Nonlinear state space smoothing using the conditional particle filter

Andreas Svensson * Thomas B. Schön * Manon Kok **

* Department of Information Technology, Uppsala University, Sweden
(e-mail: {andreas.svensson, thomas.schon}@it.uu.se)
** Division of Automatic Control, Linköping University, Sweden
(e-mail: manko@isy.liu.se)

Abstract: To estimate the smoothing distribution in a nonlinear state space model, we apply the conditional particle filter with ancestor sampling. This gives an iterative algorithm in a Markov chain Monte Carlo fashion, with asymptotic convergence results. The computational complexity is analyzed, and our proposed algorithm is successfully applied to the challenging problem of sensor fusion between ultra-wideband and accelerometer/gyroscope measurements for indoor positioning. It appears to be a competitive alternative to existing nonlinear smoothing algorithms, in particular the forward filtering-backward simulation smoother.

Keywords: Smoothing, Particle filters, Nonlinear systems, State estimation, Monte Carlo method, Sensor fusion, Position estimation.

1. INTRODUCTION

Consider the (time-varying, nonlinear, non-Gaussian) state space model (SSM)

\[ x_{t+1} | x_t \sim f_t(x_{t+1} | x_t), \quad (1a) \]
\[ y_t | x_t \sim g_t(y_t | x_t), \quad (1b) \]

with \( x_1 \sim \mu(x_1) \). We use a probabilistic notation, with \( \sim \) meaning distributed according to. The index variable \( t = 1, 2, \ldots, T \) is referred to as the time, although different interpretations are possible. The variable \( x_t \in \mathbb{R}^{n_x} \) is referred to as the state, and an exogenous input \( u_t \) is possible to include in \( f_t \) and \( g_t \). To ease the notation, the possible time dependence of \( f \) and \( g \) will be suppressed in the sequel.

For some applications, for instance system identification, the distribution of the states when the model and all measurements are known,

\[ p(x_{1:T} | y_{1:T}), \quad (2) \]

is of interest. We will refer to (2) as the smoothing distribution. The smoothing distribution is for the general model (1) not available on closed form, and approximations are necessary.

In this paper, we will present a method generating Monte Carlo samples from the smoothing distribution, akin to a particle filter. The key idea is the iterated use of a so-called conditional particle filter, which (after sufficiently many iterations) generates samples from the smoothing distribution, as illustrated in Figure 1.

An overview of existing particle smoothers addressing the problem of generating samples from (2) is provided by Lindsten and Schön (2013). In this work we will in particular compare and relate our developments to the so-called forward filtering-backward simulation (FFBSi) smoother introduced by Douc et al. (2011). The work

\[ \text*{This work was supported by the project Probabilistic modelling of dynamical systems (Contract number: 621-2013-5524) funded by the Swedish Research Council (VR) and CADICS, a Linnaeus Center, funded by the Swedish Research Council (VR).} \]

**Fig. 1. Idea on how to use the conditional particle filter to sample from the smoothing distribution \( p(x_{1:T} | y_{1:T}) \) (the true distribution is shown in grey, with time \( t \) at the horizontal axis and state \( x \) on the vertical axis): The first conditional trajectory (blue line in the top plot is an arbitrary initialization. After some iterations, the conditional trajectories converge toward the smoothing distribution.**
by Kitagawa (1996) and Briers et al. (2010) are both of interest in that they approach a similar problem using particle filters, the latter also taking inspiration from the two-filter formula. The work of Pillonetti and Bell (2008) is closely related in that they also employ a Markov chain Monte Carlo (MCMC) construction, but they consider the special case of Gaussian noise (f and g can still be nonlinear though).

An alternative to using samples to represent (2) is to solve a linearized version of the problem, by combining the extended Kalman filter (Smith et al., 1962; Schmidt, 1966) with the RTS-smoother (Rauch et al., 1965) to solve the linearized problem analytically. The work of Särkkä (2008) on the unscented RTS-smoother is along the same line.

We will make use of the recently introduced particle Gibbs with ancestor sampling (PGAS) method of Lindsten et al. (2014), which is a member of the particle Markov chain Monte Carlo (PMCMC) methods, first introduced by Andrieu et al. (2010). The contributions of this paper are twofold: First, we show how to make use of PGAS in solving the nonlinear state space smoothing problem. Second, the proposed algorithm is applied to solve a non-trivial indoor positioning problem.

