Complexity Analysis of the Marginalized Particle Filter

Rickard Karlsson, Thomas Schön, Fredrik Gustafsson

Control & Communication
Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden
WWW: http://www.control.isy.liu.se
E-mail: rickard@isy.liu.se, schon@isy.liu.se
fredrik@isy.liu.se

3th June

Report no.: LiTH-ISY-R-2611
Submitted to

Abstract

In this paper the computational complexity of the marginalized particle filter is analyzed. We introduce an equivalent flop measure to capture floating-point operations as well as other features, which cannot be measured using flops, such as the complexity in generating random numbers and performing the resampling. From the analysis we conclude how to partition the estimation problem in an optimal way for some common target tracking models. Some guidelines on how to increase performance based on the analysis is also given. In an extensive Monte Carlo simulation we study different computational aspects and compare with theoretical results.

Keywords: Nonlinear estimation, Marginalized particle filter, Rao-Blackwellization, Kalman filter, Complexity analysis, Equivalent Flop

Avdelning, Institution
Division, Department

Control & Communication
Department of Electrical Engineering

Datum
Date

2004-06-03

Språk
Language

☐ Svenska/Swedish
[✓ Engelska/English]

☐ ....

Rapporttyp
Report category

☐ Licentiatavhandling
☐ Examensarbete
☐ C-uppsats
☐ D-uppsats
[✓ Övrig rapport]
☐ ....

URL för elektronisk version

`http://www.control.isy.liu.se`

ISBN

ISRN

Serietitel och serienummer
Title of series, numbering

LiTH-ISY-R-2611

ISSN
1400-3902

Titel
Title

Complexity Analysis of the Marginalized Particle Filter

Författare
Author

Rickard Karlsson, Thomas Schön, Fredrik Gustafsson,

Sammanfattning
Abstract

In this paper the computational complexity of the marginalized particle filter is analyzed. We introduce an equivalent flop measure to capture floating-point operations as well as other features, which cannot be measured using flops, such as the complexity in generating random numbers and performing the resampling. From the analysis we conclude how to partition the estimation problem in an optimal way for some common target tracking models. Some guidelines on how to increase performance based on the analysis is also given. In an extensive Monte Carlo simulation we study different computational aspects and compare with theoretical results..

Nyckelord
Keywords

Nonlinear estimation, Marginalized particle filter, Rao-Blackwellization, Kalman filter, Complexity analysis, Equivalent Flop

Complexity Analysis of the Marginalized Particle Filter

Rickard Karlsson, Thomas Schön and Fredrik Gustafsson

***Abstract*— In this paper the computational complexity of the marginalized particle filter is analyzed. We introduce an equivalent flop measure to capture floating-point operations as well as other features, which cannot be measured using flops, such as the complexity in generating random numbers and performing the resampling. From the analysis we conclude how to partition the estimation problem in an optimal way for some common target tracking models. Some guidelines on how to increase performance based on the analysis is also given. In an extensive Monte Carlo simulation we study different computational aspects and compare with theoretical results.**

I. Introduction

In any application of the particle filter to a real world problem the computational complexity of the algorithm is a very important aspect. In this paper the computational complexity issues that arise in the use of the marginalized particle filter (another common name is the Rao-Blackwellized particle filter) are studied. The marginalized particle filter is a clever combination of the standard particle filter by Gordon et al. (1993) and the Kalman filter by Kalman (1960), which can be used when there is a linear substructure, subject to Gaussian noise, available in the model. It is a well known fact that in some cases it is possible to obtain better estimates, i.e., estimates with a smaller variance using the marginalized particle filter instead of using the standard particle filter (Doucet et al., 2001b). Intuitively this makes sense, since then we are using the optimal estimator for the linear states. By now quite a lot has been written about the marginalized particle filter, see e.g., (Doucet, 1998; Doucet et al., 2001a; Chen and Liu, 2000; Andrieu and Doucet, 2002; Andrieu and Godsill, 2000; Schön et al., 2003). However, to the best of the authors knowledge nothing has yet been written about the complexity issues surrounding the marginalized particle filter. In its most general form the marginalized particle filter requires a lot of computations, since we have one Kalman filter associated with each particle. However, there are important special cases, where the computational complexity is much smaller.

We will focus on some important special cases of models common in target tracking. The main objective will be to analyze the computational complexity for different marginalizations. We will exemplify and evaluate the complexity for a radar tracking model, but the discussed method is general and can be applied to a large class of models. The theoretical analysis is compared with simulations and some guidelines on improving the efficiency of the algorithm are given.

II. Bayesian Estimation

Many engineering problems are by nature recursive and require on-line solutions. Common applications such as state estimation, recursive identification and adaptive filtering often require recursive solutions to problems having both nonlinear and non-Gaussian characteristics. Consider the discrete state-space model

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_t), && \text{(1a)}\\ y_t &= h(x_t, e_t), && \text{(1b)} \end{aligned}$$

with state variable x_t, input signal u_t and measurements $\mathbb{Y}_t = \{y_i\}_{i=1}^t$. Furthermore, assume that the probability density functions (pdfs) for the process noise, $p_w(w)$, and measurement noise $p_e(e)$ are known. The nonlinear prediction density $p(x_{t+1}|\mathbb{Y}_t)$ and filtering density $p(x_t|\mathbb{Y}_t)$ for the Bayesian inference is given by (Jazwinski, 1970)

$$p(x_{t+1}|\mathbb{Y}_t) = \int p(x_{t+1}|x_t)p(x_t|\mathbb{Y}_t)dx_t, \tag{2a}$$

$$p(x_t|\mathbb{Y}_t) = \frac{p(y_t|x_t)p(x_t|\mathbb{Y}_{t-1})}{p(y_t|\mathbb{Y}_{t-1})}. \tag{2b}$$

These equations are in general not analytically tractable. However, for the important special case of linear dynamics, linear measurements and Gaussian noise there exist a closed-form solution, given by the Kalman filter (Kalman, 1960). We will write this linear, Gaussian model according to

$$x_{t+1} = A_t x_t + w_t, \tag{3a}$$

$$y_t = C_t x_t + e_t. \tag{3b}$$

In the three subsequent sections we will briefly introduce the Kalman filter, the particle filter and the marginalized particle filter. We will also give references to more detailed treatments of the various filters.

A. The Kalman Filter

The Kalman filter has been the standard estimation technique for many years. Hence, we only briefly give the equations below for notational purposes. For a thorough introduction to the Kalman filter see e.g., (Kailath et al., 2000; Anderson and

Moore, 1979; Gustafsson, 2000)

$$\begin{cases} \hat{x}_{t+1|t} = A_t \hat{x}_{t|t}, \\ P_{t+1|t} = A_t P_{t|t} A_t^T + Q_t, \end{cases} \tag{4a}$$

$$\begin{cases} \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - C_t \hat{x}_{t|t-1}), \\ P_{t|t} = P_{t|t-1} - K_t C_t P_{t|t-1}, \\ K_t = P_{t|t-1} C_t^T (C_t P_{t|t-1} C_t^T + R_t)^{-1}. \end{cases} \tag{4b}$$

The noise covariances are given as

$$Q_t = \text{Cov}[w_t], \tag{5a}$$

$$R_t = \text{Cov}[e_t]. \tag{5b}$$

For a general nonlinear, non-Gaussian system we are forced to use approximate methods, such as the extended Kalman filter. The particle filter, which is briefly introduced in the subsequent section, provides a method to approximate the optimal solution given in (2).

