Journal of Machine Learning Research 15 (2015) 1-48 Submitted 4/00; Published 10/00

A Quantitative Evaluation of Monte Carlo Smoothers

Jerker Nordh JERKER.NORDH@CONTROL.LTH.SE
Department of Automatic Control
Lund University
Box 118
SE-221 00 Lund, Sweden

Jacob Antonsson JACOB.ANTONSSON@CONTROL.LTH.SE
Department of Automatic Control
Lund University
Box 118
SE-221 00 Lund, Sweden
International Group for Data Analysis
Pasteur Institute
Paris, France

Editor:

Abstract

In this paper we compare the performance of several popular methods for particle smoothing to investigate if any of the algorithms can deliver better performance for a given computational complexity. We use four different models for the evaluation, chosen to illuminate the differences between the methods. When comparing the computational cost we avoid the issues of implementation efficiency by instead counting the number of evaluations required for a set of high level primitives that are common for all the algorithms. Our results give some insight into the performance characteristics of the chosen methods, even though no universal best choice can be identified since the cost/performance ratios of the methods depend on the characteristics of the problem to be solved which can be clearly seen in the results.

Keywords: Bayesian smoothing, particle Markov chain Monte Carlo, sequential Monte Carlo, Bayesian inference, state-space models

1. Introduction

We are concerned with the problem of inferring the latent states $x_{1:T} \triangleq \{x_t\}_{t=1}^T$, $x_t \in \mathbb{R}^n$, given a set of observations $y_{1:T} \triangleq \{y_t\}_{t=1}^T$, $y_t \in \mathbb{R}^m$ from a state-space model,

$$x_{t+1} \sim p(x_{t+1}|x_t), \tag{1a}$$
$$y_t \sim p(y_t|x_t), \tag{1b}$$
$$x_1 \sim p(x_1). \tag{1c}$$

Specifically, we want to estimate the *joint smoothing density* (JSD), $p(x_{1:T}|y_{1:T})$ for such a system. An estimate of the JSD can be found by the particle filter, which approximates

©2015 Jerker Nordh and Jacob Antonsson.

the marginal densities $p(x_t|y_{1:T}), t \leq T$ of the JSD by a set of weighted samples, called particles. The particle filter is a sequential Monte Carlo (SMC) algorithm applied to state-space models. Due to the degeneracy property (Doucet and Johansen, 2011) of the particle filter, the amount of unique samples approximating the marginal densities are decreasing with decreasing t; the approximations are said to lack particle diversity. The filter density, $p(x_t|y_{1:t})$, estimates are diverse in the particles however, and they can be used to simulate realizations, usually called backward trajectories, from the JSD. This is the idea of the forward filter backward simulator (FFBSi) algorithm (Godsill et al., 2004). Let N and M be the number of particles and backward trajectories respectively. The FFBSi has $\mathcal{O}(MN)$ complexity (Godsill et al., 2004), which can be prohibitive. Asymptotically in N, the complexity can be reduced to $\mathcal{O}(N)$ by the use of rejection sampling, giving the FFBSi-RS algorithm, proposed by Douc et al. (2011).

Another approach, developed by Dubarry and Douc (2011), among others, is to use the degenerate particle approximations of the JSD in a Metropolis within Gibbs sampler to get univariate realizations from all time points of the smoothed trajectory. In each step of the Gibbs sampler, the state at time t is sampled using Metropolis Hastings (MH) sampling. This algorithm is known as the Metropolis Hastings improved particle smoother (MH-IPS).

A couple of algorithms based on using the degenerate particle filter approximation of the JSD as a pseudo likelihood in a Markov chain Monte Carlo (MCMC) sampler have also been proposed, yielding the particle Gibbs (PG), and the particle Metropolis Hastings algorithms. It was shown by Andrieu et al. (2010) that such samplers do indeed target the JSD. The samplers originally proposed by Andrieu et al. (2010) have been improved by the use of backward simulation by for example Lindsten and Schön (2012); Lindsten et al. (2014); Olsson and Ryden (2011), giving the particle Gibbs with backward simulation (PGBS), particle Gibbs with ancestor sampling (PGAS) and particle Metropolis Hastings (PMMH-BS) algorithms. The particle MCMC methods generally have higher complexity than the aforementioned smoothing algorithms, but it has been proposed (Lindsten and Schön, 2013) that they might be more efficient, and can give better approximations of the JSD for fewer particles.

These algorithms have a lot of subtle differences and none have been shown to clearly perform better than the others for smoothing in general state space models. We here try to investigate the performance of the different types of smoothing methods for some state space models commonly used in the literature. We also include a model whose transition kernel (1a) does not admit a density, a common case in practice. One algorithm from each type of method is chosen to, hopefully, reflect the variety between them. The methods chosen are FFBSi-RS, with an additional improvement of adaptive early stopping, proposed by Taghavi et al. (2013), MH-IPS and PGAS. We compare the algorithms for different parameter choices using a novel approach which is independent of implementation and computer system. The theory of these algorithms will not be detailed, we refer the comprehensive introduction and review made by Lindsten and Schön (2013), and the original articles for further details.

Smoothing in state space models is an important topic in its own right, but it also has many applications. The smoothing distributions are for example a key component in several parameter estimation algorithms and can thus be used to find the posterior distributions of parameters in general state space models, or to learn the whole model itself. For example

? uses PMCMC, and Wills et al. (2013) uses an FFBSi-RS smoother together with an expectation maximization procedure, for identification of Hammerstein-Wiener systems.

2. Theoretical Preliminaries

In this section we give a brief review of the particle filter which is the basis for all the other algorithms. We also briefly discuss the backward kernel for state space models, as both the FFBSi-RS and PGAS uses it to simulate from the JSD.

2.1 The Particle Filter

The particle filter is an SMC sampler targeted at the marginal posterior densities $p(x_t|y_{1:t}), \forall t \in \{1, \ldots, t\}$, which are approximated as discrete distributions,

$$p(x_t|y_{1:t}) \approx \sum_{i=1}^{N} w_t^i \delta\left(x_t - x_t^i\right). \tag{2}$$

The sets of weighted particles targeting the filtering densities are calculated by sequential importance sampling. Given a weighted set of particles $\{w_{t-1}^i, x_{t-1}^i\}_{i=1}^N$ approximating $p(x_{t-1}|y_{1:t-1})$, a set of particles approximating $p(x_t|y_{1:t})$ can be found by sampling from an importance distribution, $q(x_t|x_{t-1}^i, y_t)$, chosen to satisfy the Markov property, and the weights can be calculated as

$$w_t^i \propto w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, y_t)},$$

after which they are normalized to sum to unity. The approach leads to an approximation with a variance that will increase toward infinity for increasing t (Doucet and Johansen, 2011). This will gradually shift all weight to one particle, leading to an effective approximation size of only one sample. To mitigate this a resampling step is introduced when the number of effective particles, measured by the effective sample size statistic $ESS(t) \approx 1/\sum(w_t^i)^2$, described by Doucet and Johansen (2011), is too low. A rule of thumb is to resample if $ESS < 2N/3$, where N as before is the number of particles. The resampling is done by sampling new equally weighted particles from the existing set of particles, where the probability of selecting the i:th particle is w_t^i, giving samples distributed according to the empirical distribution (2). This procedure is encoded in ancestor indices a_t^i that keeps track of which previous sample the particle x_t^i originates from.