Matlab code for all simulated examples as well as details on the indoor positioning problem is available in a technical report (Svensson et al., 2015) via the first author’s homepage.

2. PARTICLE METHODS

We assume the reader has some basic familiarity with particle filters (see, e.g., Doucet and Johansen (2011) for an introduction), but to set the notation we will start by a brief summary of particle filters and particle smoothers.

2.1 Particle filters

In the search for a numerical approximation to \( p(x_{1:T}|y_{1:T}) \), the following factorization is useful

\[
p(x_{1:T}, y_{1:T}) = \mu(x_1) \prod_{t=1}^{T} g(y_t|x_t) \prod_{t=1}^{T-1} f(x_{t+1}|x_t). \tag{3}
\]

as it allows for the following recursion to be derived (using Bayes’ rule and \( p(y_{1:T}) = p(y_1) \prod_{t=2}^{T} p(y_{t+1}|y_{1:t}) \))

\[
p(x_{1:t}|y_{1:t}) = \frac{g(y_t|x_t)f(x_{t+1}|x_t)}{p(y_t|y_{1:t-1})} \frac{p(x_{1:t-1}|y_{1:t-1})}{p(x_{1:t-1}|y_{1:t-1})}. \tag{4*}
\]

This factorization can be used to motivate the particle filter. Starting with a particle (Monte Carlo) approximation of \( p(x_{1:T}|y_{1:T}) \) as \( N \) particles (samples), \( \{4*\} \) can be applied to obtain a particle approximation of \( p(x_{1:2}|y_{1:2}) \). Repeating this \( T-1 \) times, a particle approximation of \( p(x_{1:T}|y_{1:T}) \) as \( N \) weighted particles \( \{4*\} \) is found.

The notation used in Algorithm 1 is

\[
W(x_{1:T}) \triangleq g(y_{1:T}|x_{1:T})/q_t(x_{1:T}|y_{1:T}) \tag{5a}
\]

\[
W(x_{1:t}) \triangleq g(y_t|x_t)f(x_{t+1}|x_t)/q_t(x_{1:t-1}, y_t) \tag{5b}
\]

Here, \( q_t \) denotes the proposal distribution, which is used to propagate the particles from time \( t \) to time \( t+1 \). If the proposal \( q_t \) is chosen as \( f \), then \( \{5b\} \) simplifies to

\[
W(x_{1:t}) = \frac{g(y_t|x_t)f(x_{t+1}|x_t)}{q_t(x_{1:t-1}, y_t)}. \tag{5b'}
\]

Algorithm 1 Particle Filter

Output: A weighted particle system \( \{x_{1:T}, w_{1:T}\} \)

approximating \( p(x_{1:T}|y_{1:T}) \) for \( t = 1, 2, \ldots, T \).

1. Draw \( x_1 \sim q_1(x_1) \).
2. Set \( w_1 = W_1(x_1) \).
3. for \( t = 2 \) to \( T \) do
   4. Draw \( a_t \) with \( \Pr (a_t = j) \propto w_{t-1} \).
   5. Draw \( x_t \sim q_t(x_t|x_{t-1}, y_t) \).
   6. Set \( x_{1:t} = (x_{1:t-1}, x_t) \).
   7. Set \( w_t = W(x_{1:t}) \).
   8. end for

\( W(x_{1:t}) = g(y_t|x_t) \) resulting in the so-called bootstrap particle filter (Gordon et al., 1993). The main steps in Algorithm 1, namely 4, 5, and 7 are often referred to as resampling, propagation, and weighting, respectively.

2.2 Forward – backward particle smoothers

A natural way to find the smoothing distribution for SSMS is to first apply a (forward) filter, and then add a backward pass, adjusting for the ‘new’ information about the state \( x_t \) at time \( t \) obtained from the later measurements \( y_{t+1}, y_{t+2}, \ldots \). Such an example is the RTS smoother for the linear Gaussian case (Rauch et al., 1965), but also the more recent particle-based FFBSi algorithm, see, e.g., Lindsten and Schön (2013) for a recent overview.