B. The Particle Filter

We will in this section give a brief introduction to the particle filter theory. More complete presentations are provided in for instance (Doucet et al., 2001a; Doucet, 1998; Liu, 2001). The particle filter provides an approximative solution to the optimal discrete time filter given in (2) by updating an approximative description of the posterior filtering density. The particle filter approximates the density $p(x_t|\mathbb{Y}_t)$ by a large set of N samples (particles), $\{x_t^{(i)}\}_{i=1}^N$, where each particle has an assigned relative weight, $q_t^{(i)}$, chosen such that all weights sum to unity. The location and weight of each particle reflect the value of the density in that particular region of the state-space, The particle filter updates the particle location and the corresponding weights recursively with each new observed measurement. For the common special case of additive measurement noise, i.e.,

$$y_t = h(x_t) + e_t, \tag{6}$$

the unnormalized weights are given by

$$q_t^{(i)} = p_e(y_t - h(x_t^{(i)})), \quad i = 1, \ldots, N. \tag{7}$$

The main idea to solve the Bayesian estimation problem is to approximate $p(x_t|\mathbb{Y}_{t-1})$ with

$$p(x_t|\mathbb{Y}_{t-1}) \approx \sum_{i=1}^N q_t^{(i)} \delta(x_t - x_t^{(i)}). \tag{8}$$

Inserting this into (2b) yields a density to sample from. This was the original idea, which was known to diverge, since almost all weights was zero after some iterations. However, the crucial resampling step introduced in (Smith and Gelfand, 1992; Gordon et al., 1993) solved the divergence problems and together with the increased computational power the algorithm has been used in many real-time, on-line estimation applications. The particle filter according to this technique is referred to as the *Sampling Importance Resampling* (SIR), (Gordon et al., 1993), and given in Algorithm 2.1.

Algorithm 2.1 (The Particle Filter):

1) Initialization: For $i = 1, \ldots, N$, initialize the particles, $x_{0|-1}^{(i)} \sim p_{x_0}(x_0)$. Set $t = 0$.
2) For $i = 1, \ldots, N$, evaluate the importance weights $q_t^{(i)} = p(y_t|x_t^{(i)})$ according to the likelihood
$$p(y_t|x_t) = p_e(y_t - h(x_t^{(i)})) \tag{9}$$
and normalize $\tilde{q}_t^{(i)} = \frac{q_t^{(i)}}{\sum_{j=1}^N q_t^{(j)}}$.
3) Measurement update: Resample N particles with replacement according to,
$$\Pr(x_{t|t}^{(i)} = x_{t|t-1}^{(j)}) = \tilde{q}_t^{(j)}.$$
4) Time update: For $i = 1, \ldots, N$, predict new particles according to
$$x_{t+1|t}^{(i)} \sim p(x_{t+1|t}|x_t^{(i)}).$$
5) Set $t := t + 1$ and iterate from step 2.

Sometimes the resampling step is omitted and just imposed when needed to avoid divergence in the filter as in the sequential importance sampling (SIS) where the weights are updated recursively as (Doucet et al., 2000)

$$q_t^{(i)} = q_{t-1}^{(i)} \cdot p_e(y_t - h(x_t^{(i)})), \quad i = 1, \ldots, N. \tag{10}$$

As the estimate of the state we chose the minimum mean square estimate, i.e.,

$$\hat{x}_t = \mathbb{E}[x_t|\mathbb{Y}_t] = \int x_t p(x_t|\mathbb{Y}_t) dx_t \approx \sum_{i=1}^N q_t^{(i)} x_t^{(i)}. \tag{11}$$

C. The Marginalized Particle Filter

In Section II-B the pdf $p(x_t|\mathbb{Y}_t)$ was approximated recursively using the particle filter for the whole state vector x_t. However, if the system has a linear, Gaussian sub-structure, we can exploit this to obtain a more efficient estimator. In practice this is done by marginalizing out the linear variables from $p(x_t|\mathbb{Y}_t)$. Denote the linear states with x_t^l and the nonlinear states x_t^n, with $X_t^n = \{x_i^n\}_{i=0}^t$. Using Bayes' theorem we obtain

$$p(X_t^n, x_t^l|\mathbb{Y}_t) = p(x_t^l|X_t^n, \mathbb{Y}_t) p(X_t^n|\mathbb{Y}_t), \tag{12}$$

where $p(x_t^l|X_t^n, \mathbb{Y}_t)$ is given by the Kalman filter and where $p(X_t^n|\mathbb{Y}_t)$ is given by a particle filter. This marginalization idea is certainly not new. It is sometimes referred to as Rao-Blackwellization (Doucet, 1998; Casella and Robert, 1996; Doucet et al., 2001b; Chen and Liu, 2000; Andrieu and Doucet, 2002; Doucet et al., 2001b; Schön, 2003; Nordlund, 2002). We will in this paper focus on the algorithmic complexity and try to analyze the performance gain.

To illustrate the complexity issues in marginalization we have chosen to study a special case where no nonlinearities enter the state-space update equation. This is a rather common case for many target tracking applications, where the dynamical model is linear and the nonlinearities enter the measurement equation. For system like that it is also common to assume Gaussian process noise. Hence, it is straightforward to use the marginalization ideas. We will use the following model for our complexity analysis of the marginalization idea

$$x_{t+1} = A_t x_t + w_t, \tag{13a}$$
$$y_t = h(x_t) + e_t. \tag{13b}$$

Hence, the nonlinearities enter the state-space model via the measurement equation. In order to be able to use the marginalized particle filter we have to marginalize out the linear states. If we partition the state vector in its nonlinear states, x_t^n, and its linear states, x_t^l, according to

$$x_t = \begin{bmatrix} x_t^n \\ x_t^l \end{bmatrix}, \tag{14}$$

we can write (13) as

$$x_{t+1}^n = A_t^n x_t^n + A_t^l x_t^l + w_t^n, \tag{15a}$$
$$x_{t+1}^l = F_t^n x_t^n + F_t^l x_t^l + w_t^l, \tag{15b}$$
$$y_t = h_t(x_t^n) + e_t, \tag{15c}$$

where

$$w_t^n \in \mathcal{N}(0, Q_t^n), \tag{15d}$$
$$w_t^l \in \mathcal{N}(0, Q_t^l), \tag{15e}$$
$$e_t \in \mathcal{N}(0, R_t). \tag{15f}$$

Here we have not considered any input signal, which should be straightforward to add. In the formulas above we assume the dimension of the linear and the nonlinear part to be $x_t^l \in \mathbb{R}^l$ and $x_t^n \in \mathbb{R}^n$, respectively. The process noise pdf is denoted p_w and the measurement noise pdf is denoted p_e. These are know, at least up to a normalization constant. Furthermore we will assume that w_t^n and w_t^l are independent and Gaussian. However, the independence assumption can easily be relaxed by rewriting the state-space model using Gram-Schmidt orthogonalization, see Kailath et al. (2000) for details. The measurement noise may have any pdf known up to a normalization constant, but in this study it is assumed Gaussian for simplicity.

