X_1 \sim \nu_\theta(\cdot)$ and transition probabilities for $n > 1$

$$X_n|(X_{1:n-1} = x_{1:n-1}) \sim f_\theta(\cdot|x_{1:n-1}). \tag{1}$$

Conditional upon $\{X_n\}_{n\geq 1}$, we have a linear Gaussian state-space model defined through $Z_0 \sim \mathcal{N}(m_0, \Sigma_0)$ and for $n \geq 1$

$$Z_n = A_\theta(X_n)Z_{n-1} + B_\theta(X_n)V_n + F_\theta(X_n)u_n, \tag{2}$$
$$Y_n = C_\theta(X_n)Z_n + D_\theta(X_n)W_n + G_\theta(X_n)u_n, \tag{3}$$

where $\mathcal{N}(m, \Sigma)$ is the normal distribution of mean m and covariance Σ, $V_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0_{v\times 1}, I_v)$, $W_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0_{w\times 1}, I_w)$, $\{A_\theta(x), B_\theta(x), C_\theta(x), D_\theta(x), F_\theta(x), G_\theta(x); x \in \mathcal{X}\}$ are matrices of appropriate dimension and u_n is an exogeneous input. Here $\theta \in \Theta$ is some static parameter which may be multidimensional, for example $\Theta \subset \mathbb{R}^d$. For purposes of precise specification of resampling algorithms in the sequel and without loss of generality we label the elements of $\mathcal{X}$ with numbers, for example $\mathcal{X} = \{1, ..., |\mathcal{X}|\}$ for some $|\mathcal{X}| \in \mathbb{N}$. We may then endow each Cartesian product space $\mathcal{X}^2, \mathcal{X}^3, ...$ with the corresponding lexicographical order relation. From henceforth, whenever we refer to ordering of a set of points in $\mathcal{X}^n$ it is with respect to the latter relation.

We give here a simple example of a SSSM. Two more sophisticated examples are discussed in Section 5.

2.1.1 Example: Auto-regression with shifting level

Let $\mathcal{X} = \{0, 1\}$ and for $\{X_n\}$ a Markov chain on $\mathcal{X}$ with transition matrix P_X, consider the process defined by

$$\begin{aligned} Y_n &= \mu_n + \phi(Y_{n-1} - \mu_{n-1}) + \sigma V_{n,1} \\ \mu_n &= \mu_{n-1} + \sigma X_n V_{n,2}, \end{aligned}$$

where for each $n \geq 1$, μ_n and Y_n are real-valued and $\{V_{n,1}\}$ and $\{V_{n,2}\}$ are i.i.d. $\mathcal{N}(0, 1)$. The initial distribution on μ_0 is $\mathcal{N}(m_0, \sigma_0^2)$. This is a natural generalization of a first order autoregressive model to the case where the level μ_n is time-varying with shifts driven by the latent process $\{X_n\}$. This model can be expressed in state-space form by setting

$$Z_n = \begin{bmatrix} Y_n - \mu_n \\ \mu_n \end{bmatrix}, \quad A_\theta(x_n) = \begin{bmatrix} \phi & 0 \\ 0 & 1 \end{bmatrix} \quad \forall x_n,$$

$$B_\theta(x_n) = \sigma \begin{bmatrix} 1 & 0 \\ 0 & x_n \end{bmatrix}, \quad C_\theta(x_n) = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad D_\theta(x_n) = F_\theta(x_n) = G_\theta(x_n) = 0, \quad \forall x_n.$$

The unknown parameters of this model are $\theta = [\phi\ \sigma^2\ P_X]$. In this model and more generally in SSSMs, inferences about the latent processes $\{\mu_n\}$ and $\{X_n\}$ from a particular data set are likely to be highly sensitive to values of these parameters if they are assumed known.

2.2 Inference aims

Our aim is to perform Bayesian inference in SSSMs, conditional upon some observations $y_{1:T}$ and for some $T \geq 1$, treating both the latent trajectories $\{X_n\}$, $\{Z_n\}$ and the parameter θ as unknowns. Where applicable, the values of the input sequence $u_{1:T}$ are assumed known, but for clarity we suppress them from our notation. We ascribe a prior density $p(\theta)$ to θ so Bayesian inference relies on the joint density

$$p(\theta, x_{1:T}, z_{0:T}|y_{1:T}) \propto p_\theta(x_{1:T}, z_{0:T}, y_{1:T})\, p(\theta), \tag{4}$$

where the definition of $p_\theta(x_{1:T}, z_{0:T}, y_{1:T})$ follows from Eq. (1)-(2)-(3). This posterior can be factorized as follows

$$p(\theta, x_{1:T}, z_{0:T}|y_{1:T}) = p(\theta, x_{1:T}|y_{1:T})\, p_\theta(z_{0:T}|y_{1:T}, x_{1:T}) \tag{5}$$

where

$$p(\theta, x_{1:T}|y_{1:T}) = \frac{p_\theta(y_{1:T}|x_{1:T})\, p(x_{1:T}|\theta)p(\theta)}{\int_\Theta \sum_{x'_{1:T}\in\mathcal{X}^T} p_\theta(y_{1:T}|x'_{1:T})\, p(x'_{1:T}|\theta)p(\theta)\, \mathrm{d}\theta}. \tag{6}$$

Conditional upon $X_{1:T} = x_{1:T}$, Eq. (2)-(3) define a linear Gaussian state-space model so it is possible to compute efficiently the statistics of the conditional multivariate Gaussian density $p_\theta(z_{0:T}|\, y_{1:T}, x_{1:T})$ in Eq. (5) and the conditional marginal likelihood $p_\theta(y_{1:T}|\, x_{1:T})$ in Eq. (6) using Kalman techniques. For example $p_\theta(y_{1:T}|\, x_{1:T})$ can be computed using the product of predictive densities

$$p_\theta(y_{1:T}|\, x_{1:T}) = \prod_{n=1}^{T} g_\theta(y_n|\, y_{1:n-1}, x_{1:n}) \tag{7}$$

where $y_{1:0} := \varnothing$. The statistics of these Gaussian predictive densities can be computed using the Kalman filter which is recalled in Appendix A for sake of convenience. For simplicity of presentation throughout the following we assume that for each $1 \leq n \leq T$ and $\theta \in \Theta$ the support of $p_\theta(x_{1:n}|y_{1:n})$ is $\mathcal{X}^n$. This assumption is satisfied in the vast majority of cases considered in practice and in all the examples we consider. The techniques discussed below can be transferred to cases where this assumption is not met with only cosmetic changes.

3 Inference techniques for switching state-space models

3.1 Exact Inference and Intractability

The main difficulty faced in the exact computation of $p(\theta, x_{1:T}|y_{1:T})$, is the need to perform the summation in the denominator of Eq. (6) over up to $|\mathcal{X}|^T$ values of $x_{1:T}$, where $|\mathcal{X}|$ is the cardinality of $\mathcal{X}$. For even modest values of T, this sum is too expensive to compute exactly. In the applications we consider, T is of the order of thousands, so exact computation is practically impossible.

Even if θ is treated as fixed, inference is intractable. In this case, we wish to compute $p_\theta(x_{1:T}|y_{1:T})$, whose normalization involves the same problematic summation. One approach is to obtain $p_\theta(x_{1:T}|y_{1:T})$ by sequential computation of $p_\theta(x_1|y_1), p_\theta(x_{1:2}|y_{1:2}), \ldots$ via the recursive relationship

$$p_\theta(x_{1:n}|y_{1:n}) = \frac{g_\theta(y_n|y_{1:n-1}, x_{1:n}) f_\theta(x_n|x_{1:n-1}) p_\theta(x_{1:n-1}|y_{1:n-1})}{\sum_{x_{1:n}\in\mathcal{X}^n} g_\theta(y_n|y_{1:n-1}, x_{1:n}) f_\theta(x_n|x_{1:n-1}) p_\theta(x_{1:n-1}|y_{1:n-1})},$$

but the computation involved increases exponentially in n. For purposes of exposition in the sequel, we remark that, as for each n the support of $p_\theta(x_{1:n}|y_{1:n})$ is $\mathcal{X}^n$ then the sequence of such supports satisfies the trivial recursion:

$$\mathcal{X}^n = \mathcal{X} \times \mathcal{X}^{n-1},$$

and is evidently growing in cardinality with n. Hence, in both the cases of computing $p(\theta, x_{1:T}|y_{1:T})$ and $p_\theta(x_{1:T}|y_{1:T})$ it is necessary to rely on Monte Carlo methods.

3.2 Monte Carlo Methods

We next review two classes of Monte Carlo techniques to perform inference in SSSM. The first method we discuss is the DPF algorithm of Fearnhead [22]. For a fixed parameter value θ, this algorithm allows us to compute an approximation of the posterior distribution $p_\theta(x_{1:T}|y_{1:T})$ and an approximation of the marginal likelihood $p_\theta(y_{1:T})$. We present this algorithm in a slightly non-standard way which allows us to describe it probabilistically in a concise and precise manner. This will prove useful for the development of PMCMC algorithms in Section 4. We also review MCMC methods which have been developed to approximate $p(\theta, x_{1:T}, z_{0:T}|y_{1:T})$ and discuss their advantages and limitations.

3.3 The discrete particle filter

The DPF algorithm proposed in [22, 23] is a non-iterative procedure approximating the posterior distribution $p_\theta(x_{1:T}|y_{1:T})$ and the marginal likelihood $p_\theta(y_{1:T})$. Practically, the DPF approximation of the posterior distributions $\{p_\theta(x_{1:n}|y_{1:n}); n \geq 1\}$ is made sequentially in time using a collection of $N|\mathcal{X}|$ weighted trajectories or "particles" $\left\{X_{1:n}^{(i)}; i = 1, ..., N|\mathcal{X}|\right\}$,

$$\widehat{p}_\theta^N(x_{1:n}|y_{1:n}) = \sum_{i=1}^{N|\mathcal{X}|} W_n^\theta\left(X_{1:n}^{(i)}\right)\delta_{X_{1:n}^{(i)}}(x_{1:n}),\ W_n^\theta\left(X_{1:n}^{(i)}\right) \geq 0,\ \sum_{i=1}^{N|\mathcal{X}|} W_n^\theta\left(X_{1:n}^{(i)}\right) = 1.$$

The parameter N controls the precision of the algorithm. The larger it is, the more accurate (on average) the approximation of the target distribution. It has been demonstrated experimentally in [7, 23, 24] that the DPF algorithm outperforms significantly, sometimes by one order of magnitude, the Rao-Blackwellized particle filters proposed in [10, 18, 19] and that it is able to provide very good approximations of $p_\theta(x_{1:T}|y_{1:T})$ in realistic scenarios even with a moderate number of particles. The action of the DPF can be summarised as follows.

Assume that we have, at time step n obtained $\widehat{p}_\theta^N(x_{1:n}|y_{1:n})$ consisting of $N|\mathcal{X}|$ distinct particles with weights that sum to 1. A resampling step is then applied, exactly N of the $N|\mathcal{X}|$ trajectories survive and their weights are adjusted accordingly. The resampling mechanism is chosen in such a way as to be optimal in some sense. Throughout the remainder of the paper we treat the case of minimising the sum of variances of the importance weights as in [23] but exactly the same method applies to other schemes discussed in [3]. Features of this resampling scheme which distinguish it from standard methods, such as multinomial resampling, are that it results in no duplicated particles and gives post-resampling weights which are non-uniform.

Whereas standard particle methods rely on a stochastic proposal mechanism to explore the space, the DPF performs all its exploration deterministically. This is possible because of the finite cardinality of the latent discrete space. Consider one of N particles which survived the resampling operation, each of which is a point in $\mathcal{X}^n$. Call the point in question $x_{1:n}$ and denote by $m_{n|n}^{z,\theta}(x_{1:n})$ and $\Sigma_{n|n}^{z,\theta}(x_{1:n})$ respectively the mean and covariance of the Gaussian density $p_\theta(z_n|y_{1:n}, x_{1:n})$. From this point $|\mathcal{X}|$ new particles $\{(x_{1:n}, x); x \in \mathcal{X}\}$ are formed, and for each one of them, $m_{n+1|n+1}^{z,\theta}(x_{1:n}, x)$, $\Sigma_{n+1|n+1}^{z,\theta}(x_{1:n}, x)$ and the associated unnormalized weight are calculated using the Kalman filtering recursions (included for reference in Appendix (A)). This procedure is repeated for the remaining $N-1$ particles, resulting in $N|\mathcal{X}|$ weighted trajectories. The weights are then normalized to yield a probability distribution constituting $\widehat{p}_\theta^N(x_{1:n+1}|y_{1:n+1})$.

This outline of the DPF operations highlights the function of the resampling step: in the case of the DPF it acts to prune the exponentially growing (in n) tree of possible paths $\{x_{1:n} \in \mathcal{X}^n; n = 1, 2, ...\}$. It is convenient to specify the DPF in a slightly non-standard way which highlights that the only randomness in this algorithm arises from the resampling step. To this end, we introduce random support sets $\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_T$ with each $\mathbf{S}_n$ taking a value $\mathbf{s}_n$ which is a subset of $\mathcal{X}^n$. It is stressed that, in the following interpretation, the $x_{1:n}$'s are not random variables, and are just points in the state space (and Cartesian products thereof) used for indexing. With this notation, we write the DPF approximation for $n > 1$ as

$$\widehat{p}_\theta^N(x_{1:n}|y_{1:n}) = \sum_{x'_{1:n} \in \mathbf{S}_n} W_n^\theta(x'_{1:n})\,\delta_{x'_{1:n}}(x_{1:n}). \tag{8}$$

Under the probability law of the DPF algorithm, which we discuss in more detail later, for each $n \geq 2$, $|\mathbf{S}_n| = N|\mathcal{X}|$, with probability 1. We thus see in (8) the effect of the parameter N: it specifies the number of support points of the approximation $\widehat{p}_\theta^N(x_{1:n}|y_{1:n})$. We next provide pseudo code for the DPF algorithm and then go on to discuss several issues related to its practical use and its theoretical representation.

DPF algorithm

At time $n=1$

• Set $\mathbf{S}_1 = \mathcal{X}$ and for each $x_1 \in \mathcal{X}$, compute $m^{z,\theta}_{1|1}(x_1)$, $\Sigma^{z,\theta}_{1|1}(x_1)$ and $g_\theta(y_1|x_1)$ using the Kalman filter.