The algorithm for FFBSi is not repeated here, but we note that it relies on the two steps

(1) A particle filter with \( N \) particles.
(2) A backward simulation drawing \( M \) (uncorrelated) samples from \( p(x_{1:T}|y_{1:T}) \) using the \( N \) particles from Step (1).
The computational complexity of FFBSi is basically $O(NM)$, although some improvements can be achieved, see (Lindsten and Schön, 2013, Section 3.3).

To prepare for the upcoming discussions on convergence, let us briefly comment on the convergence properties of FFBSi.

How well can a function $h(x_{1:T})$ be approximated as $\hat{h}$ using samples from FFBSi? Let $h_{\text{FFBSi}}^N = \frac{1}{N} \sum_{i=1}^{N} h(x_{1:T})^i$ denote an approximation of $h(x_{1:T})$ based on $M = N$ backward trajectories. Under some fairly mild assumptions, it has been shown (Douc et al., 2011, Corollary 9) that there exists a $\sigma_{\text{FFBSi}}^2 < \infty$ such that

$$\sqrt{N} (h_{\text{FFBSi}}^N - \mathbb{E}[h(x_{1:T})|y_{1:T}])$$

converges weakly to $N(0, \sigma_{\text{FFBSi}}^2)$.

To summarize, the convergence rate for FFBSi is $\sqrt{N}$, subject to a computational complexity of $O(N^2)$.

3. SMOOTHING USING THE CONDITIONAL PARTICLE FILTER

The smoothing methodology discussed in Section 2.2 builds on a forward-backward strategy. The MCMC idea offers a fundamentally different way to construct a smoother, without explicitly running a backward pass, but iteratively running a conditional particle filter as illustrated in Figure 1. As we will see, this opens up for a reduced computational complexity. The origin of the method dates back to the introduction of the PMC-MCMC methods by Andrieu et al. (2010), with important recent contributions from Lindsten et al. (2014).

First, the conditional particle filter with ancestor sampling (CPF-AS) will be introduced (Section 3.1), followed by a brief introduction to MCMC (Section 3.2), and they will in the next step (Section 3.3) be combined to form a particle smoother. The convergence properties and the computational complexity of the smoother are then examined in Section 3.4 and Section 3.5, respectively.

3.1 Conditional particle filter with ancestor sampling

The CPF-AS is thoroughly described by Lindsten et al. (2014), and also presented as Algorithm 2. In many aspects, the CPF-AS is similar to a regular particle filter, Algorithm 1, but with one particle trajectory $x_{1:T}[k]$ specified a priori (corresponding to trajectory number $N$ in Algorithm 2).

The CPF-AS generates $N$ weighted particle trajectories $\{x_{1:T}^i, w_i^T\}_{i=1}^{N}$. With the original formulation of the conditional particle filter in Andrieu et al. (2010), one of these trajectories is predestined to be $x_{1:T}[k]$. Extending the algorithm with ancestor sampling the CPF-AS is obtained and the resulting trajectories $\{x_{1:T}^i, w_i^T\}_{i=1}^{N}$ are still influenced by $x_{1:T}[k]$, but in a somewhat more involved way, as the conditional trajectory may be ‘partly’ replaced by a new trajectory; see Algorithm 2 for details.

By sampling one of the trajectories $x_{1:T}[k+1] = x_{1:T}^i$ obtained from the CPF-AS with $P(i = j) \propto w_i^j$, the CPF-AS can be seen as a procedure to stochastically map $x_{1:T}[k]$ onto another trajectory $x_{1:T}[k+1]$.

A Rao-Blackwellized formulation for mixed linear/nonlinear models analogous to Schön et al. (2005) is also possible, see Svensson et al. (2014) for details.