The simplest choice of importance distribution, which we use and which is often used in practice, is to use the transition kernel (1a), giving the bootstrap filter outlined in Algorithm 1. Doucet and Johansen (2011) gives more details of the particle filter and SMC for state-space models.

2.2 The Backward Kernel

The key to a lot of the smoothing algorithms for (1), such as FFBSi and PGAS, is the backward kernel,

$$p(x_t|x_{t+1}, y_{1:T}) = \frac{p(x_{t+1}|x_t)p(x_t|y_{1:t})p(x_{t+1}|y_{1:T})}{p(x_{t+1}|y_{1:t})}, t \leq T. \tag{3}$$

Algorithm 1 Bootstrap Particle Filter for (1)

Input: Measurements $\{y_t\}_{t=1}^T$.
Output: Particle system $\{w_t^i, x_t^i\}_{i=1}^N, t \in \{1,\dots,T\}$ and ancestor indices $\{a_t^i\}_{i=1}^N, t \in \{2,\dots,T\}$ approximating the marginal filter- and the degenerate smoothing distributions.

for $i = 1$ to N **do**
  Sample x_1^i from $p(x_1)$.
  Compute $\tilde{w}_1^i = p(y_1|x_1^i)$
end for
Set $w_1^i = \tilde{w}_1^i / (\sum_{j=1}^N \tilde{w}_1^j), \quad \forall i \in \{1,\dots,N\}$.
for $t = 1$ to $T-1$ **do**
  for $i = 1$ to N **do**
    if $ESS(t) < 2N/3$ **then**
      Sample ancestor indices, a_{t+1}^i, with $\mathbb{P}(a_{t+1}^i = j) = w_t^j, j \in \{1,\dots,N\}$.
    else
      $a_{t+1}^i = i$
    end if
    Sample x_{t+1}^i from $p(x_{t+1}|x_t^{a_{t+1}^i})$
    Compute $\tilde{w}_{t+1}^i = p(y_{t+1}|x_{t+1}^i)$
  end for
  Set $w_{t+1}^i = \tilde{w}_{t+1}^i / (\sum_{j=1}^N \tilde{w}_{t+1}^j), \quad \forall i \in \{1,\dots,N\}$.
end for

This expression is dependent on the filter distribution which can be approximated with high particle diversity by the particle filter. Mark particles representing the smoothed distribution with a tilde. Using (2) in (3), an empirical approximation to the backward kernel is given by,

$$p(x_t|\tilde{x}_{t+1}, y_{1:T}) \approx \sum_{i=1}^N w_{t|T}^i \delta\left(x_t - x_t^i\right), \tag{4}$$

with smoothing weights calculated as

$$w_{t|T}^i \propto w_t^i p(\tilde{x}_{t+1}|x_t^i).$$

Smoothed samples can then be drawn using the following recursive sampling scheme; starting at $t = T$ and iterating until $t = 1$, all the time conditioning on the already sampled future states $\tilde{x}_{t+1:T}$,

$$\text{Sample } \tilde{x}_T \text{ with } \mathbb{P}\left(\tilde{x}_T = x_T^i\right) = w_T^i, \tag{5a}$$

$$\text{sample } \tilde{x}_t \text{ with } \mathbb{P}\left(\tilde{x}_t = x_t^i\right) = w_{t|T}^i. \tag{5b}$$

3. Algorithms

We chose three algorithms to represent the diversity of particle smoothing algorithms: FFBSi-RS, MHIPS and PGAS. Here we briefly discuss these algorithms and list them as they are implemented. For details and theoretical results we refer to references given in the introductory section of the paper.

3.1 Forward Filtering Backward Simulation with Rejection Sampling - FFBSi-RS

The FFBSi-RS algorithm uses the recursive scheme (5) to sample smoothed trajectories. However, to avoid evaluating all the smoothing weights (2.2), it samples from the categorical distribution defined by $w^i_{t|T},\ i \in \{1, \ldots, N\}$, using rejection sampling. It uses the filter weights, $w^i_t,\ i \in \{1, \ldots, N\}$, as proposal distribution. When the number of particles is large, this renders the complexity of the algorithm approximately linear in the number of particles (Douc et al., 2011). However, the rejection sampler can get stuck for more improbable realizations of the backward trajectories. We therefore use a hybrid scheme where the rejection sampler is run a certain number of iterations, and if needed the final realizations are drawn by evaluating the smoothing weights explicitly. The stopping rule for the rejection sampler is adaptive and depends on an estimate of the average acceptance probability, as suggested by Taghavi et al. (2013). The algorithm used is outlined in Algorithm 2. The expression for the acceptance probability is derived by Douc et al. (2011) in the original article. Our notation approximately follows that of Lindsten and Schön (2013), $\text{Cat}(\cdot)$ for instance, refers to the categorical distribution. Rejection sampling is treated exhaustively by Robert and Casella (2013).

3.2 Metropolis Hastings Improved Particle Smoother - MH-IPS

We now move over to two algorithms that combine the inference approaches of SMC and MCMC. The first one, the MH-IPS, samples from the trajectory using Gibbs sampling by sampling the state at each time-step t separately, while conditioning on the rest of the trajectory. For a Markovian model like (1), the smoothed states x_t should thus be sampled from

$$p(x_t|x_{1:t-1}, x_{t+1:T}, y_{1:T}) \propto f(x_{t+1}|x_t)g(y_t|x_t)f(x_t|x_{t-1}).$$

This is done by MH sampling. A sample x'_t is drawn from a proposal distribution $q(x_t|x_{t-1}, x_{t+1}, y_{1:T})$ and accepted with probability,

$$1 \wedge \frac{p(x_{t+1}|x'_t)}{p(x_{t+1}|x_t)} \frac{p(y_t|x'_t)}{p(y_t|x_t)} \frac{p(x'_t|x_{t-1})}{p(x_t|x_{t-1})} \frac{q(x_t|x_{t+1}, y_t, x_{t-1})}{q(x'_t|x_{t+1}, y_t, x_{t-1})}, \tag{6}$$

where $\wedge$ is the min-operator. We use the proposal density

$$q(x_t|x_{t-1}, x_{t+1}, y_{1:T}) = p(x_t|x_{t-1}), \tag{7}$$

which simplifies the acceptance probability (6) and gives Algorithm 3. Dubarry and Douc (2011) outlines MH-IPS in greater generality, and with more theoretical details. Robert and Casella (2013) provides more details on MH and Gibbs sampling.