The nonlinear state variables will be handled by the particle filter and the linear state variables by the Kalman filter using the marginalized particle filter. The partition (14) can be done in many different ways, therefore we introduce the notation x_t^p and x_t^k to denote the part of the state vector which is estimated by the particle filter and the Kalman filter respectively. Also note that $x_t^p \in \mathbb{R}^p$ and $x_t^k \in \mathbb{R}^k$, where the values are given

$$p \in [n, n+l], \tag{16a}$$
$$k \in [0, l]. \tag{16b}$$

For the general partitioning case we have to select $p-n$ states from l possibilities. In other words the number of combinations are given by

$$\begin{pmatrix} p-n \\ l \end{pmatrix}. \tag{17}$$

Note that we have several possible permutations here. However, not all are relevant for a practical system. For a tracking system in Cartesian coordinates a natural partitioning is to consider position, velocity and acceleration states (and so on). Hence, the possible number of partitions are reduced drastically. Our main interest is now to consider which of the states to put in the nonlinear and linear partition respectively. Two relevant aspects with respect to this partitioning are how it will affect the computational load and the estimation performance.

Using the notation x_t^p for the states that are estimated using the particle filer and x_t^k for the states that are estimated using the Kalman filter, the model will thus be

$$x_{t+1}^p = A_t^p x_t^p + A_t^k x_t^k + w_t^p, \tag{18a}$$
$$x_{t+1}^k = F_t^p x_t^p + F_t^k x_t^k + w_t^k, \tag{18b}$$
$$y_t = h_t(x_t^n) + e_t, \tag{18c}$$

where

$$w_t^p \in \mathcal{N}(0, Q_t^p), \tag{18d}$$
$$w_t^k \in \mathcal{N}(0, Q_t^k), \tag{18e}$$
$$e_t \in \mathcal{N}(0, R_t). \tag{18f}$$

As before all noise signals are considered independent. In Algorithm 2.2 the marginalized particle filter is summarized for the model given in (18). For a detailed derivation of this algorithm the reader is referred to (Schön et al., 2003). In the complexity analysis the way in which the particle filter and the Kalman filter are implemented is of course crucial.

Algorithm 2.2 (Marginalized Particle Filter):

1) Initialization: For $i = 1, \dots, N$, initialize the particles, $x_{0|-1}^{p,(i)} \sim p_{x_0^p}(x_0^p)$ and set $\{x_{0|-1}^{k,(i)}, P_{0|-1}^{(i)}\} = \{\bar{x}_0^k, \bar{P}_0\}$. Set $t = 0$.
2) For $i = 1, \dots, N$, evaluate the importance weights $q_t^{(i)} = p(y_t | \mathbb{X}_t^{p,(i)}, \mathbb{Y}_{t-1})$ according to the likelihood

$$p(y_t | \mathbb{X}_t^p, \mathbb{Y}_{t-1}) = \mathcal{N}(h_t(x_t^p), R_t) \tag{19}$$

and normalize $\tilde{q}_t^{(i)} = \frac{q_t^{(i)}}{\sum_{j=1}^N q_t^{(j)}}$.

3) Particle filter measurement update: Resample with replacement N particles according to,

$$\Pr(x_{t|t}^{p,(i)} = x_{t|t-1}^{p,(j)}) = \tilde{q}_t^{(j)}.$$

4) Particle filter time update and Kalman filter update
   a) Kalman filter measurement update, using

$$\hat{x}_{t|t}^{k,(i)} = \hat{x}_{t|t-1}^{k,(i)}, \tag{20a}$$
$$P_{t|t} = P_{t|t-1}. \tag{20b}$$

   b) Particle filter time update: For $i = 1, \dots, N$, predict new particles according to

$$x_{t+1|t}^{p,(i)} \sim p(x_{t+1|t}^p | \mathbb{X}_t^{p,(i)}, \mathbb{Y}_t), \tag{21}$$

where

$$p(x_{t+1}^{p,(i)}|\mathbb{X}_t^{p,(i)}, \mathbb{Y}_t) = \mathcal{N}(A_t x_t^{p,(i)} + A_t^k \hat{x}_{t|t}^{k,(i)},\ A_t^k P_{t|t}(A_t^k)^T + Q_t^p). \quad (22)$$

c) Kalman filter time update, using

$$\begin{aligned} \hat{x}_{t+1|t}^{k,(i)} &= F_t^k \hat{x}_{t|t}^{k,(i)} + F_t^p x_t^{p,(i)} + \\ &\quad L_t(x_{t+1|t}^{p,(i)} - A_t^p x_t^{p,(i)} - A_t^k \hat{x}_{t|t}^{k,(i)}), \\ P_{t+1|t} &= F_t^k P_{t|t}(F_t^k)^T + Q_t^k - L_t M_t L_t^T, \\ M_t &= A_t^k P_{t|t}(A_t^k)^T + Q_t^p, \\ L_t &= F_t^k P_{t|t}(A_t^k)^T M_t^{-1}, \end{aligned}$$

5) Set $t := t + 1$ and iterate from step 2.

Remark 1: In order to determine the computational complexity of the process noise, we must describe the implementation. We use a Cholesky factorization in order to generate random numbers with covariance M given the process $w \in \mathcal{N}(0, I)$,i.e.,

$$M = V^T V$$

$$\begin{aligned} \text{Cov}[V^T w] &= \mathbb{E}[(V^T w)(V^T w)^T] \\ &= V^T \mathbb{E}[ww^T] V = V^T V. \end{aligned}$$

Remark 2: The reason that the measurement update (20) in the Kalman filter is so simple in Algorithm 2.2 is that the linear state variables are not present in the measurement equation. However, let us now consider what happens if we measure some combination of nonlinear and linear state variables, i.e., we have a measurement equation according to

$$y_t = h_t(x_t^n) + C_t x_t^k + e_t, \quad (24)$$

instead of (18c). The difference is obviously that by allowing $C_t \neq 0$ we measure linear states as well. The implication in Algorithm 2.2 is that we have to evaluate one Ricatti recursion for each particle. The Kalman filter measurement update in (20) is now given by

$$\hat{x}_{t|t}^{k,(i)} = \hat{x}_{t|t-1}^{k,(i)} + K_t^{(i)}(y_t - h(x_t^{p,(i)}) - C_t x_t^{k,(i)}), \quad (25a)$$

$$P_{t|t}^{(i)} = P_{t|t-1}^{(i)} - K_t^{(i)} C_t P_{t|t-1}^{(i)}, \quad (25b)$$

$$S_t^{(i)} = C_t P_{t|t-1}^{(i)} C_t^T + R_t, \quad (25c)$$

$$K_t^{(i)} = P_{t|t-1}^{(i)} C_t^T (S_t^{(i)})^{-1}. \quad (25d)$$

Remark 3: The resampling step uses the $\mathbb{O}(N)$ algorithm from Ripley (1987). In (Bergman, 1999) an explicit MATLAB-code is given. Using other resampling schemes may be preferable both to reduce the computational cost and accuracy in the resampling. The chosen method is probably not the best available for very large N. Other more asymptotically expensive resampling schemes may be preferable when the number of particles is small.