Algorithm 2 Conditional particle filter with ancestor sampling (CPF-AS)

**Input:** Trajectory $x_{1:T}[k]$

**Output:** Trajectory $x_{1:T}[k+1]$

1. Draw $x_1 \sim q_1(x_1)$ for $i = 1, \ldots, N - 1$.
2. Set $x_N^i = x_1[k]$.
3. Set $w_1^i = W_1(x_1)$ for $i = 1, \ldots, N$.
4. For $i = 2$ to $T$ do
5. Draw $a_i^j \sim P(a_i^j = j) \propto w_{i-1}^j$ for $i = 1, \ldots, N - 1$.
6. Draw $x_i^j \sim q_i(x_i | x_{i-1}^j, y_i)$ for $i = 1, \ldots, N - 1$.
7. Set $x_i^N = x_1[k]$.
8. Draw $a_i^N \sim P(a_i^N = j) \propto w_{i-1}^N f(x_i^N | x_{i-1}^j)$.
9. Set $x_{1:t} = \{x_{i-1}^j, x_i^j\}$ for $i = 1, \ldots, N$.
10. Set $w_t^i = W(x_{i-1}^j)$ for $i = 1, \ldots, N$.
11. End for
12. Draw $J$ with $P(i = J) \propto w_T^j$ and set $x_{1:T}[k+1] = x_{1:T}^J$.

3.2 Markov chain Monte Carlo

MCMC offers a strategy for sampling from a complicated probability distribution $\pi$ on the space $\mathcal{Z}$, using an iterative scheme.

A Markov chain on $\mathcal{Z}$ is a sequence of the random variables $\{\zeta[1], \zeta[2], \zeta[3], \ldots\}$, $\zeta[k] \in \mathcal{Z}$. The chain is defined by a kernel $K$, stochastically mapping one element $\zeta[k]$ onto another element $\zeta[k + 1]$. That is, the distribution of the random variable $\zeta[k]$ depends on the previous element as $\zeta[k + 1] \sim K(\zeta[k])$.

If the kernel $K$ is ergodic with a unique stationary distribution $\pi$, the marginal distribution of the chain will approach $\pi$ in the limit. Let $\zeta[0]$ be an arbitrary initial state with $\pi(\zeta[0]) > 0$, then by the ergodic theorem (Robert and Casella, 2004):

$$\frac{1}{K} \sum_{k=1}^{K} h(\zeta[k]) \rightarrow \mathbb{E}_{\pi}[h(\zeta)],$$

as $K \rightarrow \infty$ for any function $h : \{\mathbb{R}^n\}^T \mapsto \mathbb{R}$, with $\mathbb{E}_{\pi}[\cdot]$ denoting expectation w.r.t. $\zeta$ under the distribution $\pi$.

That is, for sufficient large $K$, the realization of $\{\zeta[k], \zeta[k + 1], \ldots\}$ are (possibly correlated) samples from $\pi$. This summarizes the key idea in MCMC: if $\pi$ is of interest, construct an appropriate kernel $K$ with stationary distribution $\pi$ and simulate a Markov chain to obtain samples of $\pi$.

Note that any finite realization of the chain $\{\zeta[1], \ldots, \zeta[K]\}$ may be arbitrarily bad approximation of $\pi$. This typically depends on the initialization $\zeta[0]$ and how well the kernel $K$ manages to explore $\mathcal{Z}$, referred to as the mixing.

3.3 Smoothing using MCMC

Take the general space $\mathcal{Z}$ as the more concrete space $\{\mathbb{R}^n\}^T$ (where $x_{1:T}$ lives). Note that CPF-AS in Algorithm 2 maps one element in $\{\mathbb{R}^n\}^T$ onto another element in $\{\mathbb{R}^n\}^T$, and can therefore be interpreted as an MCMC kernel. The unique stationary distribution for CPF-AS is $p(x_{1:T}|y_{1:T})$ (which is far from obvious, but shown by Lindsten et al. (2014)). Now, by constructing a Markov chain, Algorithm 3 is obtained, generating samples from the distribution $p(x_{1:T}|y_{1:T})$ (i.e., a smoother).
Algorithm 3 MCMC smoother

Input: \(x_{1:T}[0]\) (Initial (arbitrary) state trajectory)
Output: \(x_{1:T}[1], \ldots, x_{1:T}[K]\) (\(K\) samples from the Markov chain)

1. for \(k = 1, \ldots, K\) do
2. Run the CPF-AS (Algorithm 2) conditional on \(x_{1:T}[k-1]\) to obtain \(x_{1:T}[k]\).
3. end for

An illustration of Algorithm 3 was provided already in Figure 1; the initial trajectory is obviously not a sample from \(\pi = p(x_{1:T} | y_{1:T})\). Artefacts from the initializations appear to be present also in iteration \([1], [2]\), and possibly \([3]\). However, iterations \([5], [6], [7]\) appear to be (correlated) samples from the distribution \(\pi\), which is what was sought.