• Compute and normalise the weights. For each $x_1 \in \mathcal{X}$,

$$\overline{w}^\theta_1(x_1) = \nu_\theta(x_1)\, g_\theta(y_1|\, x_1)\,,\ W^\theta_1(x_1) = \frac{\overline{w}^\theta_1(x_1)}{\sum_{x'_1\in\mathcal{X}} \overline{w}^\theta_1(x'_1)}. \tag{9}$$

At times $n = 2, ..., T$

• If $|\mathbf{S}_{n-1}| \leq N$ set $C_{n-1} = \infty$ otherwise set C_{n-1} to the unique solution of

$$\sum_{x_{1:n-1}\in\mathbf{S}_{n-1}} 1 \wedge C_{n-1} W^\theta_{n-1}(x_{1:n-1}) = N.$$

• Maintain the L_{n-1} trajectories in $\mathbf{S}_{n-1}$ which have weights strictly superior to $1/C_{n-1}$, then apply the stratified resampling mechanism to the other trajectories to yield $N - L_{n-1}$ survivors. Set $\mathbf{S}'_{n-1}$ to the set of surviving and maintained trajectories.

• Set $\mathbf{S}_n = \mathbf{S}'_{n-1} \times \mathcal{X}$.

• For each $x_{1:n} \in \mathbf{S}_n$, compute $m^{z,\theta}_{n|n}(x_{1:n})$, $\Sigma^{z,\theta}_{n|n}(x_{1:n})$ and $g_\theta(y_n|y_{1:n-1}, x_{1:n})$ using the Kalman filter.

• Compute and normalise the weights. For each $x_{1:n} \in \mathbf{S}_n$,

$$\overline{w}^\theta_n(x_{1:n}) = f_\theta(x_n|x_{1:n-1}) g_\theta(y_n|y_{1:n-1}, x_{1:n}) \frac{W^\theta_{n-1}(x_{1:n-1})}{1 \wedge C_{n-1} W^\theta_{n-1}(x_{1:n-1})}, \tag{10}$$

$$W^\theta_n(x_{1:n}) = \frac{\overline{w}^\theta_n(x_{1:n})}{\sum_{x'_{1:n}\in\mathbf{S}_n} \overline{w}^\theta_n(x'_{1:n})}. \tag{11}$$

3.3.1 Exact computation at the early iterations

For small n it is practically possible to compute $p_\theta(x_{1:n}|y_{1:n})$ exactly. It is only once n is large enough that $|\mathcal{X}^n| > N$ that we need to employ the resampling mechanism to prune the set of trajectories. This action is represented conceptually in the DPF algorithm above by the artifice of setting $C_n = \infty$ if n is such that $|\mathbf{S}_{n-1}| \leq N$. When this condition is satisfied, the resampling step is not called into action. Of course in the practically unrealistic case that $\left|\mathcal{X}^T\right| \leq N$ the DPF, unlike standard SMC algorithms, thus reduces to exact recursive computation of $\{p_\theta(x_{1:n}|y_{1:n}); n = 1, ..., T\}$.

3.3.2 Computing C_n and stratified resampling

The threshold C_n is a deterministic function of the weights $\left\{W^\theta_n(x_{1:n})\right\}_{x_{1:n}\in\mathbf{S}_n}$. A method for solving $\sum_{x_{1:n}\in\mathbf{S}_n} 1 \wedge C_n W^\theta_n(x_{1:n}) = N$ is given in [23]. The stratified resampling mechanism, which is employed once C_n has been computed, proceeds as follows at time n; this was originally proposed in [6, 38], although not in the context of the DPF.

Stratified resampling

• Normalise the weights $\overline{w}^\theta_{n-1}(x_{1:n-1})$ of the $N|\mathcal{X}| - L_{n-1}$ particles and label them according to the order of the corresponding $x_{1:n-1}$ to obtain $\widehat{W}^\theta_{n-1}\left(x^{(i)}_{1:n-1}\right)$; $i = 1, ..., N|\mathcal{X}| - L_{n-1}$.

• Construct the corresponding cumulative distribution function: for $i = 1, ..., N|\mathcal{X}| - L_{n-1}$,

$$Q^\theta_{n-1}(i) := \sum_{j\leq i} \widehat{W}^\theta_{n-1}\left(x^{(j)}_{1:n-1}\right), \qquad Q^\theta_{n-1}(0) := 0.$$

• Sample U_1 uniformly on $[0, 1/(N - L_{n-1})]$ and set $U_j = U_1 + \frac{j-1}{N-L_{n-1}}$ for $j = 2, ..., N - L_{n-1}$.

• For $i = 1, ..., N|\mathcal{X}| - L_{n-1}$, if there exists $j \in \{1, ..., N - L_{n-1}\}$ such that $Q^\theta_{n-1}(i-1) < U_j \leq Q^\theta_{n-1}(i)$, then $x^{(i)}_{1:n-1}$ survives.

3.3.3 Computational Requirements

Assuming that the cost of evaluating $f_\theta(x_n|x_{1:n-1})$ is $\mathcal{O}(1)$ for all n, the computational complexity of the DPF is $\mathcal{O}(|\mathcal{X}|N)$ at each time step due to the propagation of $N|\mathcal{X}|$ Kalman filtering operations and the generation of a single uniform random variable. The parallelisation techniques described in [39] could readily be exploited when performing the Kalman computations.

3.3.4 Estimating $p_\theta(y_{1:T})$

Of particular interest in the sequel is the fact that the DPF provides us with an estimate of the marginal likelihood $p_\theta(y_{1:T})$ given by

$$\widehat{p}_\theta(y_{1:T}) := \widehat{p}_\theta(y_1) \prod_{n=2}^{T} \widehat{p}_\theta(y_n|y_{1:n-1}) \tag{12}$$

where

$$\widehat{p}_\theta(y_n|y_{1:n-1}) = \sum_{x_1 \in \mathcal{X}} \overline{w}_1^\theta(x_1), \quad \widehat{p}_\theta(y_n|y_{1:n-1}) = \sum_{x_{1:n} \in \mathbf{S}_n} \overline{w}_n^\theta(x_{1:n}), \quad n > 1. \tag{13}$$

Inevitably, for fixed N, the quality of the particle approximation to the distribution $p_\theta(x_{1:T}|y_{1:T})$ decreases as T increases. For fixed T, once N is larger than $\left|\mathcal{X}^T\right|$, the DPF computes $p_\theta(y_{1:T})$ exactly.

Before introducing the details of the new PMCMC algorithms, we review some existing MCMC algorithms for performing inference in SSSM.

3.4 Standard Markov chain Monte Carlo methods

Designing efficient MCMC algorithms to sample from $p(\theta, x_{1:T}, z_{0:T}|y_{1:T})$ is a difficult task. Most existing MCMC methods approach this problem using some form of Gibbs sampler and can be summarized as cycling in some manner through the sequence of distributions $p(\theta|y_{1:T}, x_{1:T}, z_{0:T})$, $p_\theta(z_{0:T}|y_{1:T}, x_{1:T})$ and $p_\theta(x_{1:T}|y_{1:T}, z_{0:T})$ or $p_\theta(x_{1:T}|y_{1:T})$.

Sampling efficiently from $p(\theta|y_{1:T}, x_{1:T}, z_{0:T})$ is often feasible due to the small or moderate size of θ and the fact that for many models and parameters of interest, conjugate priors are available. When conjugate priors are not used, Metropolis-within-Gibbs steps may be applied.

A variety of efficient algorithms have been developed to sample from $p_\theta(z_{0:T}|y_{1:T}, x_{1:T})$. These methods rely on the conditionally linear-Gaussian structure of the model and involve some form of forward filtering backward sampling recursion [8, 30]. Variants of these schemes which approach the task by explicitly sampling the state disturbances may be more efficient and/or numerically stable for some classes of models [13, 20]. In all the numerical examples we consider, sampling from $p_\theta(z_{0:T}|y_{1:T}, x_{1:T})$ was performed using the simulation smoother of [20].

Sampling from $p_\theta(x_{1:T}|y_{1:T}, z_{0:T})$ can also be performed efficiently using a forward filtering backward sampling recursion [8, 11] when $\{X_n\}$ is a Markov chain. The resulting Gibbs sampler is elegant but it can mix very slowly as $X_{1:T}$ and $Z_{0:T}$ are usually strongly correlated. To bypass this problem, the authors in [9, 33] proposed to integrate out $Z_{0:T}$ using the Kalman filter as discussed in Subsection 2.2. However, as mentioned in the introduction, exact sampling from $p_\theta(x_{1:T}|y_{1:T})$ is typically infeasible as the cost of computing this distribution is exponential in T. Therefore, in the algorithms of [9, 33], the discrete variables $X_{1:T}$ are updated one-at-a-time according to their full conditional distributions $p_\theta(x_n|y_{1:T}, x_{1:n-1}, x_{n+1:T})$. It was shown in [9, 33] that this strategy can improve performance drastically compared to algorithms where $X_{1:T}$ is updated conditional upon $Z_{0:T}$. From hereon we refer to the Gibbs sampler of [33] as the "standard Gibbs" algorithm.

At this stage, we comment a little further on the method of [33] as it is relevant to the new algorithms described in the later sections. The Gibbs sampler of [33] achieves a sweep of samples from $p_\theta(x_1|y_{1:T}, x_{2:T})$, $p_\theta(x_2|y_{1:T}, x_1, x_{3:T})$, etc. by a "backward–forward" procedure exploiting the identities

$$p_\theta(x_n|y_{1:T}, x_{1:n-1}, x_{n+1:T}) \propto p_\theta(y_n|y_{1:n-1}, x_{1:n}) p_\theta(x_n|x_{1:n-1}, x_{n+1:T}) p_\theta(y_{n+1:T}|y_{1:n}, x_{1:T}), \tag{14}$$

and

$$p_\theta(y_{n+1:T}|y_{1:n}, x_{1:T}) = \int p_\theta(y_{n+1:T}|z_n, x_{n+1:T}) p_\theta(z_n|x_{1:n}, y_{1:n}) dz_n. \tag{15}$$

In [33], it was shown that the coefficients of z_n in $p_\theta(y_{n+1:T}|z_n, x_{n+1:T})$ which are needed to evaluate (15) can be computed recursively for $n = T, T-1, ..., 1$ (the backward step). Then, for each $n = 1, 2, ..., T$,

$p_\theta(y_n|y_{1:n-1}, x_{1:n})$ and $p_\theta(z_n|x_{1:n}, y_{1:n})$ are obtained through standard Kalman filtering recursions, (15) is computed for each $x_n \in \mathcal{X}$ and a draw is made from (14) (the forward step). In the resulting algorithm, if the computational cost of evaluating $p_\theta(x_n|x_{1:n-1}, x_{n+1:T})$ is $\mathcal{O}(1)$, the cost of one sampling sweep through $p_\theta(x_1|y_{1:T}, x_{2:T})$, $p_\theta(x_2|y_{1:T}, x_1, x_{3:T})$, etc. grows linearly in T.

More recently, adaptive MCMC methods have been suggested to make one-at-a-time updates [34]. However, these algorithms are still susceptible to slow mixing if the components of $X_{1:T}$ are strongly correlated. Moreover even if we were able to sample efficiently using one-at-a-time updates, this algorithm might still converge slowly if $X_{1:T}$ and θ are strongly correlated; e.g. if $\{X_n\}$ is a Markov chain and θ includes the transition matrix of this chain.

4 Particle Markov chain Monte Carlo methods for switching state-space models

A natural idea arising from the previous section is to use the output $\widehat{p}_\theta(x_{1:T}|y_{1:T})$ of the DPF algorithm as part of a proposal distribution for a MCMC algorithm targeting $p_\theta(x_{1:T}|y_{1:T})$ or $p(\theta, x_{1:T}|y_{1:T})$. This could allow us, in principle, to design automatically an efficient high-dimensional proposal for MCMC. However a direct application of this idea would require us to be able to both sample from and evaluate pointwise the *unconditional* distribution of a particle sampled from $\widehat{p}_\theta(x_{1:T}|y_{1:T})$. This distribution is given by

$$q_\theta(x_{1:T}|y_{1:T}) = \mathbb{E}\left[\widehat{p}_\theta(x_{1:T}|y_{1:T})\right],$$

where the expectation is with respect to the probability law of the DPF algorithm: the stochasticity which produces the random probability measure $\widehat{p}_\theta(x_{1:T}|y_{1:T})$ in Eq. (8). While sampling from $q_\theta(x_{1:T}|y_{1:T})$ is straightforward as it only requires running the DPF algorithm to obtain $\widehat{p}_\theta(x_{1:T}|y_{1:T})$ then sampling from this random measure, the analytical expression of this distribution is clearly not available.

The novel MCMC updates presented in this section, under the umbrella term PMCMC, circumvent this problem by considering target distributions on an extended space, over all the random variables of the DPF algorithm. Details of their theoretical validity are given in Subsection 4.3 but are not required for implementation of the algorithms. The key feature of these PMCMC algorithms is that they are "exact approximations" to standard MCMC updates targeting $p(\theta, x_{1:T}|y_{1:T})$. More precisely, on the one hand these algorithms can be thought of as approximations to possibly "idealized" standard MH updates parametrized by the number N of particles used to construct the DPF approximation. On the other hand, under mild assumptions, PMCMC algorithms are guaranteed to generate asymptotically (in the number of MCMC iterations used) samples from $p(\theta, x_{1:T}|y_{1:T})$, *for any fixed number $N \geq 2$ of particles*, in other words, for virtually any degree of approximation.

In Subsection 4.1, we describe the *Particle MMH* (Marginal Metropolis-Hastings) algorithm which can be thought of as an exact approximation of an idealised "Marginal MH" (MMH) targetting directly the marginal distribution $p(\theta|y_{1:T})$ of $p(\theta, x_{1:T}|y_{1:T})$. This algorithm admits a form similar to the PMMH discussed in [1] but its validity relies on different arguments. In Subsection 4.2 we present a particle approximation of a Gibbs sampler targeting $p(\theta, x_{1:T}|y_{1:T})$, called the *Particle Gibbs* (PG) algorithm. It is a particle approximation of the "ideal" block Gibbs sampler which samples from $p(\theta, x_{1:T}|y_{1:T})$ by sampling iteratively from the full conditionals $p_\theta(x_{1:T}|y_{1:T})$ and $p(\theta|y_{1:T}, x_{1:T})$. This algorithm is significantly different from the PG sampler presented in [1] and incorporates a novel backward sampling mechanism. Convergence results for these algorithms are established in Subsection 4.3.