III. Complexity Analysis

In this section the computational complexity in the marginalized particle filter is discussed from a theoretic point of view, by giving the number of floating-point operations (flops) employed by the algorithm. A flop is here defined as one addition, subtraction, multiplication, or division of two floating-point numbers. When it comes to comparing the flop count with the actual computation time we will however run into problems. This is due to the fact that issues such as cache boundaries and locality of reference will significantly influence the computation time (Boyd and Vandenberghe, 2001). Moreover, there are certain steps in the algorithm that cannot be measured using flops, for instance the cost of generating a random number and the cost of evaluating a certain nonlinear function. Despite these drawbacks it is still possible to analyze the complexity be using the computer to measure the absolute time the different steps require. These times can then be compared to the theoretical result obtained from counting flops.

A. Theoretical Analysis

In the particle filter the resampling step is proportional to the number of particles and the amount of time for generating random numbers is proportional to the number of random numbers required. We will relate the proportionality coefficients so that they reflect the flop complexity instead of the time complexity for ease of comparison with parts that only depend on matrix and vector operations. This will be referred to as the *equivalent flop* (EF) complexity. Furthermore, it will allow us to understand how the computational time will increase with the problem size.

Definition 3.1: The equivalent flop (EF) complexity for an operation is defined as the number of flops that results in the same computational time as the operation.

The dimensions of some of the entities involved in the marginalized particle filter are given in Table I and the computational complexity for some common matrix operations are summarized in Table II. We are now ready to assess the complexity issues for the marginalized particle filter for the model given by (15). We reformulate the algorithm and give the equivalent flop complexity for each relevant code line.

Here we consider a standard implementation of the particle filter according to Gordon et al. (1993) and a straightforward implementation of the Kalman filter (Kailath et al., 2000). Different implementations, such as the MKCS and various array algorithms could also be used, (Kailath et al., 2000), but are not considered here. In the particle filter we also impose a constant number of particles, N, for all time steps.

Object	Dimension
x_t^p	$\mathbb{R}^p$
x_t^k	$\mathbb{R}^k$
A_t^p	$\mathbb{R}^{p\times p}$
A_t^k	$\mathbb{R}^{p\times k}$
F_t^p	$\mathbb{R}^{k\times p}$
F_t^k	$\mathbb{R}^{k\times k}$
Q_t^p	$\mathbb{R}^{p\times p}$
Q_t^k	$\mathbb{R}^{k\times k}$

TABLE I
DIMENSIONS OF THE ENTITIES INVOLVED IN THE MARGINALIZED PARTICLE FILTER.

1) Initialization. These computations can be neglected, since this step is done only once.
2) Update the importance weights. We have to calculate $e^{-1/2\epsilon^T R^{-1}\epsilon}$ N times. The equivalent flop complexity is denoted Nc_1.
3) Resampling according to Ripley (1987), yields an equivalent flop complexity of Nc_2.
4) Particle filter time update and the Kalman filter updates:
    a) Kalman filter measurement update. No instructions, since $C_t = 0$.
    b) Particle filter time update. See Table III for the EF complexity.
    c) Kalman filter time update. See Table IV for the EF complexity.
5) Set $t := t + 1$ and iterate from step 2.

Above we have used the coefficients, c_1 for the calculation of the Gaussian likelihood, c_2 for the resampling and c_3 for the random number complexity.

Putting it all together, the total equivalent flop complexity for one iteration of the marginalized particle filter as discussed above will thus be

$$\begin{aligned}\mathcal{C}(p,k,N) = (6kp + 4p^2 + 2k^2 + p - k + pc_3 + c_1 + c_2)N+ \\ 4pk^2 + 8kp^2 + \frac{4}{3}p^3 + 5k^3 - 5kp + 2p^2. \qquad (26)\end{aligned}$$

In the case studied above we use the same covariance matrix for all Kalman filters. However, in the general case we will be forced to use different covariance matrices for each Kalman filter. This gives rise to a significant increase in the computational complexity of the algorithm. We will now briefly discuss the total complexity for this more general case.

Oper.	Size	Mult.	Add.
$A + A$	$A \in \mathbb{R}^{n\times m}$		nm
$A \cdot B$	$A \in \mathbb{R}^{n\times m}, B \in \mathbb{R}^{m\times l}$	lmn	$(m-1)ln$
$B \cdot C$	$B \in \mathbb{R}^{m\times n}, C \in \mathbb{R}^{n\times 1}$	nm	$(n-1)m$
D^{-1}	$D \in \mathbb{R}^{n\times n}$	n^3	

TABLE II
COMPUTATIONAL COMPLEXITY OF SOME COMMON MATRIX OPERATIONS.

To simplify, consider the case when the measurement dimension is much smaller than the state dimension. We study only the part dependent on the number of particles. Since we already have the measurement residuals we can neglect the complexity in (25a). Also S_t and K_t in (25c)-(25d) are neglected since they are small. The complexity in (25b) is approximately k^3, if the measurement dimension is neglected. Hence, the only difference to the previous case is that all matrices are computed for each particle and an extra k^3 is added. To summarize, the total complexity for the case when $C_t \neq 0$ is approximately given by

$$\begin{aligned}\mathcal{C}(p,k,N) = (6kp + 4p^2 + 2k^2 + p - k + pc_3 + c_1 + c_2+ \\ 4pk^2 + 8kp^2 + \frac{4}{3}p^3 + 5k^3 - 5kp + 2p^2 + k^3)N. \qquad (27)\end{aligned}$$

The analysis provided in this section is general and the main steps, which will be discussed in more detail in the subsequent section are

1) Estimate the time for one flop using linear regression.
2) Estimate the time for likelihood calculation, resampling and random number generation.
3) Relate all times using the EF measure.
4) Calculate the overall complexity $\mathcal{C}(p,k,N)$ (given by (26) in our case).

By requiring

$$\mathcal{C}(p + k, 0, N_0) = \mathcal{C}(p, k, N(k)), \qquad (28)$$

where N_0 corresponds to the number of particles used in the standard PF we can solve for $N(k)$. This gives us the number of particles, $N(k)$, that can be used in the MPF in order to obtain the same computational complexity as if the standard particle filter had been used for all states. In Fig. 1 the ratio $N(k)/N_0$ is plotted for systems with $m = 3, \ldots, 9$ states. Hence, using Fig 1 it is possible to directly find out how much we can gain in using the MPF from a computational complexity point of view when marginalization is increased. We already know that the quality will improve or remain the same when the MPF is used Doucet et al. (2001b).