3.3 Particle Gibbs Ancestral Sampling - PGAS

We saw that MH-IPS updates each state in the trajectory univariately, which can lead to poor mixing (Lindsten and Schön, 2013). In contrast, PG samples whole trajectories at each update. The trajectories are sampled from a set of degenerate SMC approximations of the smoothed trajectory. Invariance of the sampler is achieved by doing the SMC sampling

Algorithm 2 FFBSi with rejection sampling and early stopping, for (1). For details of the stop criterion see Taghavi et al. (2013)

Input: Forward filter particle system $\{w_t^i, x_t^i\}_{i=1}^N, t \in \{1, \ldots, T\}$.
Output: M realizations from the JSD.

Sample indices $\{b_T(j)\}_{j=1}^M$ from Cat $\left(\{w_T^i\}_{i=1}^N\right)$
for $j = 1$ to M **do**
  $\tilde{x}_T^j = x_T^{b_T(j)}$
end for
for $t = T-1$ to 1 **do**
  $L \leftarrow \{1, \ldots, M\}$
  $\rho_t = \max_i \operatorname{argmax}_{x_{t+1}} p(x_{t+1}|x_t^i)$
  while length$(L)\neg 0$ or stop criterion not fulfilled **do**
    $n \leftarrow$ length(L)
    $\delta \leftarrow \emptyset$
    Sample $\{I(k)\}_{k=1}^n$ from Cat $\left(\{w_t^i\}_{i=1}^N\right)$
    Sample $\{U(k)\}_{k=1}^n$ from $\mathcal{U}([0,1])$
    for $k = 1$ **to** n **do**
      if $U(k) \leq p(\tilde{x}_{t+1}^{L(k)}|x_t^{I(k)})/\rho_t$ **then**
        $b_t(L(k)) \leftarrow I(k)$
        $\delta \leftarrow \delta \cup \{L(k)\}$
      end if
    end for
    $L \leftarrow L \setminus \delta$
  end while
  for $j = 1$ **to** length(L) **do**
    Compute $\tilde{w}_{t|T}^{i,j} \propto w_t^i p(\tilde{x}_{t+1}^j|x_t^i),\ i \in \{1, \ldots, N\}$.
    $w_{t|T}^{i,j} = \tilde{w}_{t|T}^{i,j}/(\sum_i \tilde{w}_{t|T}^{i,j}),\ i \in \{1, \ldots, N\}$.
    Sample $b_t(L(j))$ from Cat $\left(\{w_{t|T}^{i,j}\}_{i=1}^N\right)$
  end for
  for $j = 1$ **to** M **do**
    Set $\tilde{x}_t^j = x_t^{b_t(j)}$ and $\tilde{x}_{t:T}^j = \{\tilde{x}_t^j, \tilde{x}_{t+1:T}^j\}$.
  end for
end for

conditioned on a fixed trajectory. Due to the degeneracy of the trajectories from the SMC pass, the PG sampler can also suffer from bad mixing (Lindsten et al., 2014). In PGAS this is alleviated by splitting the reference trajectory at each time point t, by assigning a new history to the future trajectory. This is done by sampling a new ancestor index, a_t^i, with probability given by the smoothing weights (2.2). This keeps the invariance of the PG sampler and improves mixing. The algorithm we use, with the same choices for proposal density as in Algorithm 1, is listed in Algorithm 4. Further details and theory is given by Lindsten et al. (2014). A related sampler is the Particle Gibbs Backward Simulator (PGBS) developed by Lindsten and Schön (2012, 2013).

Algorithm 3 MHIPS with MH proposal (7) for model (1), performing R iterations of the Gibbs sampler

Input: Degenerate smoothing trajectories $\{w_T^i, x_{1:T}^i\}_{i=1}^N$.
Output: Improved smoothing trajectories $\{\tilde{x}_{1:T}^i\}_{i=1}^M$.

Initialize $\{\tilde{x}_{1:T}^j\}_{j=1}^M$ by sampling from the ancestral paths of the forward filter, drawing each trajectory with probability w_T^i.
for $r = 1$ to R **do**
  for $j = 1$ to M **do**
    Modifed acceptance probability for last time-step
    Sample ${x'}_T^j \sim p(x_T|\tilde{x}_{T-1})$
    With probability $1 \wedge \frac{p(y_T|x'_T)}{p(y_T|\tilde{x}_T)}$, set $\tilde{x}_T^j = {x'}_T^j$
    for $t = T-1$ to 2 **do**
      Sample ${x'}_t^j \sim p(x_t|\tilde{x}_{t-1})$
      With probability $1 \wedge \frac{p(\tilde{x}_{t+1}|x'_t)}{p(\tilde{x}_{t+1}|\tilde{x}_t)} \frac{p(y_t|x'_t)}{p(y_t|\tilde{x}_t)}$, set $\tilde{x}_t^j = {x'}_t^j$
    end for
    Modifed proposal distribution for first time-step
    Sample ${x'}_1^j \sim p(x_1)$
    With probability $1 \wedge \frac{p(\tilde{x}_2|x'_1)}{p(\tilde{x}_2|\tilde{x}_1)} \frac{p(y_1|x'_1)}{p(y_1|\tilde{x}_1)}$, set $\tilde{x}_1^j = {x'}_1^j$
  end for
end for

Algorithm 4 PGAS kernel for (1)

Input: Measurements and conditional trajectory $\{y_t, x'_t\}_{t=1}^T$
Output: Smoothed trajectories $\{\tilde{x}_{1:T}^i\}_{i=1}^M$ and intermediate quantities such as the same particle systems given by the particle filter.

for $i = 1$ to $N-1$ **do**
  Sample $\tilde{x}_1^i$ from $p(x_1)$.
  Compute $\tilde{w}_1^{(i)} = p(y_1|\tilde{x}_1^i)$
  Set $\tilde{x}_1^N = x'_1$.
end for
Set $w_1^i = \tilde{w}_1^i / (\sum_{j=1}^N \tilde{w}_1^j), \quad \forall i \in [1, N]$.
for $t = 1$ to $T-1$ **do**
  for $i = 1$ to $N-1$ **do**
    Sample ancestor indices, a_{t+1}^i, with $\mathbb{P}(a_{t+1}^i = j) = w_t^j, j \in [1, N]$.
    Sample $\tilde{x}_{t+1}^i$ from $p(x_{t+1}|\tilde{x}_t^{a_{t+1}^i})$
  end for
  Set $\tilde{x}_{t+1}^N = x'_{t+1}$.
  Sample a_{t+1}^N with $\mathbb{P}\left(a_{t+1}^N = i\right) = \frac{w_t^i p(x'_{t+1} \mid \tilde{x}_t^i)}{\sum_{l=1}^N w_t^l p(x'_{t+1} \mid \tilde{x}_t^l)}$
  for $i = 1$ to N **do**
    Set $\tilde{x}_{1:t+1}^i = \{\tilde{x}_{1:t}^{a_t^i}, \tilde{x}_{t+1}^i\}$
    Compute $w_{t+1}^i = p(y_{t+1}|\tilde{x}_{t+1}^i)$
  end for
end for

4. Models

We chose four different models to evaluate the smoothing algorithms. We use a standard one-dimensional nonlinear model that is commonly seen used in the particle filtering and

smoothing literature for benchmarking algorithms. A model often used in two-dimensional tracking applications is also included to test how the algorithms fare in a higher dimensional scenario. Especially interesting in that regard is the FFBSi-RS algorithm, since rejection sampling notoriously gets trickier for higher dimensions.