4.1 Particle marginal Metropolis-Hastings sampler

Let us consider the following ideal "marginal" MH (MMH) algorithm to sample from $p(\theta, x_{1:T}|y_{1:T})$ where θ and $x_{1:T}$ are updated simultaneously using the proposal given by

$$q\left((\theta^*, x^*_{1:T})|\,(\theta, x_{1:T})\right) = q\left(\theta^*|\,\theta\right) p_{\theta^*}\left(x^*_{1:T}|y_{1:T}\right) \ .$$

In this scenario the proposed $x^*_{1:T}$ is perfectly "adapted" to the proposed θ^* and the resulting MH acceptance ratio is given by

$$\frac{p(\theta^*, x^*_{1:T}|y_{1:T})}{p(\theta, x_{1:T}|y_{1:T})} \frac{q\left((\theta, x_{1:T})|\,(\theta^*, x^*_{1:T})\right)}{q\left((\theta^*, x^*_{1:T})|\,(\theta, x_{1:T})\right)} = \frac{p_{\theta^*}(y_{1:T})}{p_\theta(y_{1:T})} \frac{p(\theta^*)}{p(\theta)} \frac{q(\theta|\theta^*)}{q(\theta^*|\theta)} \ . \tag{16}$$

This algorithm is equivalent to a MH update working directly on the marginal density $p(\theta|y_{1:T})$, justifying the MMH terminology. This algorithm is appealing but typically cannot be implemented as the marginal likelihood terms $p_\theta(y_{1:T})$ and $p_{\theta^*}(y_{1:T})$ are cannot be computed exactly and it is impossible to sample exactly from $p_{\theta^*}(x_{1:T}|y_{1:T})$. We propose the following particle approximation of the MMH algorithm where, whenever a sample from $p_\theta(x_{1:T}|y_{1:T})$ and the expression for the marginal likelihood $p_\theta(y_{1:T})$ are needed, their DPF approximation counterparts are used instead.

PMMH sampler for SSSM

Initialisation, $i = 0$
- Set $\theta(0)$ arbitrarily.
- Run the DPF targeting $p_{\theta(0)}(x_{1:T}|y_{1:T})$, sample $X_{1:T}(0) \sim \widehat{p}_{\theta(0)}(\cdot|y_{1:T})$ and denote $\widehat{p}_{\theta(0)}(y_{1:T})$ the marginal likelihood estimate.

For iteration $i \geq 1$
- Sample $\theta^* \sim q(\cdot|\theta(i-1))$.
- Run the DPF targeting $p_{\theta^*}(x_{1:T}|y_{1:T})$, sample $X^*_{1:T} \sim \widehat{p}_{\theta^*}(\cdot|y_{1:T})$ and denote $\widehat{p}_{\theta^*}(y_{1:T})$ the marginal likelihood estimate.
- With probability

$$1 \wedge \frac{\widehat{p}_{\theta^*}(y_{1:T})\ p(\theta^*)}{\widehat{p}_{\theta(i-1)}(y_{1:T})\ p(\theta(i-1))} \frac{q(\theta(i-1)|\theta^*)}{q(\theta^*|\theta(i-1))} \tag{17}$$

  set $\theta(i) = \theta^*$, $X_{1:T}(i) = X^*_{1:T}$, $\widehat{p}_{\theta(i)}(y_{1:T}) = \widehat{p}_{\theta^*}(y_{1:T})$,
  otherwise set $\theta(i) = \theta(i-1)$, $X_{1:T}(i) = X_{1:T}(i-1)$, $\widehat{p}_{\theta(i)}(y_{1:T}) = \widehat{p}_{\theta(i-1)}(y_{1:T})$.

4.2 Particle Gibbs sampler

As discussed in Section 3.4, an attractive but impractical strategy to sample from $p(\theta, x_{1:T}|y_{1:T})$ consists of using the Gibbs sampler which iterates sampling steps from $p_\theta(x_{1:T}|y_{1:T})$ and $p(\theta|y_{1:T}, x_{1:T})$ or a modified Gibbs sampler where we insert a sampling step from $p_\theta(z_{0:T}|y_{1:T}, x_{1:T})$ after having sampled from $p_\theta(x_{1:T}|y_{1:T})$ to update θ according to $p(\theta|y_{1:T}, x_{1:T}, z_{0:T})$. Numerous implementations rely on the fact that sampling from the conditional density $p(\theta|y_{1:T}, x_{1:T})$ or $p(\theta|y_{1:T}, x_{1:T}, z_{0:T})$ is feasible and thus the potentially difficult design of a proposal density for θ can be bypassed. However, as mentioned before, it is typically impossible to sample from $p_\theta(x_{1:T}|y_{1:T})$. Clearly substituting to the sampling step from $p_\theta(x_{1:T}|y_{1:T})$, sampling from the DPF approximation $\widehat{p}_\theta(x_{1:T}|y_{1:T})$ would not provide Gibbs samplers admitting the correct invariant distribution.

We now present a valid particle approximation of the Gibbs sampler which assumes we can sample from $p(\theta|y_{1:T}, x_{1:T})$. Similarly it is possible to build a valid particle approximation of the modified Gibbs sampler by the same arguments, but we omit the details here for brevity.

PG sampler for SSSM

Initialisation, $i = 0$
- Set $\theta(0), X_{1:T}(0)$ arbitrarily.

For iteration $i \geq 1$
- Sample $\theta(i) \sim p(\cdot|y_{1:T}, X_{1:T}(i-1))$.
- Run a conditional DPF algorithm targeting $p_{\theta(i)}(x_{1:T}|y_{1:T})$ conditional upon $X_{1:T}(i-1)$.
- Run a backward sampling algorithm to obtain $X_{1:T}(i)$.

The remarkable property enjoyed by the PG algorithm is that under weak assumptions it generates samples from $p(\theta, x_{1:T}|y_{1:T})$ in steady state *for any number* $N \geq 2$ of particles used to build the required DPF approximations. The non-standard steps of the PG sampler are the conditional DPF algorithm and backward sampling algorithms which we now describe.

Given a value of θ and a trajectory $x^*_{1:T}$, the conditional DPF algorithm proceeds as follows. Kalman filtering recursions are given in Appendix A.

Conditional DPF algorithm

At time $n=1$

• Set $\mathbf{S}_1 = \mathcal{X}$ and for each $x_1 \in \mathcal{X}$ (which includes x_1^*), compute $m_{1|1}^{z,\theta}(x_1)$, $\Sigma_{1|1}^{z,\theta}(x_1)$ and $g_\theta(y_1|x_1)$ using the Kalman filter

• Compute and normalise the weights. For each $x_1 \in \mathcal{X}$,

$$\overline{w}_1^\theta(x_1) = \nu_\theta(x_1)\, g_\theta(y_1|\, x_1)\,,\ W_1^\theta(x_1) = \frac{\overline{w}_1^\theta(x_1)}{\sum_{x_1' \in \mathcal{X}} \overline{w}_1^\theta(x_1')}. \tag{18}$$

At times $n = 2, ..., T$

• If $|\mathbf{S}_{n-1}| \leq N$ set $C_{n-1} = \infty$ otherwise set C_{n-1} to the the unique solution of

$$\sum_{x_{1:n-1} \in \mathbf{S}_{n-1}} 1 \wedge C_{n-1} W_{n-1}^\theta(x_{1:n-1}) = N.$$

• If $W_{n-1}^\theta\left(x_{1:n-1}^*\right) > 1/C_{n-1}$, maintain the L_{n-1} trajectories which have weights strictly superior to $1/C_{n-1}$ (which includes $x_{1:n-1}^*$), then apply the stratified resampling mechanism to the other weighted trajectories to yield $N - L_{n-1}$ survivors. Set $\mathbf{S}_{n-1}'$ to the set of surviving and maintained trajectories.

• If $W_{n-1}^\theta\left(x_{1:n-1}^*\right) \leq 1/C_{n-1}$ maintain the L_{n-1} trajectories which have weights strictly superior to $1/C_{n-1}$ (which excludes $x_{1:n-1}^*$), then then apply the conditional stratified resampling mechanism to the other weighted trajectories to yield $N - L_{n-1}$ survivors. Set $\mathbf{S}_{n-1}'$ to the set of surviving and maintained trajectories.

• Set $\mathbf{S}_n = \mathbf{S}_{n-1}' \times \mathcal{X}$.

• For each $x_{1:n} \in \mathbf{S}_n$, update $m_{n|n}^{z,\theta}(x_{1:n})$ and $\Sigma_{n|n}^{z,\theta}(x_{1:n})$ and compute $g_\theta(y_n|y_{1:n-1}, x_{1:n})$ using the Kalman filter.

• Compute and normalise the weights. For each $x_{1:n} \in \mathcal{X}^n$,

$$\overline{w}_n^\theta(x_{1:n}) = f_\theta(x_n|x_{1:n-1}) g_\theta(y_n|y_{1:n-1}, x_{1:n}) \frac{W_{n-1}^\theta(x_{1:n-1})}{1 \wedge C_{n-1} W_{n-1}^\theta(x_{1:n-1})} \tag{19}$$

$$W_n^\theta(x_{1:n}) = \frac{\overline{w}_n^\theta(x_{1:n})}{\sum_{x_{1:n}' \in \mathbf{S}_n} \overline{w}_n^\theta(x_{1:n}')}. \tag{20}$$

• If backward sampling is to be used, store $W_n^\theta(x_{1:n})$, $m_{n|n}^{z,\theta}(x_{1:n})$ and $\Sigma_{n|n}^{z,\theta}(x_{1:n})$ for each $x_{1:n} \in \mathbf{S}_n$.

The conditional stratified resampling procedure can be implemented as follows.

Conditional stratified resampling

• Normalise the weights $\overline{w}_{n-1}^\theta(x_{1:n-1})$ of the $N|\mathcal{X}| - L_{n-1}$ particles and label them according to the order of the corresponding $x_{1:n-1}$ to obtain $\widehat{W}_{n-1}^\theta\left(x_{1:n-1}^{(i)}\right)$; $i = 1, ..., N|\mathcal{X}| - L_{n-1}$. Define κ to be the integer satisfying $x_{1:n-1}^{(\kappa)} = x_{1:n-1}^*$.

• Construct the corresponding cumulative distribution function: for $i = 1, ..., N|\mathcal{X}| - L_{n-1}$,

$$Q_{n-1}^\theta(i) := \sum_{j \leq i} \widehat{W}_{n-1}^\theta\left(x_{1:n-1}^{(j)}\right), \qquad Q_{n-1}^\theta(0) := 0.$$

• Sample U_* uniformly on $\left[Q_{n-1}^\theta(\kappa - 1), Q_{n-1}^\theta(\kappa)\right]$, set $U_1 = U_* - \frac{\lfloor (N - L_{n-1}) U_* \rfloor}{N - L_{n-1}}$ and compute $U_j = U_1 + \frac{j-1}{N - L_{n-1}}$ for $j = 2, ..., N - L_{n-1}$. Here $\lfloor a \rfloor$ denotes the largest integer not greater than a.

• For $i = 1, ..., N|\mathcal{X}| - L_{n-1}$, if there exists $j \in \{1, ..., N - L_{n-1}\}$ such that $Q_{n-1}^\theta(i-1) < U_j \leq Q_{n-1}^\theta(i)$, then $x_{1:n-1}^{(i)}$ survives.

The backward sampling step is an important component of the PG algorithm. In contrast to the standard PMCMC algorithms of [1], it allows the sampled trajectory obtained from the conditional SMC update not

only to be chosen from those surviving at time T, but allows full exploration of all trajectories sampled during the Conditional DPF algorithm. Further comments on the theoretical validity of alternative schemes are made in section 4.3 and demonstration of numerical performance given in section 5.

We note that this procedure is of some independent interest for smoothing in SSSM's if θ is known, as it can be combined with the standard DPF algorithm. A forward filtering-backward smoothing algorithm for SSSM was devised in [27], and involved joint sampling of both continuous and discrete variables from an approximation of $p_\theta(x_{1:T}, z_{0:T}|y_{1:T})$. The backward sampling algorithm we propose is different because the continuous component of the state is integrated out analytically, giving a further Rao-Blackwellization over the scheme of [27]. Furthermore, the fact that the backward sampling algorithm involves sampling only discrete–valued variables is central to the validity of the PG algorithm, discussed in the next section. Details of the matrix-vector recursions necessary for the implementation of the backward sampling procedure are given in Appendix B.

Backward Sampling

At time $n = T$

• Sample a path $X'_{1:T}$ from the distribution on $\mathbf{S}_T \subset \mathcal{X}^T$ defined by $\{W_T^\theta(x_{1:T})\}$, then discard $X'_{1:T-1}$ to yield X'_T. Set $\Xi_T = 0$, $\mu_T = 0$.

At times $n = T-1, ..., 1$

• Update Ξ_n and μ_n as per the procedure of Appendix B.

• For each $x_{1:n} \in \mathbf{S}_n$ compute the backward weight

$$V_n^\theta\left(x_{1:n}\left|x'_{n+1:T}\right.\right) \propto W_n^\theta(x_{1:n}) p_\theta(x'_{n+1:T}|x_{1:n}) p_\theta(y_{n+1:T}|y_{1:n}, x_{1:n}, x'_{n+1:T})$$

where

$$p_\theta(y_{n+1:T}|y_{1:n}, x_{1:n}, x'_{n+1:T}) = \int p_\theta(y_{n+1:T}|z_n, x'_{n+1:T}) p_\theta(z_n|x_{1:n}, y_{1:n}) dz_n$$

is evaluated using μ_n, Ξ_n and the stored $m_{n|n}^{z,\theta}(x_{1:n})$ and $\Sigma_{n|n}^{z,\theta}(x_{1:n})$ of $p_\theta(z_n|x_{1:n}, y_{1:n})$ as per Eq. (28) in Appendix B.

• Normalise the backward weights $\left\{V_n^\theta\left(x_{1:n}\left|x'_{n+1:T}\right.\right)\right\}_{x_{1:n}\in\mathbf{S}_n}$ and draw from the distribution they define on $\mathbf{S}_n \subset \mathcal{X}^n$ to obtain $X'_{1:n}$.

• If $n > 1$ discard $X'_{1:n-1}$, otherwise output $X'_{1:T}$.

4.3 Validity of the algorithms

The key to establishing the validity of the PMCMC algorithms is in showing that these are standard MCMC algorithms on an extended state-space including all the random variables introduced in the DPF algorithm.

The first step is observe that under our representation of the DPF algorithm, its operation remains essentially unchanged if at each iteration we adopt the convention of setting $\overline{w}_n^\theta(x_{1:n}) = W_n^\theta(x_{1:n}) = 0$ for all $x_{1:n} \notin \mathbf{S}_n$, and to replace all summations over $\mathbf{S}_n$ with summations over $\mathcal{X}^n$. We assume this convention throughout the remainder of this section. In this case the solution of $\sum_{x_{1:n}\in\mathbf{S}_n} 1 \wedge C_n W_n^\theta(x_{1:n}) = N$ is identical to the solution of $\sum_{x_{1:n}\in\mathcal{X}^n} 1 \wedge C_n W_n^\theta(x_{1:n}) = N$. Furthermore we can consider the resampling mechanism as acting on all trajectories in $\mathcal{X}^n$ and not only those in $\mathbf{S}_n$; those with zero weights clearly fall below the threshold $1/C_n$ and there is zero probability of them surviving the resampling operation. As we shall see, the intuitive implication of this observation is that once a trajectory $x_{1:n}$ has been discarded, it is lost and for any $m > n$, any subsequent trajectory $(x_{1:n}, x'_{n+1:m}) \in \mathcal{X}^m$ is also assigned zero weight. We denote by $\mathbf{W}_n^\theta$ the set of normalised importance weights at time n.