We will now exemplify the expression for the over-all complexity of the algorithm, given by (26). For this discussion we will use the model from the simulations in Section IV. This model has 2 nonlinear state variables and 4 linear state variables, giving us the following intervals for the number of states estimated by the Kalman filter, k, and the number of states estimated by the particle filter, p, respectively,

$$k \in [0, 4], \qquad (29a)$$
$$p \in [2, 6]. \qquad (29b)$$

In Table V the equivalent flop complexity for the different possible partitions is given. We will now focus on the two

Instruction	**Mult.**	**Add.**	**Other**
$P_A := P_{t\|t}(A_t^k)^T$	pk^2	$(k-1)kp$	
$M := A_t^k P_A + Q_t^p$	kp^2	$(k-1)p^2 + p^2 \ \ (k>0)$	
$T_1 := chol(M)$			$(1/3)p^3 + 2p^2$
$T_2 := randn(p, N)$			pNc_3
$w := T_1 * T_2$	p^2N	$(p-1)pN$	
$T_3 := A^p x^p$	p^2N	$(p-1)pN$	
$T_4 := A^k x^k$	pkN	$(k-1)pN \ \ (k>0)$	
$\hat{x}_{t+1\|t}^p := T_3 + T_4 + w$		$2pN$	
Total	$p(k^2 + kp + 2pN + kN)$	$p((k-1)(k+p+N) + p + 2(p-1)N + 2N)$	$1/3p^3 + 2p^2 + pNc_3$

TABLE III

THE EQUIVALENT FLOP COMPLEXITY FOR THE PARTICLE FILTER TIME UPDATE IN THE MARGINALIZED PARTICLE FILTER. THE DIMENSION OF THE STATE VARIABLES THAT ARE ESTIMATED USING THE PARTICLE FILTER IS p, AND THE DIMENSION OF THE STATES ESTIMATED BY THE KALMAN FILTER IS k. FURTHERMORE, WE USE N PARTICLES.

Instruction	**Mult.**	**Add.**	**Other**
$inv_M := M^{-1}$			p^3
$L := F_t^k P_A inv_M$	$k^2p + kp^2$	$(k-1)kp + (p-1)kp$	
$T_5 := F_t^k P_{t\|t}(F_t^k)^T$	$2k^3$	$2(k-1)k^2$	
$T_6 := L_t M_t L_t^T$	$2kp^2$	$2(p-1)pk$	
$P := T_5 + Q_t^k - T_6$		$2k^2$	
$T_7 := F^k x^k$	k^2N	$(k-1)kN$	
$T_8 := F^p x^p$	kpN	$(p-1)kN$	
$T_9 := \hat{x}_{t+1\|t}^p - T_3 - T_4$		$2pN$	
$\hat{x}_{t+1\|t}^k := T_7 + T_8 + LT_9$	kpN	$(p+1)kN$	
Total	$k(kp + 3p^2 + 2k^2 + kN + 2pN)$	$(k-1)k(p+2k+N) + (p+1)kN + (p-1)k(3p+N) + 2k^2 + 2pN$	p^3

TABLE IV

THE EQUIVALENT FLOP COMPLEXITY FOR THE KALMAN FILTER TIME UPDATE IN THE MARGINALIZED PARTICLE FILTER. THE DIMENSION OF THE STATE VARIABLES THAT ARE ESTIMATED USING THE PARTICLE FILTER IS p, AND THE DIMENSION OF THE STATES ESTIMATED BY THE KALMAN FILTER IS k. FURTHERMORE, WE USE N PARTICLES.

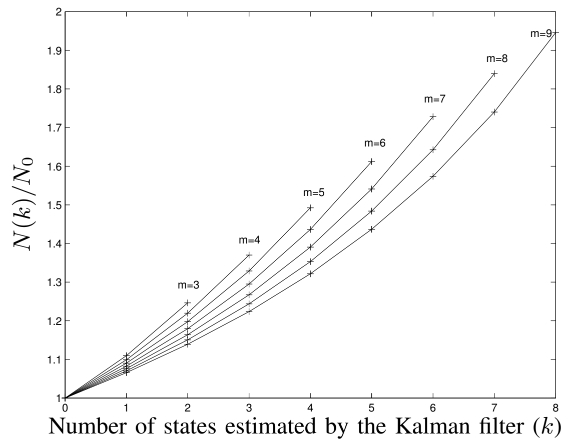

Fig. 1. This figure shows the ratio $N(k)/N_0$ for systems with $m = 3, \ldots, 9$ states. It it is apparent the MPF reduces the computational complexity, when compared to the standard PF. The case studied in this figure is $C_t = 0$.

k	p	**Complexity**
0	6	$(c_1 + c_2 + 6c_3 + 150)N + 360$
1	5	$(c_1 + c_2 + 5c_3 + 136)N + 417$
2	4	$(c_1 + c_2 + 4c_3 + 122)N + 437$
3	3	$(c_1 + c_2 + 3c_3 + 108)N + 468$
4	2	$(c_1 + c_2 + 2c_3 + 94)N + 555$

TABLE V

COMPUTATIONAL COMPLEXITY FOR DIFFERENT k AND p.

extreme cases, i.e., the full particle filter, where all states are estimated using the particle filter and the completely marginalized particle filter, where all linear states are marginalized out and estimated using the Kalman filter. If we require them to have the same computational complexity (γ) we can solve for the number of particles. The result is,

$$N_{\text{PF}} = \frac{\gamma}{c_1 + c_2 + 6c_3 + 150}, \tag{30a}$$

$$N_{\text{MPF}} = \frac{\gamma}{c_1 + c_2 + 2c_3 + 94}, \tag{30b}$$

which implies

$$N_{\text{PF}} = \underbrace{(1 - \frac{4c_3 + 56}{c_1 + c_2 + 6c_3 + 150})}_{<1} N_{\text{MPF}} \tag{31}$$

From (31) it is clear that for a given computational complexity we can use more particles in the marginalized particle filter than in the standard particle filter. In order to quantify this statement we need numerical values for the three constants c_1, c_2 and c_3. This will be given in the subsequent section.

B. Numerical Analysis

In Section III it was claimed that the equivalent flop complexity was proportional to the number of samples involved in random number generation and resampling. Below we investigate this claim numerically by simulations. The result is given in Fig. 2 and Fig. 3 for the random number generation and the resampling, respectively. Hence, after establishing the proportionality, we must estimate the coefficients and relate them to EF complexity. The coefficients, c_1, c_2, and c_3 in (26) are estimated by analyzing the actual computational time consumed by various parts of the marginalized particle filter algorithm. It was fairly easy to measure the time used for likelihood calculation, resampling and random number generation as a function of the number of particles. Hence, it should be possible to translate them into EF complexity. For hardware implementations, for instance *very large scale integration* (VLSI) or in dedicated real-time hardware systems this should be easy, since one flop would have a constant execution time. In this paper we would also want to do this on a normal desktop computer running MATLAB. Hence, we must give the EF estimation some consideration. The reason is that on a normal computer, flop count does not entirely reflect the computational time. This is due to memory caching, pipelining, efficient computational routines which are problem size dependent and memory swapping, just to mention a few things in a complex computer environment. To be able to compare the algorithmic complexity in MATLAB on a normal computer, we must deal with all these issues. The idea is to keep this part simple and fairly general, since one of our main objectives is to consider dedicated hardware implementations or a possibility to compare different computer solutions on a higher level. By repeated simulations we noticed that the flop coefficient is a function mainly of the various filter dimensions, k and p. Note also that each instruction in Table VII-VIII corresponding to the flop part is not completely negligible in time for small systems. Hence, we do an estimate of the coefficient by linear regression assuming a slope and an offset. Data from the regression is given from several simulations for each k and p, by taking the minimum value, to minimize computer system intervention. The slope EF coefficient and the offset were estimated, with values resulting in the time complexity