3.4 Convergence

The convergence analysis of Algorithm 3 can, similar to the FFBSi in Section 2.2, be posed as the question of how well \(h(x_{1:T})\) can be approximated by \(h^*_{\text{CPF-AS}} = \frac{1}{K} \sum_{k=1}^{K} h(x_{1:T}[k])\), where \(x_{1:T}[k]\) comes from Algorithm 3. Before stating the theorem, let us make the following two rather technical assumptions.

A1. The proposal \(q\) is designed such that for any \(x_{t-1}\) with non-zero probability (given the measurements \(y_{1:t-1}\)), any \(x_t\) with non-zero probability (given \(y_{1:t}\)) should be contained in the support of \(q\).

A2. There exists a constant \(\kappa < \infty\) such that \(|W| < \kappa\).

Theorem 1. (Convergence for Algorithm 3). Under the assumptions A1 and A2, for any number of particles \(N > 1\), and for any bounded function \(h: \{\mathbb{R}^p\}^T \rightarrow \mathbb{R}\), there exists a \(\epsilon_0 < \infty\) such that

\[
\sqrt{K} \left( h^*_{\text{CPF-AS}} - E[h(x_{1:T}) | y_{1:T}] \right) \leq \epsilon_0 \tag{8}
\]

converges weakly to \(\mathcal{N}(0, \sigma^2_k)\).

Proof. The CPF-AS is uniformly ergodic for \(N > 1\), (Lindsten et al., 2014, Theorem 3). Therefore (Liang et al., 2010, Theorem 1.5.4) is applicable.

Note 1. By Theorem 1 the convergence of Algorithm 3 to the smoothing distribution not dependent of the number of particles \(N \rightarrow \infty\), but is only relying on the number of iterations \(K \rightarrow \infty\).

3.5 Computational complexity

The computational complexity of Algorithm 3 is of order \(O(KN)\), where \(N\) is the number of particles in the CPF-AS and \(K\) the number of iterations. However, in some programming languages, e.g., Matlab, vectorized implementations are preferable. The sequential nature of Algorithm 3 in \(k\) does not allow such a vectorized implementation, which is a clear drawback. On the other hand, \(K\) does not have to be specified a priori, but Algorithm 3 can be run repeatedly until satisfactory results are obtained, or a given computational time limit is violated.

The short message here is that the convergence rate is \(\sqrt{K}\), obtained to the computational cost of \(O(K)\) (for a fixed number of particles \(N\)), compared to the convergence rate \(\sqrt{N}\) to the less beneficial cost of \(O(N^2)\) for FFBSi. However, one should remember that the samples obtained from FFBSi are uncorrelated, which is typically not the case for Algorithm 3.

4. SIMULATED EXAMPLES

4.1 Scalar linear Gaussian SSM

As a first example, consider the scalar linear Gaussian SSM

\[
x_{t+1} = 0.2x_t + u_t + w_t, \quad u_t \sim \mathcal{N}(0, 0.3), \quad (9a)
\]

\[
y_t = x_t + e_t, \quad e_t \sim \mathcal{N}(0, 1), \quad (9b)
\]

with \(E[x_1] = 0\) and \(E[x_T^2] = 0.1\). Implementing Algorithm 3 with \(N = 2\) (with \(T = 80\) and \(u\) being low-pass filtered white noise), the result shown in Figure 1 is obtained. As the system is linear and Gaussian, analytical expressions for \(p(x_{1:T} | y_{1:T})\) can be found using the RTS smoother, which is shown in grey in Figure 1.

4.2 Nonlinear, multi-modal example

We will now turn to a more challenging problem, pin-pointing some interesting differences between the forward-backward smoother (FFBSi) and our MCMC-based smoother in Algorithm 3. We will start with a discussion using intuitive arguments, to motivate the example.