It is not uncommon for models used in engineering to have degenerate transition kernels. This is for example true for orientation filter models, used for finding the orientation of a physical body using inertial measurement sensors, and other types of dynamical first principles models (Gustafsson, 2010). Such models impose a further difficulty for the smoothing algorithms, and are very common in practice. As an example of such a model we therefore include a double integrator. It is included both in its degenerate formulation and as a non-Markovian model, that is the result of marginalizing over the deterministic state. The following subsections state the models in some more detail.

4.1 Standard Benchmark Model

The model

$$\begin{aligned} x_{t+1} &= 0.5x_t + 25\frac{x_t}{1+x_t^2} + 8\cos 1.2t + w_t, && \text{(8a)} \\ y_t &= 0.05x_t^2 + e_t, \quad x_1 \sim \mathcal{N}(0,5), && \text{(8b)} \\ w_t &\sim \mathcal{N}(0,10), \quad e_t \sim \mathcal{N}(0,1), && \text{(8c)} \end{aligned}$$

where $\mathcal{N}(m,P)$ is the Gaussian distribution with mean m and covariance P, is commonly used for evaluating filter and smoother performance. It has been used by for example Lindsten and Schön (2013); Arulampalam et al. (2002); Godsill et al. (2004) as a benchmark model. For such a model the filtering and smoothing distributions are multi-modal, which makes algorithms that assume a fixed distribution on the posterior, such as the extended Kalman filter (Särkkä, 2013), perform very poorly.

4.2 Double Integrator

An example of a state space model with a degenerate transition kernel is the double integrator

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w_t, && \text{(9a)} \\ y_t &= \begin{pmatrix} 1 & 0 \end{pmatrix} x_t + e_t, && \text{(9b)} \\ w_t &\sim \mathcal{N}(0,Q), \quad e_t \sim \mathcal{N}(0,R). && \text{(9c)} \end{aligned}$$

Due to the noise only acting on the input there is a deterministic relation between the two states making the model degenerate and not suitable for the standard particle smoothing methods. This coupling means that $p(x_{t+1}^i | x_t^j) = 0,\ \forall j \neq a_t^i$, where a_t^i is the index of the ancestor for particle x_{t+1}^i.

The model in (9) can be rewritten as a first order system with a non-Markovian structure. For notational brevity introduce the notation $x_t = (p_t \quad v_t)^T$. The model can then be

rewritten as

$$\begin{aligned} v_{t+1} &= v_t + w_t, && \text{(10a)} \\ y_t &= p_1 + \sum_{i=1}^{t-1} v_i + e_t, && \text{(10b)} \\ w_t &\sim \mathcal{N}(0, Q), \quad e_t \sim \mathcal{N}(0, R), && \text{(10c)} \end{aligned}$$

and the smoothing problem can be solved using a non-Markovian particle smoother (Lindsten and Schön, 2013). For this particular model it is possible to reduce the computational effort by propagating additional information in the forward and backward steps of the algorithms. Writing the model (10) as

$$\begin{aligned} v_{t+1} &= v_t + w_t, && \text{(11a)} \\ s_{t+1} &= s_t + v_t, && \text{(11b)} \\ y_t &= s_t + e_t, && \text{(11c)} \\ s_0 &= p_0, && \text{(11d)} \\ w_t &\sim \mathcal{N}(0, Q), \quad e_t \sim \mathcal{N}(0, R), && \text{(11e)} \end{aligned}$$

it is clear that during the filtering step, each particle also stores the sum of all its previous states. At a quick glance this looks like simply reintroducing the p-state from the original model, but the key distinction is that this new variable is a function of the past trajectory, and not included as a state in the model.

The smoothing weights for the model (11) are computed using the density

$$\begin{aligned} &\prod_{k=t+1}^{T} p(y_k|v_k)p(v_k|v_{1:k-1}, y_{1:k-1}) && \text{(12)} \\ &\qquad \underset{v_{1:t}}{\propto} p(y_{t+1:T}|v_{1:T})p(v_{t+1}|v_t), && \text{(13)} \end{aligned}$$

as noted by (Lindsten and Schön, 2013). Evaluating this directly leads to a computational effort for each time-step that grows with the length of the full dataset. This is clearly undesirable, but using the same approach as Lindsten and Schön (2013), and noticing that (13) only needs to be evaluated up to proportionality (with regard to $v_{1:t}$) it is possible to propagate information backwards during the smoothing in the same way as the s_t variables propagate the sum during filtering. The first factor of (13) can be evaluated up to proportionality as follows,

$$\begin{aligned} p(y_{t+1:T}|s_t, v_{t:T}) &= \prod_{k=t+1}^{T} p(y_k|s_t, v_{t:T}) && \text{(14a)} \\ &\underset{s_t, v_t}{\propto} \prod_{k=t+1}^{T} e^{(s_t+v_t)^2 - 2(y_k - \sum_{j=t+1}^{k-1} v_j)(s_t+v_t)} && \text{(14b)} \\ &= e^{(T-t)(s_t+v_t)^2 - 2\sum_{k=t+1}^{T}(y_k - \sum_{j=t+1}^{k-1} v_j)(s_t+v_t)}. && \text{(14c)} \end{aligned}$$

Through the introduction of two new variables N_t, γ_t that are propagated backwards during the smoothing this allows (14) to be evaluated as

$$\log p(y_{t+1:T}|s_t, v_t, v_{t+1:T}) + \text{constant} = \frac{1}{2R}(N_{t+1}(s_t + v_t)^2 - 2\gamma_{t+1}(s_t + v_t)), \tag{15a}$$

$$N_t = N_{t+1} + 1, \quad N_T = 1, \tag{15b}$$

$$\gamma_t = \gamma_{t+1} + y_t - N_{t+1}v_t, \quad \gamma_T = y_T. \tag{15c}$$

Using (15) it is now possible to evaluate the required smoothing density in constant time. A more detailed derivation of these expressions for this particular model is given by Nordh (2015, Submitted).