We next write an expression for the joint distribution of the sequence of random support sets $\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_T$ generated through the DPF algorithm. By definition of the algorithm, for $n \geq 2$, $\mathbf{S}_n$ is conditionally independent of the history of the algorithm given $\mathbf{W}_{n-1}^\theta$:

$$\mathbf{S}_n|\left(\mathbf{W}_{n-1}^\theta = \mathbf{w}_{n-1}^\theta\right) \sim r_n^N(\cdot|\mathbf{w}_{n-1}^\theta), \tag{21}$$

where for each N, n and $\mathbf{w}_{n-1}^\theta$, $r_n^N(\cdot|\mathbf{w}_{n-1}^\theta)$ can be understood as a probability distribution over the set of subsets of $\mathcal{X}^n$, and we denote this set of subsets by $\mathcal{P}(\mathcal{X}^n)$. This density is parameterized by N because for all $n \geq 2$, for each point $\mathbf{s}_n$ in the support of $r_n^N(\cdot|\mathbf{w}_{n-1}^\theta)$, $|\mathbf{s}_n| = N|\mathcal{X}|$. In the case of $n = 1$, $r_n^N(\cdot) = \mathbb{I}[\cdot = \mathcal{X}]$.

We will not need an explicit expression for the distribution (21), but from the definition of the optimal resampling mechanism [22, 23], we know that it has the following marginal property: for all $x_{1:n} \in \mathcal{X}^n$, we have

$$r_n^N(x_{1:n} \in \mathbf{s}_n | \mathbf{w}_{n-1}^\theta) = 1 \wedge c_{n-1} w_{n-1}^\theta (x_{1:n-1}) . \tag{22}$$

where we have adopted the abusive notation that

$$r_n^N(x_{1:n} \in \mathbf{s}_n | \mathbf{w}_{n-1}^\theta) := \sum_{\mathbf{s}'_n : x_{1:n} \in \mathbf{s}'_n} r_n^N(\mathbf{s}'_n | \mathbf{w}_{n-1}^\theta)$$

Eq. (22) implies

$$r_n^N(x_{1:n} \in \mathbf{s}_n | w_{n-1}^\theta (x_{1:n-1}) = 0) = 0.$$

Combined with Eq. (21) we see that for any n and $x_{1:n-1}$, conditional on the event that $W_{n-1}^\theta (x_{1:n-1}) = 0$, any subsequent paths which have $x_{1:n-1}$ as their first $n-1$ coordinates are also assigned zero weight and are not members of any subsequent $\mathbf{S}_n$. Thus the corresponding subsequent weights need never be computed or stored, as required to control the cost of the algorithm. We thus have the property as claimed earlier that once a trajectory is discarded it is not recovered. To summarize the law of the DPF algorithm, we can write the density of $\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_T$ on $\prod_{n=1}^T \mathcal{P}(\mathcal{X}^n)$ as

$$\psi_\theta^N(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T) = r_1^N(\mathbf{s}_1) \prod_{n=2}^T r_n^N(\mathbf{s}_n | \mathbf{w}_{n-1}^\theta). \tag{23}$$

As the weights $\mathbf{W}_n^\theta$ are just a deterministic function of $\mathbf{S}_1, \ldots, \mathbf{S}_n$, it is not necessary to introduce them as arguments of ψ_θ^N.

The key to the PMCMC algorithms described here is to define the following artificial target density on $\Theta \times \mathcal{X}^T \times \prod_{n=1}^{T-1} \mathcal{P}(\mathcal{X}^n)$ through

$$\pi^N(\theta, x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T) = p(\theta, x_{1:T} | \, y_{1:T}) \left\{ \prod_{n=2}^T \mathbb{I}[x_{1:n} \in \mathbf{s}_n] \right\} \frac{\psi_\theta^N(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)}{\prod_{n=2}^T r_n^N(x_{1:n} \in \mathbf{s}_n | \mathbf{w}_{n-1}^\theta)} \tag{24}$$

which admits $p(\theta, x_{1:T} | \, y_{1:T})$ as a marginal by construction. Let $\pi_\theta^N(x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$ denote the density of $X_{1:T}, \mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_T$ conditional upon θ under $\pi^N(\theta, x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$. In the following results we show that the PMMH and PG algorithms are just standard MCMC updates targeting this artificial distribution. Proofs can be found in Appendix C.

We first present a result establishing the convergence of the PMMH sampler which relies on the following assumption.

(**A1**) The MH sampler of target density $p(\theta | \, y_{1:T})$ and proposal density $q(\theta^* | \theta)$ is irreducible and aperiodic (and hence converges for almost all starting points).

We have the following result.

Theorem 1 *For any $N \geq 2$*

1. *the PMMH sampler is an MH sampler defined on the extended space $\Theta \times \mathcal{X}^T \times \prod_{n=1}^T \mathcal{P}(\mathcal{X}^n)$ with target density $\pi^N(\theta, x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$ defined in Eq. (24) and proposal density*

$$q(\theta^* | \theta) \; w_T^{\theta^*} (x_{1:T}^*) \; \psi_{\theta^*}^N (\mathbf{s}_1^*, \mathbf{s}_2^*, ..., \mathbf{s}_T^*) \tag{25}$$

   where $w_T^{\theta^*} (x_{1:T}^*)$ is the realisation of the normalised importance weight associated to the population of particles proposed by the DPF algorithm.

2. *if additionally (A1) holds, the PMMH sampler generates a sequence $\{\theta(i), X_{1:T}(i)\}$ whose marginal distributions $\{\mathcal{L}^N((\theta(i), X_{1:T}(i)) \in \cdot)\}$ satisfy*

$$\left\| \mathcal{L}^N((\theta(i), X_{1:T}(i)) \in \cdot) - p(\cdot, \cdot | \, y_{1:T}) \right\|_{TV} \to 0 \text{ as } i \to \infty \; .$$

   for almost all starting points.

Next we consider the backward sampling procedure and establish its invariance properties.

Proposition 1 *For any $N \geq 2$ and $\theta \in \Theta$, assume $(X_{1:T}, \mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_T)$ is distributed according to $\pi_\theta^N(\cdot)$ and let $X'_{1:T}$ be the trajectory obtained at* any *time step m of the backward sampling procedure operating on $(X_{1:T}, \mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_T)$. Then $X'_{1:T}$ is distributed according to $p_\theta(x_{1:T}|y_{1:T})$.*

We now state a sufficient condition for the convergence of the PG sampler and provide a simple convergence result.

(**A2**) The Gibbs sampler defined by drawing alternately from the conditionals $p(\theta|y_{1:T}, x_{1:T})$ and $p_\theta(x_{1:T}|y_{1:T})$ is irreducible and aperiodic (and hence converges for p-almost all starting points).

We have the following result.

Theorem 2

1. *steps $1-4$ of the PG update define a transition kernel on the extended space $\Theta \times \mathcal{X}^T \times \prod_{n=1}^T \mathcal{P}(\mathcal{X}^n)$ of invariant density $\pi^N(\theta, x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$ defined in Eq. (24) for any $N \geq 2$.*
2. *if additionally (A2) holds, the PG sampler generates a sequence $\{\theta(i), X_{1:T}(i)\}$ whose marginal distributions $\{\mathcal{L}_{PG}^N((\theta(i), X_{1:T}(i)) \in \cdot)\}$ satisfy for any $N \geq 2$*

$$\left\|\mathcal{L}_{PG}^N((\theta(i), X_{1:T}(i)) \in \cdot) - p(\cdot, \cdot|\, y_{1:T})\right\|_{tv} \to 0 \text{ as } i \to \infty,$$

for almost all starting points.

Remark 1 *The reader will observe that as Proposition 1 applies for any time step of the backward sampling, modification of the PG algorithm to the case where $X_{1:T}(i)$ is set to the $X'_{1:T}$ obtained at any time step of the backward sampling procedure also corresponds to a Markov kernel of the required invariant distribution. For example, one could simply apply only the first backward sampling step: sample $X'_{1:T}$ from the distribution defined by $\{W_T(x_{1:T})\}$ and then set $X_{1:T}(i) = X'_{1:T}$. The resulting algorithm is closer akin to the original Particle Gibbs algorithm of [1]. However, in numerical experiments in the context of SSSMs this approach has been found to be relatively inefficient. This phenomenon is discussed further and demonstrated numerically in section 5.*

5 Applications

5.1 Example 1: Autoregression with shifting level

In our first numerical experiments we return to the toy model specified in section 2.1.1 and address some generic issues regarding algorithmic settings and performance.

5.1.1 Particle Gibbs and the effect of backward sampling

We first demonstrate the effect of applying the backward sampling procedure as part of the PG algorithm. The purpose of this section is to show the importance of applying backward sampling as part of the PG algorithm and to show its advantage over the standard Gibbs sampler. From hereon we refer to as "PG without backward sampling" the alternative PG scheme described in Remark 1 which involves sampling $X'_{1:T}$ from the distribution defined by $\{W_T(x_{1:T})\}$ and immediately setting $X_{1:T}(i) = X'_{1:T}$.

Recall that for this model the parameters are $\theta = [\phi\ \sigma^2\ P_X]$. Conjugate priors are readily available: a Gaussian distribution for ϕ, inverse-gamma for σ^2 and independent Dirichlet for each row of P_X. A data record of length $T = 1000$ was generated from the model with true parameter values of $\phi = 0.1$, $\sigma = 0.1$ and $P_X = \begin{bmatrix} 0.99 & 0.01 \\ 0.99 & 0.01 \end{bmatrix}$. Flat Dirichlet priors were set on each row of P_X. A $\mathcal{N}(0, 10)$ distribution restricted to $|\phi| \leq 1$ was set over ϕ and a $(0.1, 0.1)$ inverse gamma distribution was set over σ^2. The initial distribution over μ_0 was $\mathcal{N}(0, 10)$. For various numbers of particles the PG algorithm, with and without backward sampling, was run and compared to the standard one-at-a-time Gibbs algorithm in terms of the sample lag 1 autocorrelation for each component of the discrete latent trajectory $\{X_n\}$. In all cases the simulation smoother of [20] was used to sample from $p_\theta(z_{0:T}|y_{1:T}, x_{1:T})$.

In both panes of Figure 1. the vertical dashed lines show the true times at which $X_n = 1$. The bottom pane shows the lag 1 autocorrelation for PG with backward sampling and the standard one-at-a-time Gibbs sampler: here it was found that for all components of the trajectory, increasing N monotonically decreased the autocorrelation and for any N the PG algorithm exhibited lower autocorrelation than the standard one-at-a-time algorithm. Spikes in the autocorrelation coincide with the true times at which $X_n = 1$ and between these times the autocorrelation, even using the standard Gibbs sampler, was found to be very low. By contrast, for the PG without backward sampling and the same numbers of particles, the autocorrelation from the PG algorithm was higher than that from the standard Gibbs algorithm for most components of discrete trajectory. In all cases the sample autocorrelation was computed from 10^5 iterations after a burn-in of 10^4 iterations. After the 10^4+10^5 iterations, with $N = 10$ and $N = 20$ particles, the PG without backward sampling had entirely failed to converge: in the plots of Figure 1, we use the ranges in which the plots reach the value exactly 1 to represent those components of the discrete trajectory never having changed from their initial condition (such a sample sequence does not have a well defined autocorrelation as its sample variance is zero). Very similar results were observed for other initialisations and data records.

This performance can be explained in terms of the well-known particle path degeneracy phenomenon which arises from the resampling mechanism in SMC algorithms: the act of repeated selection of sampled paths inevitably leads to a loss in diversity in their early components. In the present context the path degeneracy influences the performance of the PG algorithms via the conditional DPF update. During the conditional DPF operation at MCMC iteration $i + 1$, by construction of the conditional DPF, $X_{1:T}(i)$ is forced to survive until time step T. Thus for the PG without backward sampling, for some $m < T$, the path degeneracy phenomenon implies there is a significant probability that $X_{1:m}(i)$ coincides with $X_{1:m}(i + 1)$. This explains the strong correlations between components of consecutive samples of the latent trajectory shown in the top pane of Figure 1. By contrast, backward sampling provides a chance for the path degeneracy to be circumvented. The CPU time for one iteration of the PG with backward sampling was found to be between 1 and 1.5 times that without backward sampling for the same number of particles. The results therefore indicate that overall it is significantly more efficient to use the backward sampling method and from now on it is the only PG algorithm we consider.

Figure 2 shows sample autocorrelation as a function of lag for various numbers of particles from the PG algorithm with backward sampling and the standard one-at-a-time Gibbs sampler. We observe that using large N leads to lower autocorrelation and very little decrease in autocorrelation was observed using more than $N = 50$ particles. As we go on to discuss in more details in the next section, under the Dirichlet prior for each row of P_X it is possible to analytically integrate out P_X both when using the standard Gibbs sampler and the PG, and we did so. The above experiments were also conducted in the case where P_X is not integrated out and we obtained results which were almost identical (not shown).

5.1.2 Treatment of P_X

A common feature of SSSMs is that it is possible to analytically integrate out P_X under Dirichlet priors for each of its rows and the autoregressive model with shifting level is no exception. It is natural to ask, even in the context of standard MCMC algorithms, whether it is beneficial to perform this integration analytically, or to treat P_X as part of the sampling problem. To the authors' knowledge, in the context of SSSMs this issue has not been treated in the literature.