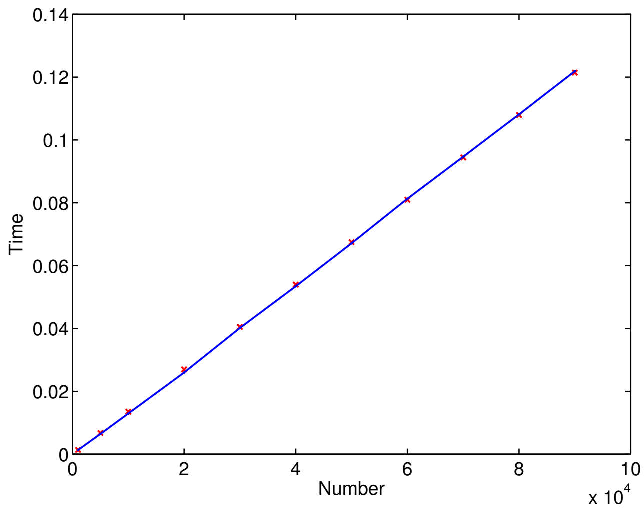

Fig. 2. The time required to generate Gaussian random number for simulated (crosses) and estimated values (solid line).

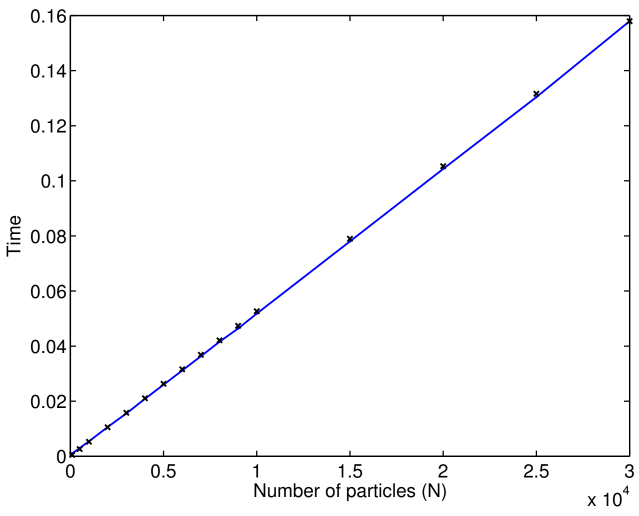

Fig. 3. The time required for the resampling step for simulated (crosses) and estimated values (solid line).

$$1.076 \cdot 10^{-8} N + 9.114 \cdot 10^{-4} \tag{32}$$

Using these and the measured time complexity for the different algorithmic operations the values for the EF coefficients are given in Table VI. These values were estimated on a Sun Blade 100 with 640 MB memory, and are processor and memory dependent. However, the technique described above can be applied on any system.

Weight	**Resampling**	**Random**
c_1	c_2	c_3
445	487	125

TABLE VI
COEFFICIENTS USED IN THE EQUIVALENT FLOP COUNT.

With the estimated EF coefficients we can now compute the complexity for the various algorithmic parts. The total computational complexity for the entire marginalized particle filter given in Algorithm 2.2 is of fundamental interest. We will only study the dominating part of (26), i.e., the part depending on the number of particles N. The remaining terms can safely be neglected due to the fact that $N \gg 1$. In Table VII the equivalent flop complexity for the various parts of the marginalized particle filter are shown in percentage of the complexity for the entire filter, for the case where $p = 6, C_t = 0$. In Table VIII we give the same data for the case $n = 6$, $C_t \neq 0$.

k	**Weight** c_1 [%]	**Resampling** c_2 [%]	**Random** pc_3 [%]	**Other** [%]
0	24	27	41	8
1	26	29	37	8
2	29	31	32	8
3	31	34	27	8
4	35	38	20	7

TABLE VII
COMPLEXITY FOR THE VARIOUS PARTS OF THE MARGINALIZED PARTICLE FILTER IN PERCENTAGE OF THE COMPLEXITY FOR THE ENTIRE FILTER. THE CASE STUDIED HERE IS $n = 6$ ($p = n - k$), $C_t = 0$.

k	**Weight** c_1 [%]	**Resampling** c_2 [%]	**Random** pc_3 [%]	**Other** [%]
0	20	22	34	23
1	21	23	30	26
2	22	24	25	28
3	23	25	20	32
4	23	26	13	38

TABLE VIII
COMPLEXITY FOR THE VARIOUS PARTS OF THE MARGINALIZED PARTICLE FILTER IN PERCENTAGE OF THE COMPLEXITY FOR THE ENTIRE FILTER. THE CASE STUDIED HERE IS $n = 6$ ($p = n - k$), $C_t \neq 0$.

In Fig. 4 the full particle filter is compared to the completely marginalized particle filter for different dimensions of the state space, n, when we have two nonlinear states. The complexity in the plot is normalized against the completely marginalized particle filter for $n = 2$. The comparison is made for the two different cases $C_t = 0$ and $C_t \neq 0$. As mentioned above, the difference between these two cases is that in the first case we only have to use one Ricatti recursion for all particles, whereas in the second case we need one Ricatti recursion for each particle. Here we neglect the offset term in the EF computation, since this analysis is most relevant for hardware implementation, and that we want the theoretical predicted time for a general system. From Fig. 4 it is clear that with $C_t = 0$, the marginalized particle filter outperforms the standard particle filter. However, when $C_t \neq 0$ and the covariance has to be calculated for each particle this is no longer true if the state dimension increases. From Fig. 4 it can be seen that for $n > 7$ the standard particle filter is cheaper than the marginalized.

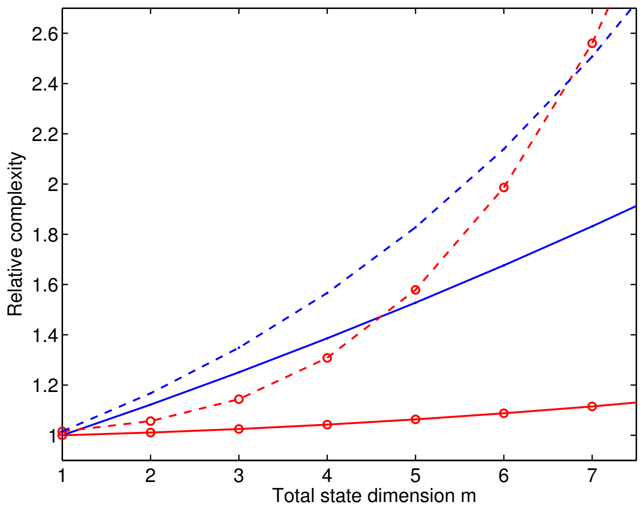

Fig. 4. Different complexities for the full particle filter and completely marginalized particle filter as a function of the state dimension when $C_t = 0$ (solid) and $C_t \neq 0$ (dashed). The marginalized particle filter is denoted by circles.