The FFBSi smoother handles the path degeneracy problem in the particle filter discussed in Section 2. However, the support for the backward simulation is still limited to the particles sampled by the particle filter. As these particles, for \(t < T\), are sampled from the filtering distribution \(p(x_t | y_{1:T})\) (and not the smoothing distribution \(p(x_t | y_{1:T})\), due to the factorization (4)), only few of the particles may be useful if the difference between the filtering and smoothing distribution is ‘large’. This might cause a problem for the FFBSi smoother, since there might exist cases where the particles do not explore the relevant part of the state space. An interesting question is now if Algorithm 3 can be expected to explore the relevant part of the state space better than the FFBSi smoother?

One way to understand the effect of the conditional trajectory in CPF-AS is as follows: If a proposal distribution \(q_t \neq f\) is used in a regular particle filter (Step 5 in Algorithm 1), it is compensated for in the update of the weights, Step 7 (and (5b), so that \(x^*_t, w^*_t\) for \(i = 1, \ldots, N\) are still an approximation of \(p(x_t, y_{1:T})\), even if \(q_t \neq f\).

The CPF-AS can be thought of as a regular particle filter, but with a ‘proposal’ \(q_t(x_t)\) that deterministically sets \(x_t = x_t[k]\) (Step 7 of Algorithm 2) and ‘artificially’ assigns an ancestor to it (Step 8). However, there is no compensation for this ‘proposal’ in Step 10. Therefore, the samples \(\{x^i_t, w^i_t\}^{N}_{i=1}\) obtained from the CPF-AS can be expected to be biased towards the conditional trajectory \(x_{1:T}[k]\).

On the other hand, we know from Lindsten et al. (2014) that the conditional trajectories in the limit \(k \rightarrow \infty\) are samples of \(p(x_{1:T} | y_{1:T})\). The bias towards \(x_{1:T}[k]\) in the CPF-AS can therefore be thought of as ‘forcing’ the CPF-AS to explore areas of the state space relevant for the smoothing distribution \(p(x_{1:T} | y_{1:T})\) (rather than the filtering distribution \(p(x_{1:T} | y_{1:T})\)) for large \(k\).

A simulated example, appealing to this discussion, is now given. The problem is to sample from the smoothing distribution for a one-dimensional SSM with multi-modal properties of \(g(x_t | y_t)\). The state space model is \(f(x_{t+1} | x_t) = \mathcal{N}(x_t + x_t, \sigma^2)\) and \(g(y_t | x_t)\) is implicitly defined through the surface in Figure 3, where the surface level in point \((x, t)\) defines \(g(y_t | x_t)\), for a given \(y_t\) (not shown).
Given \( x_0 \), finding the maximum a posteriori estimate of the smoothing distribution \( p(x_{1:T} \mid y_{1:T}) \) amounts to finding the path \( x_{1:T} \) maximizing \( p(x_{1:T} \mid y_{1:T}) \propto \prod_{t=1}^{T} f(x_t | x_{t-1}) \prod_{t=1}^{T} g(y_t | x_t) \), where \( g(y_t | x_t) \) is defined through the surface in Figure 3. Intuitively, this can be thought of as going from left (\( t = 0 \)) to right (\( t = 100 \)) in Figure 3, playing the children’s game ‘the floor is hot lava’ with the cost \( f \) of moving sideways.

The mean of the filtering distribution \( p(x_t \mid y_{1:t}) \) for \( t = 1, \ldots, T \) (obtained by Algorithm 1) is shown in Figure 3, together with the mean from two different smoothers; FFBSi (Section 2.2) and Algorithm 3, respectively.

The two smoothing approximations can indeed be expected to approach each other in the limit \( N \to \infty / K \to \infty \). The problem is interesting because the ‘likelihood landscape’ in Figure 3 contains a ‘trap’. The filtering distribution (and hence the particle filter in FFBSi) will follow the right ‘shoulder’ and discover ‘too late’ (the valley at \( t \approx 70 \)) that it ‘should’ have walked along the left. The smoothing distribution, however, walks along the left shoulder earlier, as it ‘knows’ that the valley at the right hand side will come.