Consider first the standard one-at-a-time Gibbs sampling case. The reader will recall from section 3.4 and [33] that the algorithm involves sampling from

$$p_\theta\left(x_n|y_{1:T}, x_{1:n-1}, x_{n+1:T}\right) \propto p_\theta(y_n|y_{1:n-1}, x_{1:n})p_\theta(x_n|x_{1:n-1}, x_{n+1:T})p_\theta(y_{n+1:T}|y_{1:n}, x_{1:T}) \tag{26}$$

for each n. Conditionally on P_X, the process $\{X_n\}_{n\geq 1}$ is Markov and so in the above display we have the simplification $p_\theta(x_n|x_{1:n-1}, x_{n+1:T}) = p_\theta(x_n|x_{n-1}, x_{n+1})$. Conversley, when P_X is integrated out, in which case the parameter reduces to $\theta = [\phi\ \sigma^2]$, the process $\{X_n\}_{n\geq 1}$ is not Markov and the former simplification is not applicable. Thus, in terms of the correlation structure of the Markov chains generated by the corresponding Gibbs samplers, there appears to be a trade-off between conditioning on P_X and conditioning on components of the $\{X_n\}_{n\geq 1}$ process when drawing from distributions of the form (26). In terms of computational cost there is no significant difference: in the case that P_X is integrated out analytically evaluation of $p_\theta(x_n|x_{1:n-1}, x_{n+1:T})$ requires only state-transition count statistics which are cheap to compute and store.

Analogous remarks to those above hold for the PG algorithm. It involves computing $f_\theta(x_n|x_{1:n-1})$ in the conditional DPF step and $p_\theta(x_{n+1:T}|x_{1:n})$ in the backward sampling step and it is in these places that the

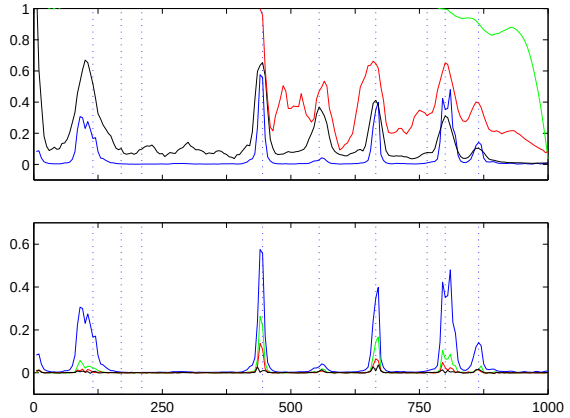

Figure 1: Example 1. Sample lag-1 autocorrelation for each of the discrete trajectory components $\{X_n(i),\ n = 1, ..., 1000\}$ with (bottom) and without (top) backward sampling for various numbers of particles: green $N = 10$, red $N = 20$, black $N = 50$. In both top and bottom the blue line is sample autocorrelation for standard one-at-a-time Gibbs. Vertical dashed lines are true locations of $X_n = 1$.

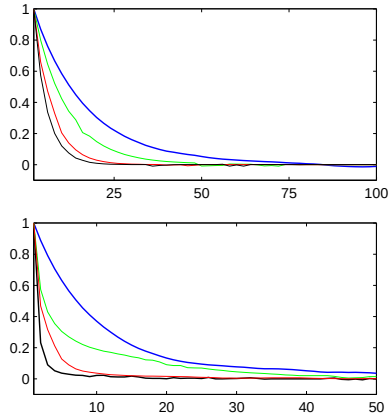

Figure 2: Example 1. Autocorrelation against lag for standard Gibbs sampler (blue) and PG with backward sampling and various numbers of particles: green $N = 10$, red $N = 20$, black $N = 50$. Top pane is for ϕ and bottom pane for σ^2.

same conditioning issues arise. In our numerical experiments for this model and others we were unable to establish that either incorporating P_X into the sampling problem or integrating it out analytically lead to a significant advantage in terms of sample autocorrelation, both for the standard one-at-a-time Gibbs sampler and the PG algorithm (results not shown). It would be very interesting to study the theoretical properties underlying this issue in Gibbs sampling algorithms for SSSMs but such an investigation is well beyond the scope of this document.

We found more obvious effects in the context of the PMMH algorithm, which we now go on to discuss. In this case $f_\theta(x_n|x_{1:n-1})$ is computed as part of the DPF algorithm, which is where the same conditioning issues arise. A data record of length $T = 1000$ was generated from the model with the same true parameter values as stated in the previous section. The same prior distributions were also employed. Central to the performance of the PMMH algorithm is the normalizing constant estimate $\widehat{p}_\theta(y_{1:T})$ computed using the DPF. When the variance of this estimate is large the PMMH algorithm performs poorly, exhibiting a high rejection rate - a characteristic shared with the standard PMCMC algorithms in [1]. We found that in the two cases (where P_X was integrated out and where it was not), the DPF exhibited striking differences in the variance of this estimate. The parameter θ was set to its true value and the DPF was run 1000 times on the simulated data set. Figure 3 shows the sample variance of $\log \widehat{p}_\theta(y_{1:n})$ as a function of n. The bottom pane corresponds to the case in which P_X is integrated out analytically. In this case the sample variance grows super-linearly with n. By contrast, as shown in the top pane, when conditioning on P_X the variance grows far more slowly. Very similar results were obtained when conditioning on values of P_X other than the truth. A step towards explaining this phenomenon is noting that integrating out P_X destroys the ergodicity properties of the latent process $\{X_n\}$conditional on θ. For standard SMC algorithms it is now theoretically well understood that assumptions about the ergodicity properties of the latent process are central to establishing linear growth rates (with respect to n) for the error in normalizing constant-type estimates [15, 16]. Our numerical results are consistent with the DPF having similar properties.

The variance of $\widehat{p}_\theta(y_{1:T})$ influences the acceptance rates of the corresponding two PMMH algorithms. Of course the trade-off is that when implementing a PMMH algorithm which incorporates P_X into the sampling problem one has the added burden of designing proposal moves for P_X and the contribution to the variability of the MH acceptance ratio from these proposals also influences the acceptance rate. In our experiments we found that an effective approach to making proposals for P_X was to reparameterize the model in terms of the unnormalized components of each row of P_X, with the Dirichlet prior corresponding to gamma priors over these components. Proposals could then be made using log-Gaussian random walks (an analogous approach was advocated in the [36] in the context of static mixture models). In numerical experiments we

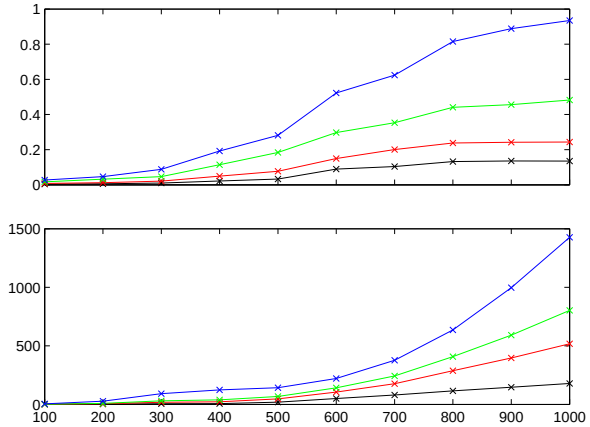

Figure 3: Example 1. Sample variance of $\log \widehat{p}_\theta(y_{1:n})$ for fixed θ as a function of n for various numbers of particles: blue: $N = 10$, green: $N = 50$, red: $N = 100$, black: $N = 200$. Top pane is conditional on the true value of P_X and bottom pane is with P_X integrated out.

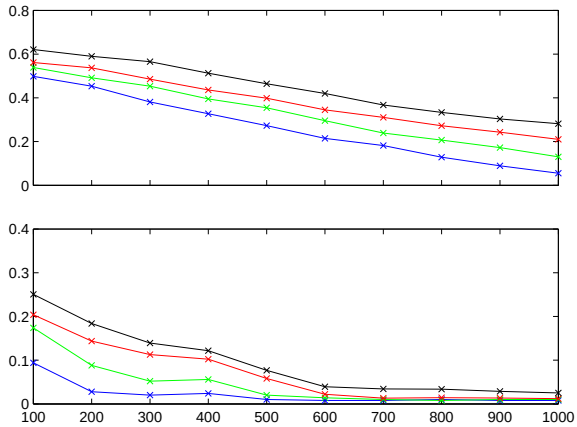

Figure 4: Example 1. PMMH acceptance rate as function of data record length for various numbers of particles: blue: $N = 10$, green: $N = 50$, red: $N = 100$, black: $N = 200$. Top pane is PMMH making proposals for P_X and bottom pane is with P_X integrated out.

adopted this approach with independent log-Gaussian random walk proposals made on each unnormalized component of P_X. After a couple of preliminary runs, the standard deviation of the increment in the log domain was set to 0.05. A log-Gaussian random walk proposal with the same standard deviation was also used for the parameter σ^2 and a Gaussian random walk with standard deviation 0.1 was used for ϕ. For the case where P_X is integrated out we used the same proposals as above for ϕ and σ^2. Figure 4 shows the PMMH acceptance rates as a function of the length of the data record. These results were obtained over 10^5 iterations of the algorithms after a burn-in of 10^4. The results show that the acceptance rate drops much more rapidly in the case that P_X is integrated out. However, we cannot conclude that the PMMH algorithm is always more efficient when P_X is incorporated into the sampling problem as the overall efficiency naturally depends on the particular choice of proposal mechanism for P_X. Our numerical results do indicate that even using a fairly simple proposal mechanism for P_X one can obtain acceptance rates which are superior to those in the case that P_X is integrated out analytically and the autocorrelation plots in Figure 5 show that this is carried over to lower sample autocorrelation for the parameters ϕ and σ^2.

5.1.3 PMMH and label switching in an unidentified model

The attractive property of Gibbs samplers in general is the relative ease of their implementation. Typically there is no algorithmic design involved and one only has to derive the full conditional distributions from which to sample. However, in some situations their performance can be poor. To represent the potential for performance benefits when using the PMMH algorithm we consider a slight generalization of the above auto-regressive model with level switching. Again we have $\mathcal{X} = \{0, 1\}$ and given P_X, the process $\{X_n\}_{n\geq 1}$ is Markov. However, we now have

$$Z_n = \begin{bmatrix} Y_n - \mu_n \\ \mu_n \end{bmatrix}, \quad A_\theta(x_n) = \begin{bmatrix} \phi & 0 \\ 0 & 1 \end{bmatrix}, \quad B_\theta(0) = \begin{bmatrix} \sigma_Y & 0 \\ 0 & \sigma_{\mu,0} \end{bmatrix}, \quad B_\theta(1) = \begin{bmatrix} \sigma_Y & 0 \\ 0 & \sigma_{\mu,1} \end{bmatrix},$$

$$C_\theta(x_n) = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad D_\theta(x_n) = F_\theta(x_n) = G_\theta(x_n) = 0, \quad \forall x_n.$$

The unknown parameters of this model are $\theta = [\phi \ \sigma_Y \ \sigma^2_{\mu,0} \ \sigma^2_{\mu,1} \ P_X]$. An important characteristic of this model is its invariance to labelling of the discrete latent states, a feature which is common in switching and mixture models in general. Under a symmetric prior, the posterior distribution over the parameters exhibits two symmetric modal patterns corresponding to the two possible labellings of the states. Permutation sampling methods for dynamic mixture models have been discussed in [31]. For purposes of exposition we use the multi-modality of the posterior as an informal test of the mixing properties of MCMC algorithms. Following the rationale of [36] in the context of static mixtures, as we know the posterior has several

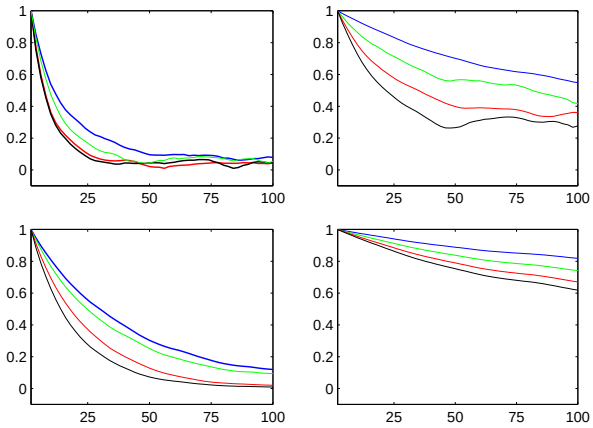

Figure 5: Example 1. Sample autocorrelation against lag for various numbers of particles, blue: $N = 10$, green: $N = 50$, red: $N = 100$, black: $N = 200$. Left column is PMMH with P_X sampled and right column is with P_X integrated out analytically. Top plots are for ϕ and bottom plots are for σ^2.

symmetric modes, we can be more confident in the mixing abilities of a sampler which visits all of these modes than one that does not. A data record of length $T = 1000$ was generated from the model with true parameter values $\phi = 0.1$, $\sigma_{\mu,0} = 0.1$, $\sigma_{\mu,1} = 0.5$, $\sigma_Y = 0.1$ and $P_X = \begin{bmatrix} 0.99 & 0.01 \\ 0.99 & 0.01 \end{bmatrix}$. Flat Dirichlet priors were set on each row of P_X. A $\mathcal{N}(0, 10)$ distribution restricted to $|\phi| \leq 1$ was set over ϕ and independent $(0.1, 0.1)$ inverse gamma distributions were set over $\sigma^2_{\mu,0}$, $\sigma^2_{\mu,1}$, and σ^2_Y. We ran the PMMH algorithm in which P_X is incorporated in to the sampling problem. The proposal for ϕ was Gaussian random walk of standard deviation 0.1. We again take the approach of reparameterizing in terms of the unnormalized components of P_X and for each of these components we used an independent, mixture of log-Gaussian random walks proposal with two components: the first with weight 0.8, and log-domain standard deviation 0.05 and the second with weight 0.1, standard deviation 2. The same proposals were also used for $\sigma^2_{\mu,0}$, $\sigma^2_{\mu,1}$, and σ^2_Y. We set $N = 100$ (for larger N no difference in performance was observed) and ran the PMMH algorithm and the PG sampler for 10^5 iterations after discarding the first 10^4 samples. Segments of trace plots for $\log \sigma^2_{\mu,0}$ and $\log \sigma^2_{\mu,1}$ are shown in Figure 6. We observe that the PG sampler is not able to switch between modes where as the PMMH sampler is. This demonstrates the better mixing performance which can be achieved using the PMMH algorithm. It may be possible to devise more sophisticated, correlated proposals for the unnormalised components of P_X but we do not pursue this here.

5.2 Example 2: Multiple change-point model with dependence between segments.

There is an extensive literature on statistical time series analysis based on multiple change-point models. In such models it is often assumed that given the position of a change-point, the data after that change-point are conditionally independent of those before, see for example [4, 25], amongst many others. This modelling assumption may be restrictive in some circumstances. A natural way to relax it is via a SSSM, which allows the notion of change-points to be introduced whilst allowing potentially complex dependence structures across segments of the data.