IV. Simulation

The marginalized particle filter will now be analyzed in an extensive Monte Carlo simulation. The main purpose of this simulation is to illustrate the implications of the results derived in this paper in practice. More specifically we are studying the problem of estimating the position of an aircraft using the following model of the aircraft,

$$x_{t+1} = \begin{bmatrix} 1 & 0 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & T & 0 & 0 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} w_t^n \\ w_t^l \end{bmatrix} \tag{33a}$$

$$y_t = \begin{bmatrix} \sqrt{p_x^2 + p_y^2} \\ \arctan\left(p_y/p_x\right) \end{bmatrix} + e_t \tag{33b}$$

where the state vector is $x_t = \begin{bmatrix} p_x & p_y & v_x & v_y & a_x & a_y \end{bmatrix}^T$, i.e., position, velocity and acceleration. The measurement equation gives the range and azimuth from the tracking radar system. We have discarded the height component, since a level flight is considered. The sample time is constant and denoted by T. From the model it is clear that there are two states that appear in a nonlinear fashion, the two position states $[p_x, p_y]$. Hence,

$$x_t^n = \begin{bmatrix} p_x \\ p_y \end{bmatrix}, \qquad x_t^l = \begin{bmatrix} v_x \\ v_y \\ a_x \\ a_y \end{bmatrix}. \tag{34}$$

The aircraft model (33) is contained in the model class described by (15). Here we also see that we may optionally move some of the linear state variables to the nonlinear part of the partition. The idea is now to analyze the performance

Parameter	Value	Description
T	1	Sampling time
MC	100	Number of Monte Carlo simulations
t_{final}	50	Length of the simulation
x_0	$[2000, 2000, 20, 20, 0, 0]^T$	Initial state
P_0	diag($[4, 4, 16, 16, 0.04, 0.04]$)	Initial state covariance
R	diag($[100, 10^{-6}]$)	Measurement noise covariance
Qn	diag($[4, 4]$)	Process noise covariance
Ql	diag($[4, 4, 0.01, 0.01]$)	Process noise covariance
$RMSE_{start}$	30	Start time for the RMSE calc.

TABLE IX
PARAMETER VALUES USED IN THE SIMULATIONS.

and the computational complexity of several different partitions. We restrict the study to entities of common variables, such as position, velocity and acceleration. In Table IX the parameter values relevant for the simulation are listed. In the simulations we have used one state trajectory for all Monte Carlo simulations, but different noise realizations. We have chosen to implement four different partitions. First we use a particle filter for the entire state vector, then we marginalize out the velocity states, and then the acceleration states. Finally, we arrive in the fully marginalized case, where we use the particle filter for the two nonlinear state variables only, and the Kalman filter for all four linear state variables (velocity and acceleration).

Before we describe the different simulations in detail we will give some general comments about the simulations. The time is defined as the time elapsed during a simulation in MATLAB. Hence, the time estimates must be used with caution, since the operating system and memory management may affect these values. To reduce these unwanted effects we perform several Monte Carlo simulations with the same setup and chose the minimum time value as the estimate for the computational time, i.e., the one with minimal system intervention. When the number of particles, N, or the $RMSE$ are the variable to optimize against we perform several Monte Carlo simulations in the search algorithm as well, to produce a reliable value. In the different cases the marginalization is indicated as follows. If a certain state variable is estimated using the particle filter this is indicated with a P, and if the Kalman filter is used this is indicated using a K. See the example below.

Ex: PPPP (all states are estimated using the particle filter) and KKPP (the velocity state variables are estimated using the Kalman filter and the acceleration state variables are estimated using the particle filter).

A. Investigating the Computational Complexity

The purpose of the two simulations presented in this section is to show that not only do we obtain better quality of the estimates using marginalization, we also save time, due to the reduced computational complexity. The connections to the theoretical results given in Section III are also elaborated upon.

Using a constant time we want to find the number of particles that is needed to achieve this. The study is performed by first running the full particle filter and measure the time consumed by the algorithm. A Monte Carlo simulation, using 2000 particles, is performed in order to obtain a stable estimate of the time consumed by the algorithm. To avoid intervention from the operating system the minimum value is chosen. The time is then used as the target function for the different partitions in the marginalized particle filter. To find the number of particles needed a search method is implemented and Monte Carlo simulations are used to get a stable estimate. In Table X the number of particles (N), RMSE and simulation times are shown for the different marginalization cases. The RMSE is calculated by ignoring a possible initial transient. Hence, the first $RMSE_{start}$ values according to Table IX are discarded. The total simulation length is t_{final} samples (Table IX). As seen in Table X the target function, i.e., time, varies slightly. From Table X it is clear that the different marginalized particle filters can use more particles for a given time, which is in perfect correspondence with the theoretical result given in (31). This might come as a somewhat surprising result, since the computational load has increased by running one Kalman filter for each particle. On the other hand the Kalman filter will reduce the dimension of the space in which the process noise lives. Hence, fewer random numbers are drawn, which seems to be more demanding then the calculations introduced by the Kalman filter.

	PPPP	KKPP	PPKK	KKKK
N	2000	2152	2157	2781
RMSE pos	7.82	6.53	6.72	6.40
RMSE vel	5.69	5.64	5.64	5.54
RMSE acc	0.63	0.62	0.51	0.51
Time	0.64	0.60	0.63	0.62

TABLE X
RESULTS FROM THE CONSTANT TIME SIMULATION USING MARGINALIZATION. THE MORE LINEAR VARIABLES WE MARGINALIZE OUT AND ESTIMATE USING THE OPTIMAL KALMAN FILTER THE MORE PARTICLES WE CAN USE FOR THE NONLINEAR STATES.

From the study we also conclude that the RMSE is decreasing when marginalization is used. This is also in accordance with theory, which states that the variance should decrease when we marginalize (Doucet et al., 2001b). Hence, using the marginalized particle filter we obtain better estimates at a lower computational complexity. Using the complete marginalization we have that $m = 6, k = 4$ which according to Fig. 1 gives $N(k)/N_0 = 1.44$. Hence, the theoretically predicted number of particles is $2000 \times 1.44 = 2880$. This is in good agreement with the result reported in Table X, 2781.

In the simulation just explained we used a constant time. We will now study what happens if a constant velocity RMSE is used instead. First the velocity RMSE for the full particle filter is found by a Monte Carlo simulation. This value is then used as a target function in the search for the number of particles needed by the different marginalized particle filters to achieve this value. As seen in Table XI it was hard to achieve a constant velocity RMSE in the search. However, Table XI clearly indicates that the marginalized particle filter can obtain the same RMSE using fewer particles. Since fewer particles were used in the marginalized versions of the filter the time spent for the estimation can be reduced drastically. This is further illustrated in Fig. 5 where we have plotted the computational times for the different marginalizations relative to the computational time used by the standard particle filter. The result is that using full marginalization only requires 30% of the computational resources as compared to the standard particle filter for the example studied here.

	PPPP	KKPP	PPKK	KKKK
N	1865	603	915	506
RMSE pos	7.87	7.82	7.95	7.82
RMSE vel	5.75	5.68	5.70	5.57
RMSE acc	0.62	0.68	0.54	0.53
Time	0.58	0.21	0.29	0.16

TABLE XI

RESULTS USING A CONSTANT VELOCITY RMSE FOR DIFFERENT PARTITIONS OF THE LINEAR VARIABLES. THIS TABLE CLEARLY INDICATES THAT THE MARGINALIZED PARTICLE FILTER PROVIDES GOOD ESTIMATES AT A MUCH LOWER COST THAN THE STANDARD PARTICLE FILTER. SEE FIG. 5 FOR AN ALTERNATIVE ILLUSTRATION OF THIS FACT.