To quantify this discussion on how well the particles explore the state space for the two smoothers, the densities of the sampled particles for both smoothers are plotted in Figure 4. This suggest that Algorithm 3 is able to give a better approximation of the smoothing distribution, as a larger proportion of the particles are sampled in a relevant part of the state space.

In this section, the presented algorithm is applied to a real-world sensor fusion problem; indoor positioning using ultra wideband (UWB), gyroscope and accelerometer measurements. We apply the model from Kok et al. (2015), but rather than using the optimization-based approach in that paper, we employ Algorithm 3. Instead of obtaining the Maximum a Posteriori (MAP) estimate as a point (as in Kok et al. (2015)), we will obtain samples from the posterior distribution, which can be used to estimate the MAP, mean, credibility intervals, etc.

### 5.1 Problem setup

We take the problem as presented by Kok et al. (2015), a 9-dimensional non-Gaussian problem. The goal is to estimate the position, velocity and orientation of the sensor board with the UWB transmitter, accelerometer and gyroscope, placed on a human body moving in the room. The UWB transmitter sends out pulses at (unknown) times \( \tau_t \), and the time of arrival at the \( M \) number receivers (indexed by \( m \)) are measured. The setup is calibrated using the algorithm in Kok et al. (2015), making sure that the receiver positions \( r_m \) are known and that their clocks are synchronized.

In the model, the state vector is \( x_t = [p_t^1, p_t^2, p_t^3, v_t^1, v_t^2, v_t^3, q_t^1, q_t^2, q_t^3, q_t^4] \), \( p_t \) is the (3D) position, \( v_t \) the velocity and \( q_t \) the orientation (parametrized using unit quaternions). All in all, this gives a 9-dimensional nonlinear non-Gaussian problem. The SSM is given by

\[
\begin{align}
    p_{t+1}^n &= p_t^n + T_s v_t^n + \frac{T_s^2}{2} a_t^n, \\
    v_{t+1}^n &= v_t^n + T_s a_t^n, \\
    q_{t+1}^n &= q_t^n \odot \exp \left( \frac{T_s}{2} \omega_t \right), \\
    y_{m,t} &= \tau_t + \| r_m^n - p_t^n \|_2 + e_{m,t}
\end{align}
\]

where (10a) – (10c) are the dynamics and (10d) is the measurement equation. The superscripts \( n \) and \( b \) denote coordinate frames (\( n \) is the navigation frame aligned with gravity, and \( b \) is the body frame, aligned with the sensor axes of the accelerometers), \( \odot \) denotes the quaternion product and \( \exp \) denotes the vector exponential (Hol, 2011). \( T_s \) is the time between two data samples from the accelerometer and gyroscope, sampled with 120 Hz. However, the UWB samples are at approximately 10 Hz. Due to the nature of UWB measurements, \( e_{m,t} \) is modelled as

\[
e_{m,t} \sim \begin{cases} (2 - \alpha) N(0, \sigma_e^2), & e_{m,t} < 0, \\
\alpha \text{Cauchy}(0, \gamma), & e_{m,t} \geq 0 \end{cases}
\]

because measurements can only arrive later (and not earlier) in case of multipath and non-line-of-sight propagation. The acceleration \( a_t^n \) is found via accelerometer measurements \( y_{a,t} \), modelled as

\[
y_{a,t} = R_{n,n}^b (a_t^n - \mu) + \delta_a + e_{a,t}
\]

with \( \mu \) denoting gravity, \( R_{n,n}^b \) is a rotation matrix representation of the state \( e_t^n \), and \( R_{n,n}^b = (R_{n,n}^b)^T \). The angular velocity \( \omega_t \) is obtained from the gyroscope measurements \( y_{\omega,t} \) as

\[
y_{\omega,t} = \omega_t + \delta_\omega + e_{\omega,t}
\]

The noise \( e_{a,t} \) and \( e_{\omega,t} \) are modelled as \( N(0, \sigma_a^2) \) and \( N(0, \sigma_\omega^2) \), respectively. \( \delta_a \) and \( \delta_\omega \) are sensor biases. Note that the accelerometer and gyroscope measurements are
Algorithm 3 was applied to data presented by Kok et al. (2015) with (10) – (13). The details on the implementation are available in the technical report Svensson et al. (2015). The results for K = 1000 iterations and N = 500 particles are summarized in Figure 5 in terms of the mean and credibility intervals (cf. Figure 13 and 14 in Kok et al. (2015)). For reference, the ground truth (obtained by an optical reference system) is also shown in the plot. In terms of computational load, the presented results took about 1 day to obtain on a standard desktop computer.