4.3 Tracking Model

To test how the algorithms fare in a higher dimensional setting we have included a four dimensional model commonly used for tracking. The model was also used to test a new FFBSi smoother, similar to MHIPS, in Bunch and Godsill (2013). Define the state vector of positions and velocities in two dimensions, $x_t = \begin{pmatrix} x_t & y_t & \dot{x}_t & \dot{y}_t \end{pmatrix}$, let $h = 0.1$ be the sampling time, I_2 the identity matrix in $\mathbb{R}^{2\times 2}$, $0_{2\times 2}$ the null matrix in $\mathbb{R}^{2\times 2}$, and 0_n the null vector in $\mathbb{R}^n$ respectively. The model is then given by,

$$x_{t+1} = \begin{pmatrix} I_2 & hI_2 \\ 0_{2\times 2} & I_2 \end{pmatrix} x_t + w_t, \tag{16a}$$

$$y_t = \begin{pmatrix} \text{atan2}(y_t, x_t) \\ \sqrt{x_t^2 + y_t^2} \end{pmatrix} + e_t, \tag{16b}$$

$$w_t \sim \mathcal{N}\left(0_4, \begin{pmatrix} \frac{h^3}{3} I_2 & \frac{h^2}{3} I_2 \\ \frac{h^2}{3} I_2 & hI_2 \end{pmatrix}\right), \tag{16c}$$

$$e_t \sim \mathcal{N}\left(0_2, \begin{pmatrix} \left(\frac{\pi}{720}\right)^2 & 0 \\ 0 & 0.1 \end{pmatrix}\right), \tag{16d}$$

where $\text{atan2}(\cdot, \cdot)$ is the four-quadrant inverse tangent.

5. Method

To compare the smoothing performance of the algorithms we use the Root Mean Square Error (RMSE) metric which is a standard choice in the literature. It is however flawed in that it only compares the estimated mean value to the true value, whereas the goal of a particle smoother is to provide a good approximation of the JSD. The fact that the average of the posterior density most of the time will not coincide with the true state means that there is a problem specific lower bound for the RMSE. This is related to the Cramér-Rao Lower Bound (CRLB)(Tichavsky et al., 1998) which provides a bound on the achievable performance of any estimator for a given problem. In theory all the methods examined in this paper should reach the same RMSE since they all converge to the true posterior

distribution as the quality of the approximations are increased. To reduce the variance of the measured RMSE metric we compute the average over a large number of realizations from each model. To ensure a fair comparison for the different algorithms they are all given the exact same set of realizations. Denote, as before, the number of realizations as N and the length of each realization as T, the true value as x and the estimated mean value as $\bar{x}$. The average RMSE is then calculated as

$$\frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{T}\sum_{t=1}^{T}(\bar{x}_t - x_t)^2}.$$

The standard error of the average RMSE is also computed to determine if the observed differences between the methods are statistically significant.

Our goal is not only to investigate which methods can provide the lowest RMSE for a given problem, but also to analyze the computation effort required to do so. To avoid any issues arising from the efficiency of our implementation, the execution time is not used as the metric of computation effort. The methods are all based on a few common operations. We therefore use the number of times these operations are performed by the algorithms as a complexity measure. The high-level primitives are listed in Table 1. In the presented graphs all the operations are equally weighted to give a quick overview, in practice the relative cost of the operations are both problem and implementation specific. The level of parallelization that can be exploited, both in the hardware and software, and which simplifying assumptions that can be made about the model will greatly affect the cost of each operation. Therefore we chose to not include the actual execution times for the algorithms as that would be more a benchmark of our implementation than of the methods themselves. This is especially true in our case since we used a flexible generic framework to test the methods. This facilitates implementation to a great degree, but causes an overhead in terms of execution time, since the framework does not exploit model specific characteristics to increase speed, such as knowledge about which matrices are time-invariant, sparsity structures and so on. As a drastic example consider an embedded system where for performance reasons it is decided to sample from distributions by simply iterating over a pre-calculated array with numbers. The relative cost of the operations in that scenario will differ drastically compared to an implementation running on a desktop system and where accuracy and correctness might be more important than execution time. The interested reader can download the raw results together with the source code from the homepage of Nordh and Antonsson (2013) and make their own analysis for their particular target platform.

6. Results

The mean of the RMSE together with a band of 2 standard errors on both sides is plotted as a function of computation effort, encoded in numer of high-level primitive operations, for all the methods and a few different parameter choices are presented in Figure 1 for the standard nonlinear model, Figure 2 for the tracking example, Figure 3 for the degenerate double integrator and in Figure 4 for the marginalized double integrator.

For the marginalized double integrator model we use the normal FFBSi-smoother instead of rejection sampling. This is because we can only evaluate the required probability densities

Table 1: List of common high-levels operations for all the algorithms used. The number of times each operation is performed is compared for the different algorithms and used as the computational cost

Sample from $p(x_{t+1}\|x_t)$	Used in e.g the particle filter when propagating the estimates forward
Evaluate $p(y_t\|x_t)$	Used in e.g the particle filter when updating the particle weights
Evaluate $p(x_{t+1}\|x_t)$	Used in most of the smoothers to update the weights of particles given information about the future trajectories
Compute $\underset{x_{t+1}}{\text{argmax}}\ p(x_{t+1}\|x_t)$	Used for rejections sampling in combination with FFBSi smoothers
Sample from $p(x_1)$	Draw particles from the initial distribution

up to proportionality and we can therefore not compute the required normalization factor. Additionally, in theory the rejection sampling would work very poorly since it is actually sampling the full ancestral path in each step, $x_{1:t}$, which is a very high-dimensional problem and therefore not suitable for rejection sampling.