We consider a multiple-change point model in which observations arise from a latent process which is piece-wise linear. Changes in the latent process are of two varieties: those in which there is a discontinuity in the latent trajectory and its gradient and those in which there is a discontinuity only in the gradient. More specifically, we have $\mathcal{X} = \{0, 1, 2\}$ and we assume that $\{X_n\}$ is Markov with unknown transition matrix P_X. The observations $\{Y_n\}$ are valued in $\mathbb{R}$, as are the latent trajectory $\{\mu_n\}$ and its gradient $\{\dot{\mu}_n\}$. In state-space form we have

$$Z_n = \begin{bmatrix} \mu_n \\ \dot{\mu}_n \end{bmatrix}, \quad A_\theta(0) = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}, \quad A_\theta(1) = \begin{bmatrix} 1 & \Delta \\ 0 & 0 \end{bmatrix}, \quad A_\theta(2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

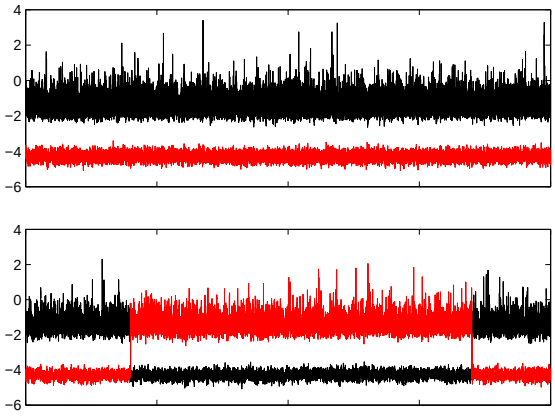

Figure 6: Example 1. Trace plots for PG (top) and PMMH (bottom) for $\log \sigma^2_{\mu,0}$ and $\log \sigma^2_{\mu,1}$ in the unidentified model. True values are $\log \sigma^2_{\mu,0} = -4.61$ and $\log \sigma^2_{\mu,0} = -1.39$.

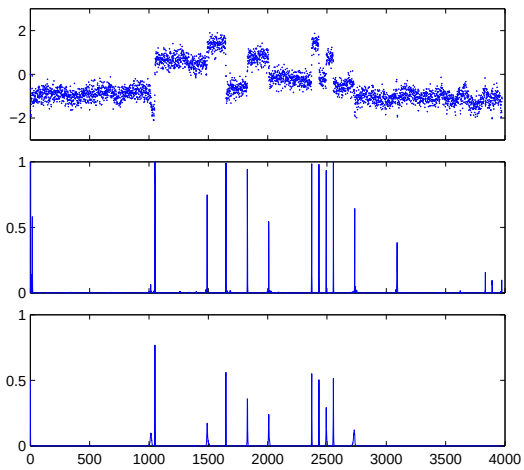

Figure 7: Example 2. Top: well-log data. Middle and bottom panes are estimated $p(X_n = 2|y_{1:T})$ from respectively the standard Gibbs and PG samplers.

$$B_\theta(0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B_\theta(1) = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_{\mu,1} \end{bmatrix}, \quad B_\theta(2) = \begin{bmatrix} \sigma_{\mu,0} & 0 \\ 0 & \sigma_{\mu,1} \end{bmatrix},$$

$$C_\theta(x_n) = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D_\theta(x_n) = \sigma_Y, \quad F_\theta(x_n) = G_\theta(x_n) = 0, \quad \forall x_n.$$

Here Δ is a fixed time incremement and the unknown parameters are $\theta = [\sigma^2_Y \ \sigma^2_{\mu,0} \ \sigma^2_{\mu,1} \ P_X]$. We apply this model to the analysis of well-log data: measurements of the nuclear resonance of underground rocks, as studied originally in [28]. Observations arise from a drill bit which passes down through layers of rock over time and each datum is a measurement of the resonance of the rock through which the bit is passing at that time. The aim is to identify segments in the data, each corresponding to a stratum of a single type of rock. The data set we analyse was treated in [23, 25, 26] under a variety of models, but in all these cases the static parameters of the models were assumed known. In [26] a change-point model with dependence across segments was employed and its advantages in terms of avoiding spurious detection of change-points was demonstrated. We are interested in similar analysis, but without assuming fixed values for the static parameters of the model. As in [23, 25, 26] a few extreme outliers were removed from the data set manually resulting in 3975 data points.

Flat Dirichlet priors were set on each row of P_X. Independent inverse gamma $(2, 3)$ priors were placed over σ^2_Y, $\sigma^2_{\mu,0}$ and $\sigma^2_{\mu,1}$. In our experiments, inference was found to be insensitive to choice of parameters for these inverse gamma priors (not shown). For the initial distribution over Z_0 we set a relatively diffuse, zero mean Gaussian prior with diagonal covariance components 100 and 100, corresponding to μ_0 and $\dot{\mu}_0$ respectively. We set $\Delta = 0.1$. The standard Gibbs sampler was run for 2×10^6 iterations and PG sampler with $N = 50$ for 4×10^4 iterations so as to equate computational cost. Histograms of sample output for σ^2_Y, $\sigma^2_{\mu,0}$ and $\sigma^2_{\mu,1}$ are shown in Figure 8. These results indicate that despite the long run the standard Gibbs sampler has not converged: most noticeably in the case of the histograms for $\sigma^2_{\mu,1}$, it appears not to have explored the support as thoroughly as the PG sampler and has become stuck in a mode of the distribution. The difference in performance is even more striking when considering the corresponding estimated posterior probabilities for the latent switching process. Figure 7 shows the estimated marginal posterior probabilities of each X_n being in state 2 (recall this state corresponds to a discontinuity in the latent process $\{\mu_n\}$ and its gradient) for each time step of the data record. Due to the lack of full exploration of the parameter space, the results for the standard Gibbs sampler show erroneously high posterior probabilities that each X_n is in state 2. We can conclude that for the same computational cost the performance of the PG sampler is superior.

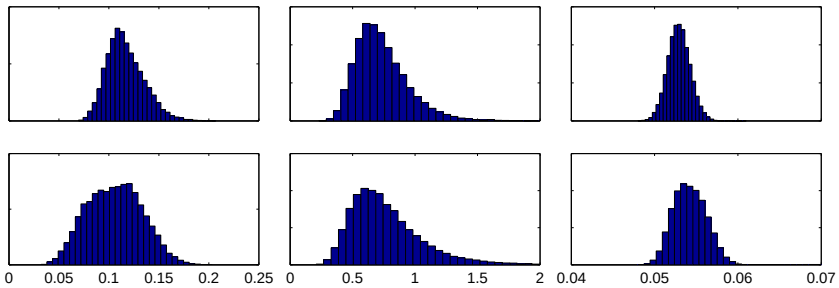

Figure 8: Example 2. Histograms estimates of posterior marginals. Top row: standard Gibbs sampler. Bottom row: PG sampler. Columns from left to right are $\sigma^2_{\mu,1}$, $\sigma^2_{\mu,0}$ and σ^2_Y.

5.3 Example 3: Exchange Rate Model

The following model was investigated in [21, 32], where it was used to analyze economic data. The model consists of a latent random walk component observed in auto-regressive noise, where the variance of the observation noise innovations can switch between different values. In [21], this model was advocated to reflect the heteroscedasticity evident in the price index adjusted U.S./U.K. exchange rate during the late 19th and 20th centuries. The data consist of 1322 monthly log exchange rate values. We consider the case treated in [32] where the auto-regressive noise process is of order 2 and there are 4 switching states. In this model, $\mathcal{X} = \{1, 2, 3, 4\}$ and the discrete latent process $\{X_n\}$ is a Markov chain with transition matrix P_X. The observations $\{Y_n\}$ are log-exchange rate values. The latent process $\{\mu_n\}$ is a random walk and we denote by $\{\eta_n\}$ the auto-regressive noise process:

$$\begin{aligned} Y_n &= \mu_n + \eta_n, \\ \mu_n &= \mu_{n-1} + \sigma_\mu V_{n,1} \\ \eta_n &= a_1 \eta_{n-1} + a_2 \eta_{n-2} + \sigma_{\eta,X_n} V_{n,2} \end{aligned}$$

where $\{V_{n,1}\}$ and $\{V_{n,2}\}$ are i.i.d. $\mathcal{N}(0,1)$ noise sequences. In state-space form we then have

$$Z_n = \begin{bmatrix} \mu_n \\ \eta_n \\ \eta_{n-1} \end{bmatrix}, \quad A_\theta(x_n) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a_1 & a_2 \\ 0 & 1 & 0 \end{bmatrix}, \quad B_\theta(x_n) = \begin{bmatrix} \sigma_\mu & 0 & 0 \\ 0 & \sigma_{\eta,x_n} & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$C_\theta(x_n) = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \quad D_\theta(x_n) = F_\theta(x_n) = G_\theta(x_n) = 0, \quad \forall x_n.$$

The unknown parameters of the model are $\theta = [\sigma^2_{\mu,}\ \sigma^2_{\eta,1}\ \sigma^2_{\eta,2}\ \sigma^2_{\eta,3}\ \sigma^2_{\eta,4}\ a_1\ a_2\ P_X]$. Under symmetric priors the labeling of the discrete states is not identifiable. We consider the same prior distributions on the parameters and initial conditions on Z_0 as in [32] and we refer to the latter for full details, including a stability constraint on the auto-regressive coefficients (a_1, a_2). The only difference is that we do not impose an identifiability constraint a priori on $\sigma^2_{\eta,1}, \sigma^2_{\eta,2}, \sigma^2_{\eta,3}, \sigma^2_{\eta,4}$, but instead target the unidentified model and impose the ordering $\sigma^2_{\eta,1} < \sigma^2_{\eta,2} < \sigma^2_{\eta,3} < \sigma^2_{\eta,4}$ after sampling (see [31, 36] and references therein for various approaches to drawing inference in models with unidentifiable state labels).

We implemented an algorithm for this model with P_X incorporated into the sampling. Each iteration of the algorithm consisted of a sequence of two PMMH updates. The first holding P_X and $\sigma^2_{\eta,1}, \sigma^2_{\eta,2}, \sigma^2_{\eta,3}, \sigma^2_{\eta,4}$ constant and the second holding (a_1, a_2) and σ^2_μ constant (using standard arguments for Metropolis-within-Gibbs algorithms and Theorem 1 it is straightforward to show this sequence of updates is invariant with respect to the extended target distribution). After a couple of preliminary runs the following proposals were selected. A symmetric random walk proposal of standard deviation of 0.001 was used for (a_1, a_2) and for σ^2_μ a log-Gaussian random walk with log-domain standard deviation of 0.01. We used a mixture of log-Gaussian random walks for the unnormalised components of P_X and $\sigma^2_{\eta,1}, \sigma^2_{\eta,2}, \sigma^2_{\eta,3}, \sigma^2_{\eta,4}$. For each individual parameter, the mixture had two components, the first with weight 0.9 and standard deviation 0.05 in the log domain and the second with weight 0.1 and standard deviation 1 in the log domain. With these settings and $N = 200$ we achieved an overall acceptance rate of 0.2. This is a reasonable rate given the mixture proposals. The algorithm was run for 2×10^5 iterations after an initial burn-in of 10^4. Inferential summaries are presented in Figures 9-11. We note that there are some differences between the results we obtained and

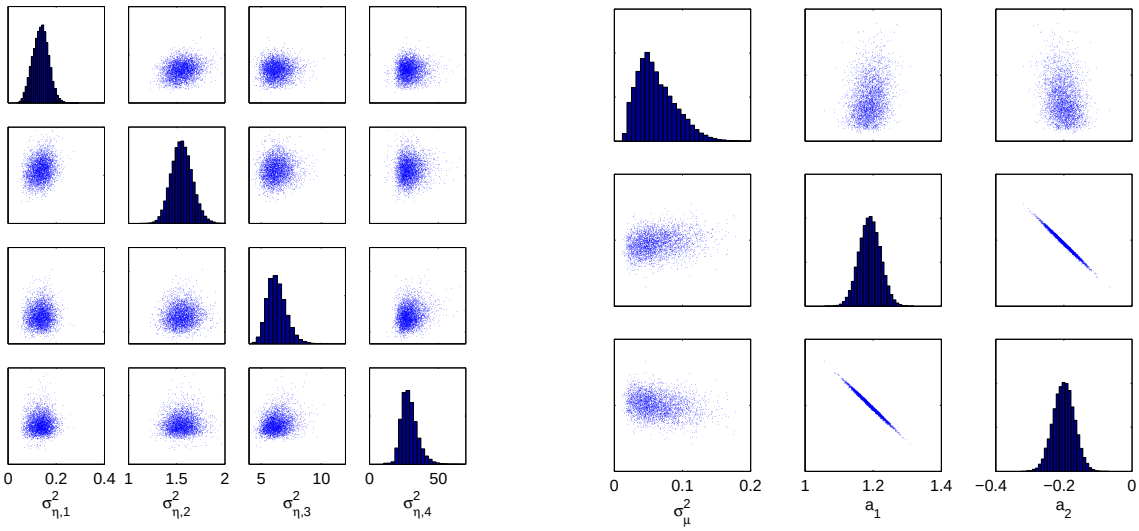

Figure 9: Example 3. Histogram estimates of posterior marginals and scatter plots of pairwise marginals for the exchange rate model.

those from [32], where a standard Gibbs sampler was applied. We conjecture that the latter had not fully explored the support of the posterior distribution. Noticeable differences are that the posterior marginal for $\sigma^2_{\eta,1}$ we obtain is more diffuse than that reported in [32] and we obtain a much flatter trajectory in the posterior estimates of $\{\mu_n\}$ in Figure 11. Another significant difference is that we obtain concentration of the marginal posterior over the auto-regressive coefficients (a_1, a_2) in a different region than that reported in [32]. Using other proposals for (a_1, a_2) we were not able to find another major mode. Furthermore the posterior marginal for σ^2_{μ} we obtained is concentrated on lower values. Overall, we feel that the ability to integrate out approximately the latent variables makes the PMMH algorithm a powerful tool: as these results demonstrate it gives us the chance to explore regions of posterior support which Gibbs sampling algorithms may struggle to find.

6 Discussion and extensions

In this article, we have extended the PMCMC methodology to be able to use the efficient DPF algorithm within MCMC. This yields a set of generic MCMC algorithms to perform Bayesian inference in SSSM. We have shown experimentally that these algorithms outperform state-of-the-art MCMC algorithms for a given computational complexity. Moreover the DPF can be easily parallelised so further substantial improvements could be obtained.