Note the credibility intervals, which are the gain of using this method producing samples (as opposed to a method based on point estimates). The credibility intervals are varying over time and are different for different states, which indeed adds information to the results.

6. CONCLUSIONS

We have shown how the recently proposed PGAS algorithm can be used to solve the nonlinear state smoothing problem in a fundamentally different way as compared to the currently available most competing particle smoothers. The asymptotical convergence of our smoother was also established. We also illustrated the use of the smoother on two academic examples and one challenging real-world application.

Based on the results of Theorem 1 and the numerical examples we conclude that the MCMC-based smoother in Algorithm 3 is indeed a competitive alternative to the existing state-of-the-art smoothers. The present development opens up for interesting future work. It would for example most likely be quite interesting to consider hybrid versions of FFBSI and Algorithm 3, where FFBSI is used to initialise Algorithm 3. Also further studies on how to tackle the trade-off between the number of particles N and the number of iterations K in Algorithm 3 for optimal performance (given a computational limit) would probably be interesting.

REFERENCES

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

Briers, M., Doucet, A., and Maskell, S. (2010). Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1), 61–89.

Douc, R., Garivier, A., Moulines, E., and Olsson, J. (2011). Sequential Monte Carlo smoothing for general state space hidden Markov models. *The Annals of Applied Probability*, 21(6), 2109–2145.

Doucet, A., and Johansen, A.M. (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky (eds.), *Nonlinear Filtering Handbook*, 656–704. Oxford University Press, Oxford.

Gordon, N.J., Salmond, D.J., and Smith, A.F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, 107–113.

Hol, J.D. (2011). *Sensor Fusion and Calibration of Inertial Sensors, Vision, Ultra-Wideband and GPS*. Ph.D. thesis, Linköping University, Sweden.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1), 1–25.

Kok, M., Hol, J.D., and Schön, T.B. (2015). Indoor positioning using ultra-wideband and inertial measurements. *IEEE Transactions on Vehicular Technology*, forthcoming.

Liang, F., Liu, C., and Carroll, R. (2010). *Advanced Markov Chain Monte Carlo Methods Learning from Past Samples*. John Wiley & Sons, Ltd., Hoboken.

Lindsten, F., Jordan, M.I., and Schön, T.B. (2014). Particle Gibbs with Ancestor Sampling. *Journal of Machine Learning Research*, 15, 2145–2184.

Lindsten, F. and Schön, T.B. (2013). Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends in Machine Learning*, 6(1), 1–143.

Pillonetti, G. and Bell, B.M. (2008). Optimal smoothing of non-linear dynamic systems via Monte Carlo Markov chains. *Automatica*, 44(7), 1676–1685.

Rauch, H.E., Striebel, C., and Tung, F. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8), 1445–1450.

Robert, C.P. and Casella, G. (2004). *Monte Carlo statistical methods*. Springer, New York, 2. ed. edition.

Särkkä, S. (2008). Unscented Rauch–Tung–Striebel smoother. *IEEE Transactions on Automatic Control*, 53(3), 845–849.

Schmidt, S.F. (1966). Application of state-space methods to navigation problems. *Advances in Control Systems*, 3, 293–340.

Schön, T., Gustafsson, F., and Nordlund, P.J. (2005). Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7), 2279–2289.

Smith, G.L., Schmidt, S.F., and McGee, L.A. (1962). Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. Technical Report TR R-135, NASA.

Svensson, A., Schön, T.B., and Kok, M. (2015). Some details on state space smoothing using the conditional particle filter. Technical Report 2015-xxx, Department of Information Technology, Uppsala University. To appear, preliminary version available at www.it.uu.se/katalog/andsv164/Research/Publications.

Svensson, A., Schön, T.B., and Lindsten, F. (2014). Identification of jump Markov linear models using particle filters. In *Proceedings of the IEEE 53rd Conference on Decision and Control (CDC)*, Los Angeles, CA, 6505–6509.