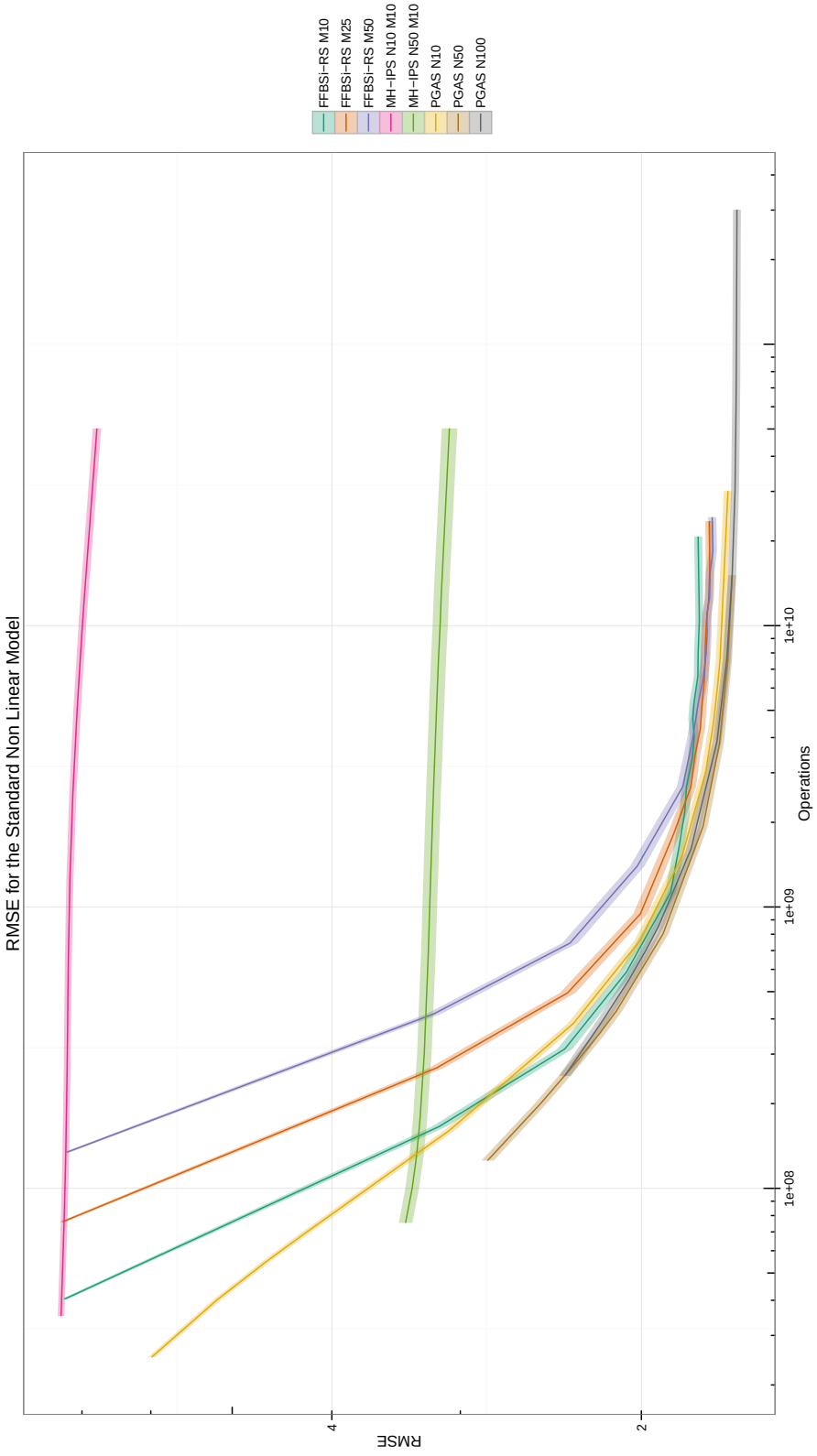

Figure 1: RMSE as a function of the computation effort for all methods applied to the standard nonlinear model (8). The mean of the RMSE is plotted together with a band of two standard errors on both sides. For the FFBSi-RS plots, the number of backward trajectories are held fixed at 10, 25 and 50, while the number of particles used in the forward filter is increased. For the two MH-IPS cases the number of particles in the forward filter is either 10 or 50. In both cases there are 10 backward trajectories. The computational complexity is varied by changing the number of performed MH-IPS passes. For the PGAS plots, the number of particles used in the filter is either 10, 50 or 100, and the computational complexity is varied by changing the number of iterations that are performed.

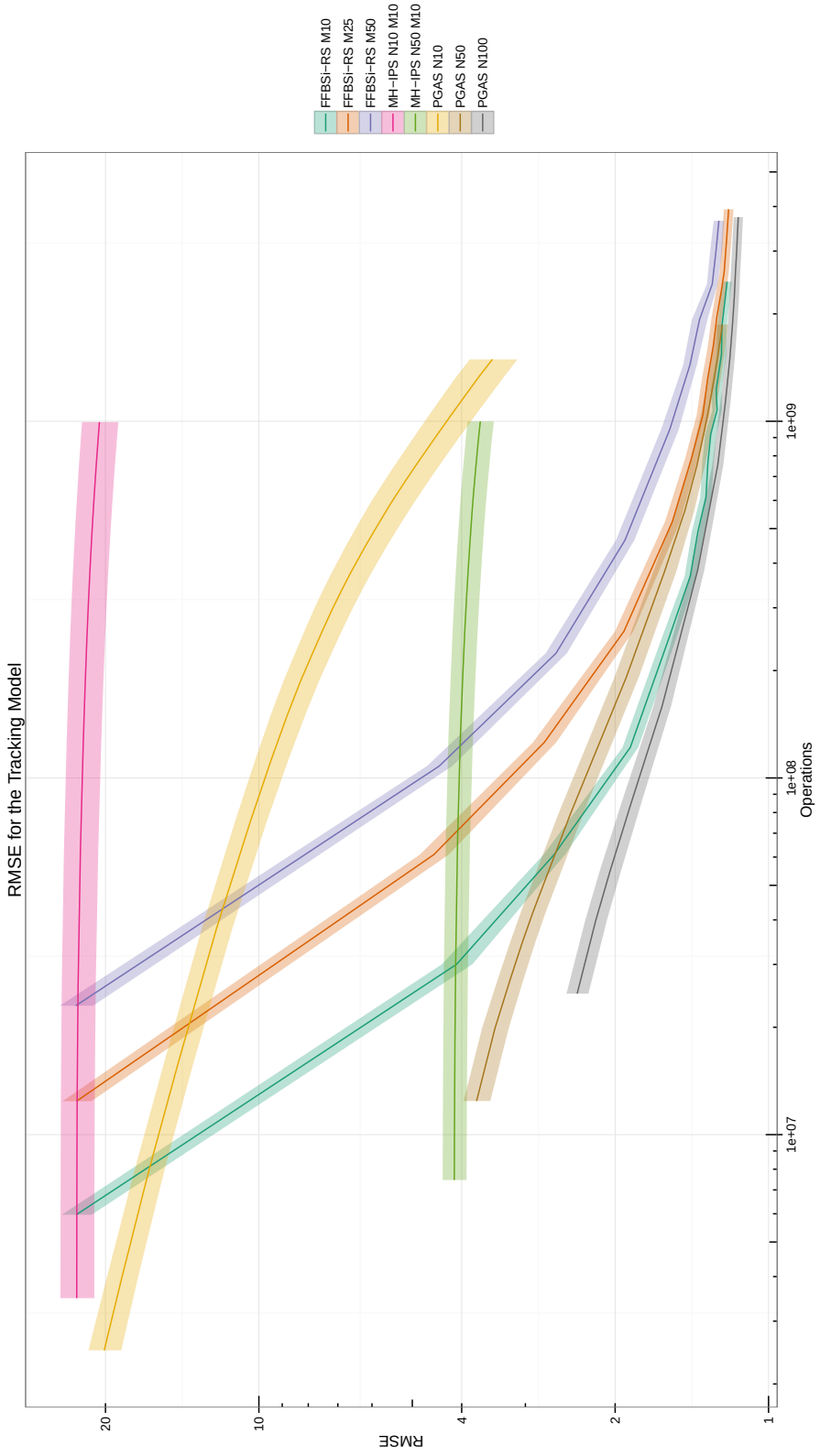

Figure 2: RMSE as a function of the computation effort for all methods applied to the tracking model (16). The notation is the same as in Figure 1. As can be seen, the MH-IPS performs poorly for this model, yielding almost no visible improvements as the number of iterations, R, is increased. In this particular case, PGAS seems to be the clear winner.