There are various possible extensions to this work. First, we have restricted ourselves to SSSM but the DPF can be applied to any model where the latent process is discrete-valued. This includes for example Dirichlet process mixtures [24] and the infinite hidden Markov model introduced in [42]. Compared to the SSSM framework, the differences are that, in these scenarios, X_n takes values in a set whose cardinality increases over time and computations required to evaluate the importance weights are not performed using the Kalman filter. However, the PMCMC methodology discussed here can be straightforwardly extended to these cases. Second, it would be possible to extend the DPF and the associated PMCMC methodology by using look-ahead techniques. In a look-ahead strategy with an integer lag L, we resample trajectories at time n by considering the weights proportional to $p_\theta(x_{1:n}|y_{1:n+L})$ instead of $p_\theta(x_{1:n}|y_{1:n})$ for the standard DPF. This is obviously more expensive than the DPF as computation of the weights involves summing over $x_{n+1:L}$ for each particle, but this might be of interest in scenarios where future observations are very informative about X_n. Third, it would be interesting and of practical significance to develop variants of PMCMC which involve tempering of the target distribution so as to improve mixing, in the spirit of standard simulated/parallel tempering schemes.

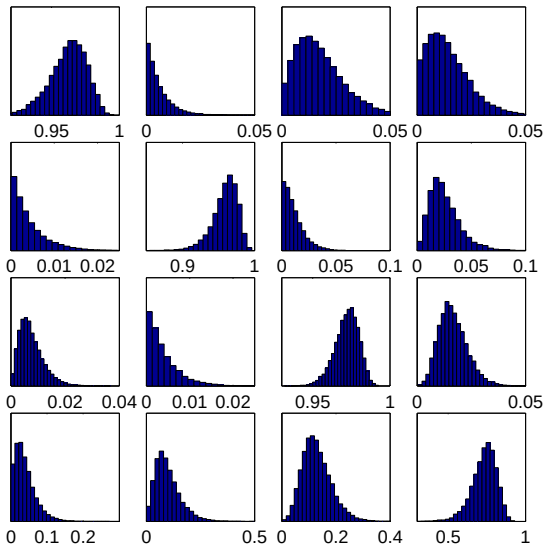

Figure 10: Example 3. Histogram estimates of marginal posterior distributions for entries of the state transition matrix P_X. Panes are arranged as per the transition matrix itself.

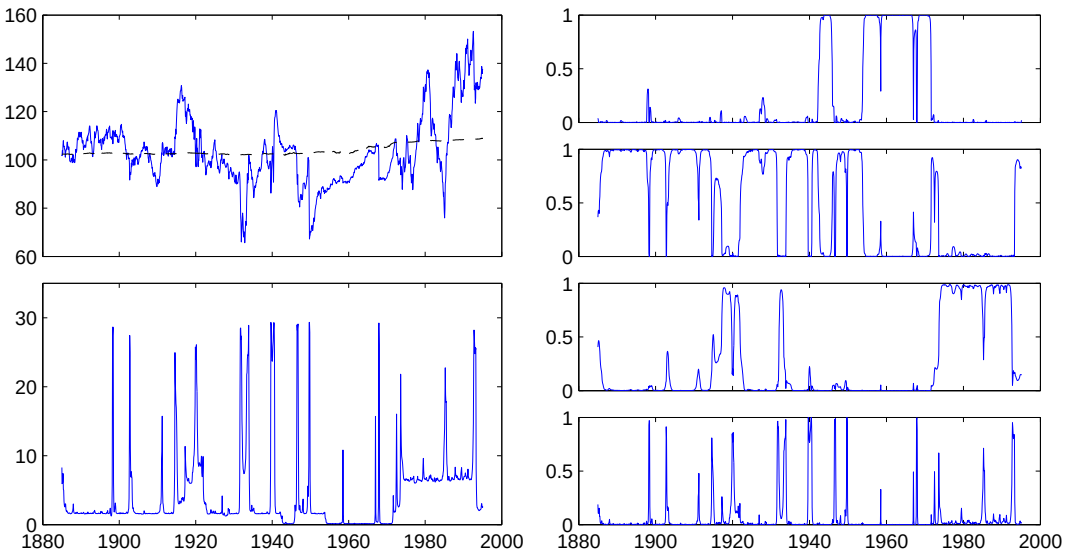

Figure 11: Example 3. Top left: data (solid) and $\mathbb{E}\left[\mu_n \middle| y_{1:T}\right]$ (dashed). Bottom left: $\mathbb{E}\left[\sigma^2_{\eta,X_n} \middle| y_{1:T}\right]$. Right: estimated posterior probabilities $p(X_n = j|y_{1:T})$ for, top to bottom, $j = 1, 2, 3, 4$.

A Kalman Filter

Conditional upon $X_{1:T} = x_{1:T}$, Eq. (2)-(3) defines a linear Gaussian state-space model. The Kalman filter allows us to compute recursively in time $p_\theta\left(z_n \middle| y_{1:n-1}, x_{1:n}\right) = \mathcal{N}\left(z_n; m^{z,\theta}_{n|n-1}\left(x_{1:n}\right), \Sigma^{z,\theta}_{n|n-1}\left(x_{1:n}\right)\right)$, $p_\theta\left(z_n \middle| y_{1:n}, x_{1:n}\right) = \mathcal{N}\left(z_n; m^{z,\theta}_{n|n}\left(x_{1:n}\right), \Sigma^{z,\theta}_{n|n}\left(x_{1:n}\right)\right)$ and the predictive density $g_\theta\left(y_n \middle| y_{1:n-1}, x_{1:n}\right) = \mathcal{N}\left(y_n; m^{y,\theta}_{n|n-1}\left(x_{1:n}\right), \Sigma^{y,\theta}_{n|n-1}\left(x_{1:n}\right)\right)$. For $n \geq 1$ these statistics are computed using the following recursion initialized with $m^x_{0|0} = m_0$, $\Sigma^x_{0|0} = \Sigma_0$

$$
\begin{aligned}
m^{z,\theta}_{n|n-1}\left(x_{1:n}\right) &= A_\theta(x_n) m^z_{n-1|n-1}\left(x_{1:n-1}\right) + F_\theta(x_n) u_n, \\
\Sigma^{z,\theta}_{n|n-1}\left(x_{1:n}\right) &= A_\theta(x_n) \Sigma^z_{n-1|n-1} A^{\mathrm{T}}_\theta(x_n) + B_\theta(x_n) B^{\mathrm{T}}_\theta(x_n), \\
m^{y,\theta}_{n|n-1}\left(x_{1:n}\right) &= C_\theta(x_n) m^{z,\theta}_{n|n-1}\left(x_{1:n}\right) + G_\theta(x_n) u_n, \\
\Sigma^{y,\theta}_{n|n-1}\left(x_{1:n}\right) &= C_\theta(x_n) \Sigma^{z,\theta}_{n|n-1}\left(x_{1:n}\right) C^{\mathrm{T}}_\theta(x_n) + D_\theta(x_n) D^{\mathrm{T}}_\theta(x_n), \\
m^{z,\theta}_{n|n}\left(x_{1:n}\right) &= m^{z,\theta}_{n|n-1}\left(x_{1:n}\right) + \Sigma^{z,\theta}_{n|n-1}\left(x_{1:n}\right) C^{\mathrm{T}}_\theta(x_n) \left[\Sigma^{y,\theta}_{n|n-1}\left(x_{1:n}\right)\right]^{-1} \left(y_n - m^{y,\theta}_{n|n-1}\left(x_{1:n}\right)\right), \\
\Sigma^{z,\theta}_{n|n}\left(x_{1:n}\right) &= \Sigma^{z,\theta}_{n|n-1}\left(x_{1:n}\right) - \Sigma^{z,\theta}_{n|n-1}\left(x_{1:n}\right) C^{\mathrm{T}}_\theta(x_n) \left[\Sigma^{y,\theta}_{n|n-1}\left(x_{1:n}\right)\right]^{-1} C_\theta\left(x_n\right) \Sigma^{z,\theta}_{n|n-1}\left(x_{1:n}\right).
\end{aligned}
$$

B Backward Sampling

A key component of the backward sampling algorithm is the evaluation of the backward weight

$$
V^\theta_n\left(x_{1:n} \middle| x'_{n+1:T}\right) \propto W^\theta_n(x_{1:n}) p_\theta(x'_{n+1:T}|x_{1:n}) p_\theta(y_{n+1:T}|y_{1:n}, x_{1:n}, x'_{n+1:T})
$$

for each candidate sub-trajectory $x_{1:n}$ and where $x'_{n+1:T}$ is the complementing sub-trajectory which has been obtained from previous steps of the backward sampling procedure. Central to the computation of this weight is the identity

$$
p_\theta(y_{n+1:T}|y_{1:n}, x_{1:n}, x'_{n+1:T}) = \int p_\theta(y_{n+1:T}|z_n, x'_{n+1:T}) p_\theta(z_n|x_{1:n}, y_{1:n}) dz_k, \tag{27}
$$

where $p_\theta(z_n|x_{1:n}, y_{1:n})$ is the Gaussian conditional filtering density associated with the sub-trajectory $x_{1:n}$ and is specified by its mean vector $m^{z,\theta}_{n|n}\left(x_{1:n}\right)$ and co-variance matrix $\Sigma^{z,\theta}_{n|n}\left(x_{1:n}\right)$. In order to compute (27) (at least up to a constant of proportionality) it is necessary to obtain the coefficients of z_n in $p_\theta(y_{n+1:T}|z_n, x'_{n+1:T})$. The latter can be expressed as

$$
p_\theta(y_{n+1:T}|z_n, x'_{n+1:T}) \propto \exp\left[-\frac{1}{2}\left(z_n^T \Xi_n z_n - 2\mu_n^T z_n\right)\right]
$$

where Ξ_n and μ_n are respectively a matrix and vector of appropriate dimension, both depending on $x'_{n+1:T}$, $y_{n+1:T}$ and θ. In the following this dependence is suppressed from the notation for convenience. For ease of presentation we use the similarly abusive conventions in writing $m_n = m^{z,\theta}_{n|n}\left(x_{1:n}\right)$, $\Sigma_n = \Sigma^{z,\theta}_{n|n}\left(x_{1:n}\right)$, $A_n = A_\theta(x_n)$, $B_n = [B_\theta(x_n) \;\; 0_{z\times w}]$, $C_n = C_\theta(x_n)$, $D_n = [0_{y\times v} \;\; D_\theta(x_n)]$, $F_n = F_\theta(x_n)$, $G_n = G_\theta(x_n)$. Then let Υ_n be a matrix satisfying $\Sigma^{z,\theta}_{n|n}\left(x_{1:n}\right) = \Upsilon_n \Upsilon_n^T$. We have

$$
\begin{aligned}
&p_\theta(y_{n+1:T}|y_{1:n}, x_{1:n}, x'_{n+1:T}) \\
&\propto \exp\left(-\frac{1}{2}\left[m_n^T \Xi_n m_n - 2\mu_n^T m_n - \left(\mu_n - \Xi_n m_n\right)^T \Upsilon_n \left(\Upsilon_n \Xi_n \Upsilon_n + I\right)^{-1} \Upsilon_n^T \left(\mu_n - \Xi_n m_n\right)\right]\right) \\
&\quad \times \left|\Upsilon_n^T \Xi_n \Upsilon_n + I\right|^{-1/2}.
\end{aligned} \tag{28}
$$

where I is the identity matrix of appropriate dimension. We now specify equations for updating (μ_n, Ξ_n), which are given without proof of validity: they are a direct application of Lemmata 1 and 2 in [33]. As in

[33], for simplicity we present recursions only for the case in which the observations are scalar-valued, but they can readily be extended to the vector-valued case. Let

$$\begin{aligned}
r_{n+1} &= \left(C_{n+1}B_{n+1} + D_{n+1}\right)\left(C_{n+1}B_{n+1} + D_{n+1}\right)^T, \\
\Phi_{n+1} &= B_{n+1}\left(B_{n+1}^T C_{n+1}^T + D_{n+1}^T\right)/r_{n+1}, \\
\Lambda_{n+1} &= \left(1 - \Phi_{n+1}C_{n+1}^T\right)A_{n+1}, \\
a_{n+1} &= \left(1 - \Phi_{n+1}C_{n+1}^T\right)F_{n+1}u_{n+1} - \Phi_{n+1}G_{n+1}u_{n+1},
\end{aligned}$$

and let Γ_{n+1} be a matrix which satisfies

$$\Gamma_{n+1}\Gamma_{n+1}^T = B_{n+1}\left(I - \frac{1}{r_{n+1}}\left(B_{n+1}^T C_{n+1}^T + D_{n+1}^T\right)\left(B_{n+1}^T C_{n+1}^T + D_{n+1}^T\right)^T\right)B_{n+1}^T.$$

The recursion for (μ_n, Ξ_n) is then given by

- Set $\Xi_T = 0$, $\mu_T = 0$.
- For $n = T-1, ..., 1$

$$\begin{aligned}
M_{n+1} &= \Gamma_{n+1}^T \Xi_{n+1}\Gamma_{n+1} + I, \\
\Xi_n &= \Lambda_{n+1}^T\left(\Xi_{n+1} - \Xi_{n+1}\Gamma_{n+1}M_{n+1}^{-1}\Gamma_{n+1}^T\Xi_{n+1}\right)\Lambda_{n+1} + A_{n+1}^T C_{n+1}^T C_{n+1}A_{n+1}\frac{1}{r_{n+1}}, \\
\mu_n &= \Lambda_{n+1}^T\left(I - \Xi_{n+1}\Gamma_{n+1}M_{n+1}^{-1}\Gamma_{n+1}^T\right)\left(\mu_{n+1} - \Xi_{n+1}\left(a_{n+1} + \Phi_{n+1}y_{n+1}\right)\right) \\
&\quad + A_{n+1}^T C_{n+1}^T\left(y_{n+1} - G_{n+1}u_{n+1} - C_{n+1}F_{n+1}u_{n+1}\right)\frac{1}{r_{n+1}}.
\end{aligned}$$

C Proofs

Proof of Theorem 1. We obtain from Eq. (22)-(23)-(24) that on the event $x_{1:T} \in \mathbf{S}_T$,

$$\begin{aligned}
\frac{\pi_\theta^N(x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)}{w_T^\theta(x_{1:T})\ \psi_\theta^N(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)} &= \frac{p_\theta(x_{1:T}|\, y_{1:T})\left\{\prod_{n=2}^T \mathbb{I}[x_{1:n} \in \mathbf{s}_n]\right\}}{w_T^\theta(x_{1:T})\prod_{n=2}^T r_n^N(x_{1:n} \in \mathbf{s}_n|\mathbf{w}_{n-1}^\theta)} \\
&= \frac{p_\theta(x_{1:T}|\, y_{1:T})\left\{\prod_{n=2}^T \mathbb{I}[x_{1:n} \in \mathbf{s}_n]\right\}}{w_T^\theta(x_{1:T})\prod_{n=1}^{T-1}\left(1 \wedge c_n w_n^\theta(x_{1:n})\right)}.
\end{aligned}$$