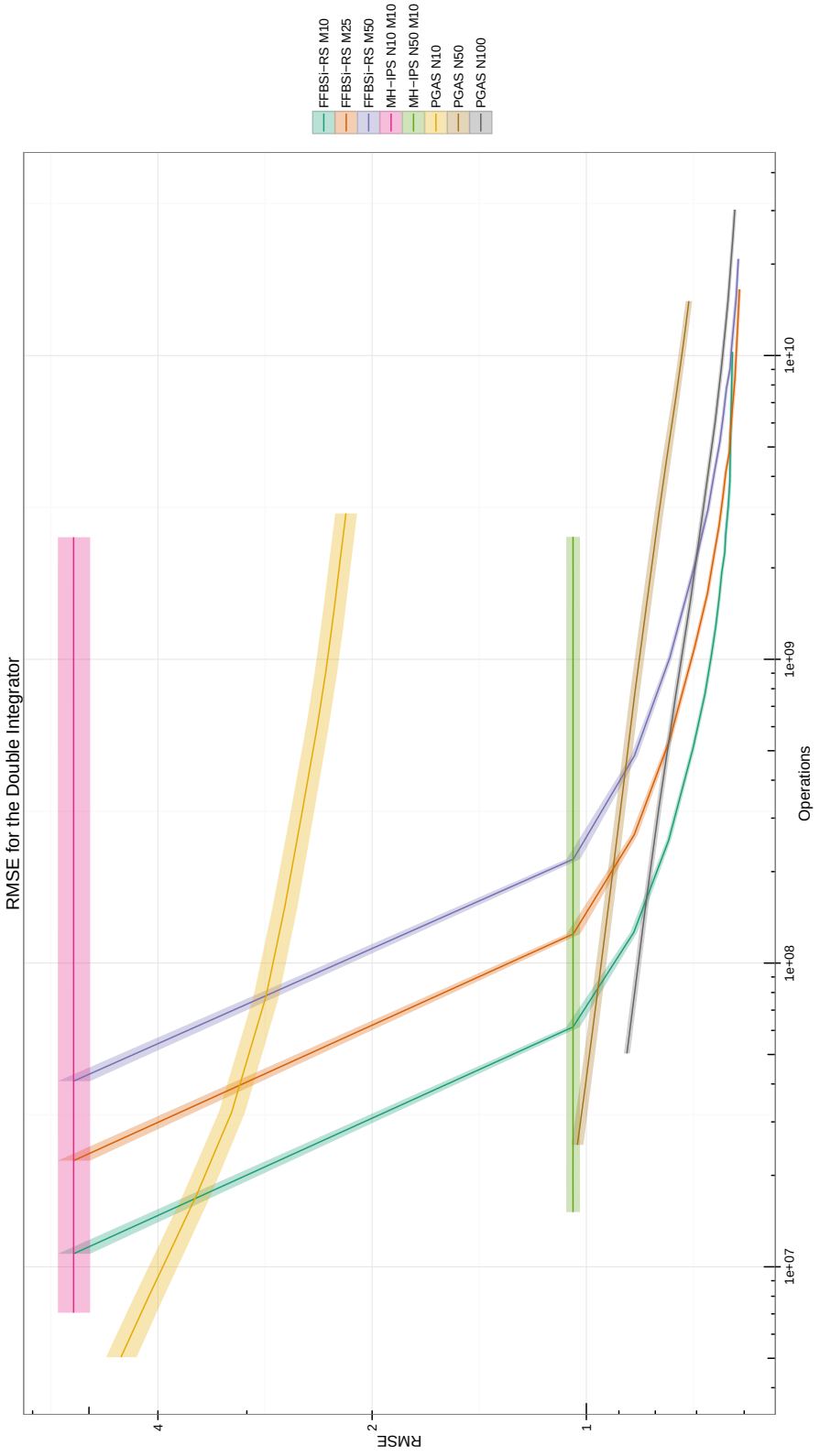

Figure 3: RMSE as a function of the computation effort for all methods applied to the degenerate double integrator model (9). The notation is the same as in Figure 1. Both PGAS and MH-IPS perform poorly for this model, which is not unexpected since they both rely on the backward kernel $p(x_t|x_{t+1})$, which due to the degeneracy of the model will only be non-zero when following the original ancestral path.

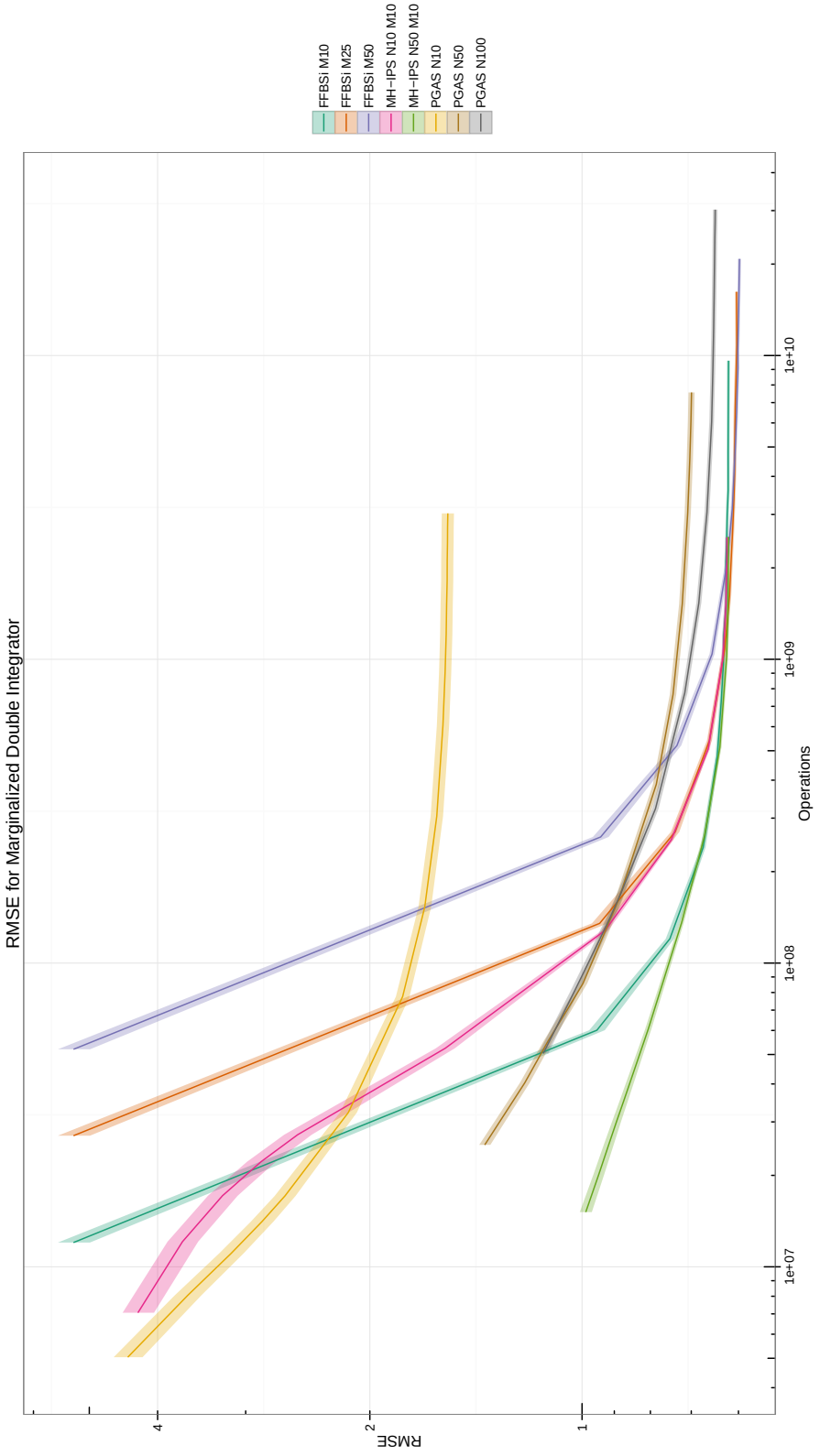

Figure 4: RMSE as a function of the computation effort for all methods applied to the marginalized double integrator model (10). Note that for this model the FFBSi smoother is used without rejection sampling. Other than that, the notation is the same as in Figure 1. For this model the MH-IPS smoother is very competitive, delivering much lower RMSE in the region of low computation effort.

7. Discussion

By studying the figures in Section 6 we see that there is no method that consistently outperforms the others, however some general trends can be observed. For the FFBSi smoothers, using a larger number of backward trajectories only gives an improvement when also increasing the number of forward particles. Thus, unless a large amount of computation effort can be spent, it is typically better to use a large number of forward particles and fewer backward trajectories.

The MH-IPS algorithm shows the most varying results. For the degenerate case it shows no improvement in the RMSE when increasing the number of iterations performed. This is expected since due to the degeneracy it will not be possible to find any new particles that give nonzero probability for $p(x_{t+1}|x_t)$ and $p(x_t|x_{t-1})$. For both the tracking example and the standard nonlinear model we see that the average RMSE only slowly decreases with the number of iterations performed, making MH-IPS a less attractive method for these models. However, for the marginalized double integrator we see that for low effort of computation MH-IPS clearly outperforms the other methods, and it also remains competitive for larger computation efforts. For the results presented in this paper the proposal was chosen as $q(x_t|x_{t-1}, y_t, x_{t+1}) = p(x_t|x_{t-1})$. This a common choice since it is typically already used in the particle filter, however using a proposal taking more information in account might improve the performance of MH-IPS. Finding such a proposal density is a model specific problem and in many cases it might not be easily obtainable.

We can see that PGAS in general performs very well, but for the degenerate model it is outperformed by the FFBSi smoother for all but the lowest computation efforts. Worth noting is that in this case the ancestral sampling part of the PGAS algorithm will not be effective and the extra computation effort required for the AS step is wasted. The end result will therefor be the same as for a regular PG smoother, which is thus preferable in this case.

8. Conclusion

In this paper we have clearly demonstrated that there is no universal best choice of method for performing particle smoothing. Assuming that the goal is to have the lowest possible RMSE for a given number of computations, all the available methods have to be compared for the particular problem. To facilitate easy comparison and experimentation it is useful to use a software that separates the model specific implementation parts from the generic parts of algorithms. One such framework is pyParticleEst(Nordh, 2015) which is what we used for the work presented in this paper.

Acknowledgments

We would like to thank Prof. Thomas Schön, Upsala University, for the suggestions and ideas that led to this work. The authors are members of the LCCC Linnaeus Center and the eLLIIT Excellence Center at Lund University.

References

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

M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.*, 50(2): 174–188, feb 2002. ISSN 1053-587X.

Pete Bunch and Simon Godsill. Improved particle approximations to the joint smoothing distribution using markov chain monte carlo. *Signal Processing, IEEE Transactions on*, 61(4):956–963, 2013.

Randal Douc, Aurélien Garivier, Eric Moulines, Jimmy Olsson, et al. Sequential monte carlo smoothing for general state space hidden markov models. *The Annals of Applied Probability*, 21(6):2109–2145, 2011.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *Handbook of Nonlinear Filtering*. Oxford, UK: Oxford University Press, 2011.

Cyrille Dubarry and Randal Douc. Particle approximation improvement of the joint smoothing distribution with on-the-fly variance estimation. *arXiv preprint arXiv:1107.5524*, 2011.

Simon J Godsill, Arnaud Doucet, and Mike West. Monte carlo smoothing for nonlinear time series. *Journal of the american statistical association*, 99(465), 2004.

Fredrik Gustafsson. *Statistical sensor fusion*. 2010.

Fredrik Lindsten and TB Schön. On the use of backward simulation in the particle gibbs sampler. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 3845–3848. IEEE, 2012.

Fredrik Lindsten and Thomas B. Schön. Backward simulation methods for monte carlo statistical inference. *Foundations and Trends® in Machine Learning*, 6(1):1–143, 2013. ISSN 1935-8237. doi: 10.1561/2200000045. URL `http://dx.doi.org/10.1561/2200000045`.

Fredrik Lindsten, Michael I Jordan, and Thomas B Schön. Particle gibbs with ancestor sampling. *The Journal of Machine Learning Research*, 15(1):2145–2184, 2014.

Jerker Nordh. pyParticleEst: A Python framework for particle based estimation methods. *Journal of Statistical Software*, 2015. In Review.

Jerker Nordh. Metropolis-Hastings Improved Particle Smoother and marginalized models. In *European Conference on Signal Processing 2015*, Nice, France, August 2015, Submitted.

Jerker Nordh and Jacob Antonsson. Raw results and source code for all the experiments in this paper, 2013. URL: http://www.control.lth.se/Staff/JerkerNordh/pscomp.html.

Jimmy Olsson and Tobias Ryden. Rao-blackwellization of particle markov chain monte carlo methods using forward filtering backward sampling. *Signal Processing, IEEE Transactions on*, 59(10):4606–4619, 2011.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

S. Särkkä. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013. ISBN 9781107030657.

Ehsan Taghavi, Fredrik Lindsten, Lennart Svensson, and TB Schön. Adaptive stopping for fast particle smoothing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6293–6297. Ieee, 2013.

P. Tichavsky, C.H. Muravchik, and Arye Nehorai. Posterior cramer-rao bounds for discrete-time nonlinear filtering. *Signal Processing, IEEE Transactions on*, 46(5):1386–1396, May 1998. ISSN 1053-587X. doi: 10.1109/78.668800.

Adrian Wills, Thomas B Schön, Lennart Ljung, and Brett Ninness. Identification of hammerstein–wiener models. *Automatica*, 49(1):70–81, 2013.