It follows from Eq. (10)-(11) that on the event $x_{1:T} \in \mathbf{S}_T$ the normalized weight can be expanded as follows

$$\begin{aligned}
w_T^\theta(x_{1:T}) &= \nu_\theta(x_1)g_\theta(y_1|x_1)\prod_{n=2}^T f_\theta(x_n|x_{1:n-1})g_\theta(y_n|y_{1:n-1}, x_{1:n}) \\
&\times \prod_{n=1}^{T-1}\frac{1}{1 \wedge c_n w_n^\theta(x_{1:n})}\prod_{n=1}^T\left\{\sum_{x'_{1:n} \in \mathbf{s}_n}\overline{w}_n^\theta(x'_{1:n})\right\}^{-1}
\end{aligned} \tag{29}$$

Hence, using Eq. (12)-(13), we obtain

$$\frac{\pi_\theta^N(x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)}{w_T^\theta(x_{1:T})\ \psi_\theta^N(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)} = \frac{\widehat{p}_\theta(y_{1:T})}{p_\theta(y_{1:T})}. \tag{30}$$

From (30), we can now easily establish that an MH sampler of target density (24) and proposal density (25) admits indeed Eq. (17) as MH ratio and the first part of the theorem follows. The second part of the proof is a direct consequence of Theorem 1 in [2] and (A1). ■

Proof of Proposition 1. First note that, from Eq. (24) and Eq. (29), for $\theta, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T$ in the support of $\pi^N(\theta, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$,

$$
\begin{aligned}
\pi^N(x_{1:T}|\theta, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T) &\propto p_\theta(x_{1:T}|\, y_{1:T}) \left\{ \prod_{n=2}^{T} \mathbb{I}[x_{1:n} \in \mathbf{s}_n] \right\} \frac{\psi_\theta^N(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)}{\prod_{n=2}^{T} r_n^N(x_{1:n} \in \mathbf{s}_n|\mathbf{w}_{n-1}^\theta)} \\
&\propto \nu_\theta(x_1) g_\theta(y_1|x_1) \prod_{n=2}^{T} f_\theta(x_n|x_{1:n-1}) g_\theta(y_n|y_{1:n-1}, x_{1:n}) \\
&\quad \times \left\{ \prod_{n=2}^{T} \mathbb{I}[x_{1:n} \in \mathbf{s}_n] \right\} \prod_{n=1}^{T-1} \frac{1}{1 \wedge c_n w_n^\theta(x_{1:n})} \qquad (31) \\
&\propto w_T^\theta(x_{1:T}).
\end{aligned}
$$

Furthermore, for $1 \leq n \leq T-1$,

$$
\begin{aligned}
\pi^N(x_{1:n}|\theta, x_{n+1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n) &\propto p_\theta(x_{1:T}|y_{1:T}) \left\{ \prod_{k=2}^{n} \mathbb{I}[x_{1:k} \in \mathbf{s}_k] \right\} \frac{\psi_\theta^N(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n)}{\prod_{k=2}^{n} r_k^N(x_{1:k} \in \mathbf{s}_k|\mathbf{w}_{k-1}^\theta)} \\
&\propto \frac{p_\theta(x_{1:n}|y_{1:n})}{\prod_{k=2}^{n} r_k^N(x_{1:k} \in \mathbf{s}_k|\mathbf{w}_{k-1}^\theta)} p_\theta(x_{n+1:T}|x_{1:n}) p_\theta(y_{n+1:T}|y_{1:n}, x_{1:T}) \\
&\quad \times \left\{ \prod_{k=2}^{n} \mathbb{I}[x_{1:k} \in \mathbf{s}_k] \right\} \\
&\propto w_n^\theta(x_{1:n}) p_\theta(x_{n+1:T}|x_{1:n}) p_\theta(y_{n+1:T}|y_{1:n}, x_{1:T}) \\
&\propto v_n^\theta(x_{1:n}\,|x_{n+1:T}), \qquad (32)
\end{aligned}
$$

where for the third proportionality we have used (19)-(20) and an expansion of $W_n^\theta(x_{1:n})$ which is the direct analogue of (29) but for final time index n.
To establish the assertion of the proposition we use an inductive argument over the iterations of the backward sampling algorithm (indexed by $n = T, T-1, ..., 1$). The inductive hypothesis is that for some index n satisfying $1 < m < n < T$ of the backward sampling procedure, $(X'_{1:T}, \mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_n)$ obtained immediately after sampling from the backwards weights is distributed according to the marginal distribution $\sum_{\mathbf{s}_{n+1}} ... \sum_{\mathbf{s}_T} \pi_\theta^N(x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$. This implies $(X'_{n:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{n-1})$ is distributed according to $\sum_{x_{1:n-1}} \sum_{\mathbf{s}_n} ... \sum_{\mathbf{s}_T} \pi_\theta^N(x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$. Then at time step $n-1$, due to Eq. (32), $(X'_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{n-1})$ obtained after sampling from the backward weights is distributed according to $\sum_{\mathbf{s}_n} ... \sum_{\mathbf{s}_T} \pi_\theta^N(x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$ and thus $X'_{1:T}$ is distributed according to $\sum_{\mathbf{s}_1} ... \sum_{\mathbf{s}_T} \pi_\theta^N(x_{1:T}, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T) = p_\theta(x_{1:T}|y_{1:T})$. Next note that, due to Eq. (31) the first step of the backward sampling procedure draws from $\pi_\theta^N(x_{1:T}|\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)$. The proof is then complete under the assumption of the proposition. ■

Proof of Theorem 2. For part 1, it is easy to check that steps $1-4$ of the PG algorithm define a collapsed Gibbs sampler targeting Eq. (24). This follows from Proposition 1 and the fact that the conditional DPF update, given a value of θ and $x_{1:T}$, is nothing but an algorithm sampling from

$$
\left\{ \prod_{n=2}^{T} \mathbb{I}[x_{1:n} \in \mathbf{s}_n] \right\} \frac{\psi_\theta^N(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)}{\prod_{n=2}^{T} r_\theta^N(x_{1:n} \in \mathbf{s}_n|\mathbf{w}_{n-1}^\theta)}.
$$

For part 2, we focus on establishing irreducibility and aperiodicity of the transition probability of this algorithm. We denote by $\mathcal{L}_G$ the law of the Gibbs sampler to which assumption 2 applies and $\mathcal{L}_{PG}^N$ the law of the PG sampler using N particles.

For any set U write 2^U for the power set of U and let $\mathcal{B}(\Theta)$ denote a σ-algebra on Θ. Let $A \times B \times C \in \mathcal{B}(\Theta) \times 2^{\mathcal{X}^T} \times \prod_{n=1}^{T} 2^{\mathcal{P}(\mathcal{X}^n)}$ be such that $\pi^N(\theta \in A, X_{1:T} \in B, \mathbf{S}_1, ..., \mathbf{S}_{T-1} \in C) > 0$. It follows that $\pi((\theta, X_{1:T}) \in A \times B) > 0$ and then from irreducibility of the corresponding Gibbs sampler (assumption 2) there exists a finite j such that $\mathcal{L}_G((\theta(j), X_{1:T}(j)) \in A \times B) > 0$.

From the definition of the conditional DPF update, it is straightforward to check that, for any $\theta \in \Theta$, $N \geq 2$, given any $x_{1:T}$ and for any time step, any particle which has positive weight immediately before resampling has a positive probability of surviving that resampling step. Thus, by an inductive argument in n, any point in the support of $p_\theta(x_{1:T}|y_{1:T})$ has positive probability of being assigned a positive weight

at time T. It then follows from the above arguments that $A \times B$ is marginally an accessible set of the PG sampler for the same j: i.e. $\mathcal{L}_{PG}^N((\theta(j), X_{1:T}(j)) \in A \times B) > 0$. Furthermore, as the conditional DPF update corresponds to drawing from the conditional of π^N given θ and $X_{1:T}$,

$$\mathcal{L}_{PG}^N((\theta(j+1), X_{1:T}(j+1), \mathbf{S}_1(j+1), ..., \mathbf{S}_T(j+1)) \in A \times B \times C) > 0$$

and irreducibility follows. Furthermore, aperiodicity of the PG sampler holds by contradiction: if the PG sampler were periodic, then the Gibbs sampler would be too; this violates Assumption A2.■

References

[1] Andrieu, C., Doucet, A. and Holenstein, R. (2010) Particle Markov chain Monte Carlo methods (with discussion). *J. Roy. Stat. Soc.* B, **72**, 269-342.

[2] Andrieu, C. and Roberts, G.O. (2009) The pseudo-marginal approach for efficient computation. *Annals Statist.*, **37**, 697-725.

[3] Barembruch, S., Garivier, A. and Moulines, E. (2009) On approximate maximum likelihood methods for blind identification: how to cope with the curse of dimensionality. *IEEE Trans. Signal Proc.*, **57**(11), 4247-4259.

[4] Barry, D. and Hartigan, J. A., (1993) A Bayesian analysis for change point problems. *J. Am. Stat. Assoc.*, **88**(421), 309-319.

[5] Billio, M. and Monfort, A. (1998) Switching state space models: likelihood, filtering and smoothing. *J. Stat. Planning Inf.*, **68**, 65-103.

[6] Carpenter, J., Clifford, P. and Fearnhead, P. (1999) An improved particle filter for non-linear problems. *IEE Proc.* F, **146**, 2-7.

[7] Cappé, O., Moulines, E. and Rydén, T. (2005) *Inference in Hidden Markov Models.* New York: Springer-Verlag.

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

[9] Carter, C.K. and Kohn, R. (1996). Markov chain Monte Carlo in conditionally Gaussian state space models. *Biometrika*, **83**, 589-601.

[10] Chen, R. and Liu, J.S. (2000) Mixture Kalman filters. *J. Roy. Stat. Soc.* B, **62**, 493-508.

[11] Chib, S. (1996) Calculating posterior distributions and modal estimates in Markov mixture models. *J. Econometrics*, **75**, 79-97.

[12] Costa, O.L.V., Fragoso, M. D. and Marques, R.P. (2005) *Discrete-Time Markovian Jump Linear Systems.* Springer-Verlag.

[13] De Jong, P. and Shephard, N. (1995) The simulation smoother for time series models. *Biometrika*, **82**, 339-350.

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

[15] Cérou. F., Del Moral, P. and Guyader, A. (2008) A non asymptotic variance theorem for unnormalized Feynman-Kac particle models. HAL-INRIA Research Report-6716.

[16] Del Moral, P., Doucet, A. and Singh S.S. (2009) A Backward Particle Interpretation of Feynman-Kac Formulae. HAL-INRIA Research Report-7019.

[17] Doucet, A., de Freitas, J.F.G. and Gordon, N.J. (eds.) (2001) *Sequential Monte Carlo Methods in Practice.* New York: Springer-Verlag.

[18] Doucet, A., Godsill, S.J. and Andrieu, C. (2000) On sequential Monte Carlo sampling methods for Bayesian filtering. *Statist. Comput.*, **10**, 197-208.

[19] Doucet, A., Gordon, N.J. and Krishnamurthy, V. (2001) Particle filters for state estimation of jump Markov linear systems. *IEEE Trans. Signal Proc.*, **49**, 613-624.

[20] Durbin, J. and Koopman, S.J. (2002) A simple and efficient simulation smoother for state space time series analysis. *Biometrika*, **89**, 603-616.

[21] Engle, C. and Kim, C-J. (1999) The long-run U.S./U.K. real exchange rate. *J. Money, Credit and Banking,* **31**, 335-356.

[22] Fearnhead, P. (1998) Sequential Monte Carlo methods in filter theory. PhD Thesis, Department of Statistics, University of Oxford.

[23] Fearnhead, P. and Clifford, P. (2003) Online inference for well-log data. *J. Roy. Stat. Soc.* B, **65**, 887-899.

[24] Fearnhead, P. (2004) Particle filters for mixture models with an unknown number of components. *Statist. Comput.*, **14**, 11-21.

[25] Fearnhead, P., and Liu, Z. (2007). On-line inference for multiple changepoint problems. *J. Roy. Statist. Soc.* B, **69**, 589-605.

[26] Fearnhead, P. and Liu, Z. (2009) Efficient Bayesian analysis of multiple changepoint models with dependence across segments. *Statist. Comput.*, to appear.

[27] Fong, W., Godsill, S.J., Doucet, A. and West, M. (2002) Monte Carlo smoothing with application to audio signal enhancement. *IEEE Trans. Signal Proc.*, **50**(2), 438-449.

[28] O Ruanaidh, J.J.K. and Fitzgerald, W.J. (1996) *Numerical Bayesian Methods Applied to Signal Processing.* New York: Springer.

[29] Flury, T. and Shephard, N. (2010) Bayesian inference based only on simulated likelihood: particle filter analysis of dynamic economic models. *Econometrics Theory*, to appear.

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

[31] Frühwirth-Schnatter, S (2001). Markov chain Monte Carlo estimation of classical and dynamic switching and ixture Models. *J. Amer. Statist. Assoc.* **96**(453): 194-209.

[32] Frühwirth-Schnatter, S. (2006) *Finite Mixture and Markov Switching Models.* New York: Springer Verlag.

[33] Gerlach, R., Carter, C. K. and Kohn, R. (2000) Efficient Bayesian inference for dynamic mixture models, *J. Amer. Statist. Assoc.*, **95**, 819–828.

[34] Giordani, P. and Kohn, R. (2008) Efficient Bayesian inference for multiple change-point and mixture innovation models. *J. Business Economic Statist.*, **26**, 66-77.

[35] Giordani, P., Kohn, R. and van Dijk, D. (2007) A unified approach to nonlinearity, structural change and outliers, *J. Econometrics*, **137**, 112-133.

[36] Jasra, A., Holmes, C. C. and Stephens D. A. (2005) Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. *Statist. Sci.* **20**(1), 50-67.

[37] Kim, C.J. and Nelson, C. (1999) *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*, MIT Press.

[38] Kitagawa, G. (1996) Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *J. Comp. Graph. Statist.*, **5**, 1-25.

[39] Lee, A., Yau, C., Giles, M., Doucet, A. and Holmes, C.C. (2009) On the utility of graphics cards to perform massively parallel simulation with advanced Monte Carlo methods. Technical report Oxford-Man Institute of Quantitative Finance. Available at http://arxiv.org/abs/0905.2441.

[40] Liu, J.S. (2001) *Monte Carlo Strategies in Scientific Computing.* New York: Springer Verlag.

[41] Shephard, N. (1994). Partial non-Gaussian state space. *Biometrika*, **81**, 115-131.

[42] Teh, Y.W., Jordan, M.I., Beal, M.J. and Blei, D.M. (2006) Hierarchical Dirichlet processes. *J. Am. Statist. Assoc.*, **101**, 1566-1581.