B. Predicting the Computational Complexity

In this simulation we will us a constant number of particles, N, and measure the computational time required to obtain the estimates for the different partitions. This time is compared to the time we predict using the theoretical results from Section III. More specifically we compare the equivalent flop count given by (26) to the actual computational time measured in MATLAB. The result is given in Fig. 6, where it is clear that the predictions of the computational complexity based on theoretical considerations are quite good. However, there is a small error, especially in the two marginalized cases (the two

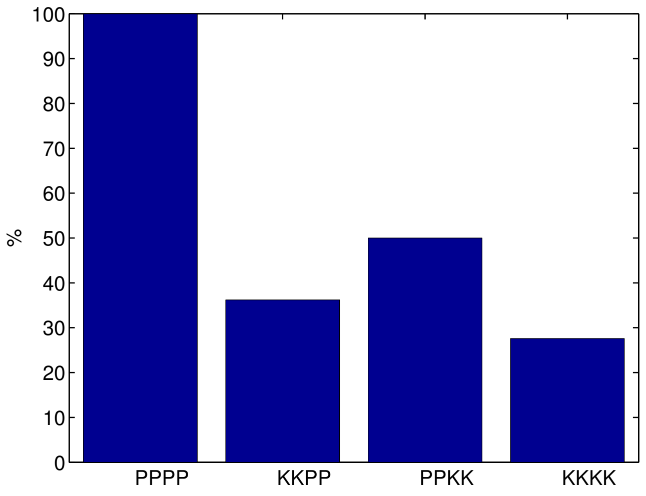

Fig. 5. The simulation times for the different partitions given in relative measures with respect to the full particle filter. From this figure it is obvious that besides providing us with better estimates the marginalized particle filter only requires 30% of the computational recourses as compared to the standard particle filter for the example studied here.

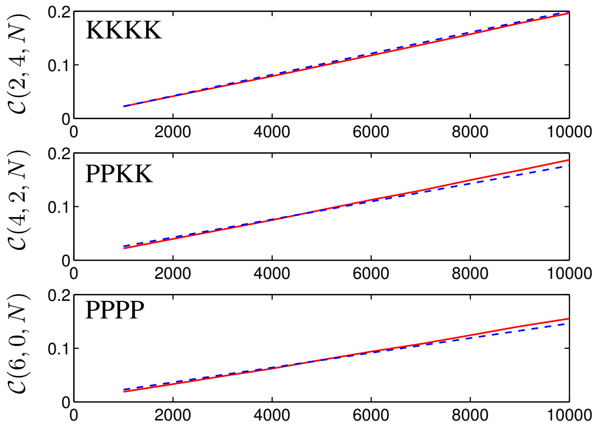

Fig. 6. Using a constant number of particles the time predicted from the theoretical results are shown by the bashed line. The solid line corresponds to the actual time measured using MATLAB. Furthermore, the top plot corresponds to the full particle filter, in the middle two linear states are marginalized and in the bottom plot all linear states are marginalized. The result is that the theoretical time corresponds very well to the measured time.

lower plots in Fig. 6). This error is mainly due to the fact that it is quite hard to predict the time used for matrix operations. This has previously been discussed in detail in Section III-B. Since we do not utilize the linear sub-structure in the full particle filter there are no Kalman filters in this case, and hence the number of matrix operations is much smaller as opposed to the marginalized particle filter, which relies on Kalman filters for estimation the linear state variables.

C. Simulation Results

From the simulations performed, we conclude that the marginalized particle filter for the model given will outperform the standard particle filter in both required computation time and performance. It is worth noting that even with a constant number of particles or even increased number of particles, the marginalized particle filter will have a lower time complexity then the full particle filter. This somewhat surprising result is due to the reduced dimension in the process noise, even if the Kalman filter will introduce extra computational load. We have also shown that the theoretical results derived for the computational complexity are in accordance with the actual time used in the simulations.

V. Algorithm Modifications

In the previous sections we have discussed the performance and complexity issues for the particle filter and the marginalized particle filter. In this section we will briefly discuss some methods to reduced computations given any of these methods. We focus on the algorithmic parts described in the analysis and simulations. i.e., likelihood evaluation, resampling and random number generation.

a) Likelihood: A lot of time is spent evaluating the likelihood weights, i.e., $w_t^{(i)} = p_e(y_t - h(x_t^{(i)}))$. For a Gaussian pdf we could in principle save some computations by approximating the density with a similar one, that requires fewer computations.

b) Resampling: The resampling in the SIR particle filter can be postponed until needed to avoid divergence. This is the SIS method, Doucet et al. (2000). For many systems the particle filter algorithm compete with other resources during the recursive update. Hence, postponing some calculations may give the processor a chance to do some other crucial computations. However, for a dedicated hardware or for some application with just the particle filter algorithm implemented there is no benefit using this method since the system must be dimensioned for the loop with full resampling. Here we propose a method to spread the resampling burden over several sample periods. The idea is that we reduce the number of participation particles in the resampling procedure and hence the computational time. We use different particles in the consecutive resampling steps to ensure that every particle is re-sampled over some time horizon. The easiest way is probably to do this by selecting a portion of the particles by a uniform sampling, but since this involve an enhanced computation we propose a deterministic approach, where the indices are chosen in such a way that every particle will interact after a fixed time horizon. In Hol (2004) four different resampling schemes are compared with respect to their computational complexity and performance.

c) Random number generation: Since a lot of time is spent producing random numbers, one way to reduce this computation is to have a look-up table with pre-processed random numbers. If the storage of these numbers is not a problem a lot of computations can be saved. However, for most problems the table is not big enough so sooner or later values must be re-used introducing a risk for dependence in the random numbers. If this is handled properly many systems may work. However, for Monte Carlo evaluation this introduces a problem.

d) Number of particles: Since the computational burden is highly dependent on the number of particles, N, one method is to introduce a varying number of particles, i.e., N_t. Using some criteria we may increase or decrease the number of particles. This method assumes that we are in a system so that other resources may benefit from the reduction. Otherwise we must adjust the sample time in order to keep a fixed computational power during a sample period.

e) System design: For many systems there is a choice if a signal should be interpreted as a measurement or as a input signal. The only difference is if we should incorporate the noise in the measurement- or process noise term. One could evaluate the computations required and chose the one with lowest computational burden for the selected filter type.

VI. Conclusions

The contribution in this paper is a way to analyze and partition the marginalized particle filter from a complexity point of view. It also gives insight in the marginalized particle filter algorithm. The method is general and can be applied to a large class of problems. In particular a common target tracking problem is analyzed in detail and for this exaple it is shown that the marginalized particle filter only need 30% of the computational resources as compare to the standard particle filter to obtain the same estimation performance. The complexity analysis is done theoretically, counting the number of floating-point operations, estimating the impact on complex algorithmic parts such as random number generation and resampling, introducing the equivalent flop measure. In an extensive Monte Carlo simulation different aspects, such as minmum computational time and estimation performance are compared for different partitions. Based on the results we gave some guidelines on how to improve performance and how to reduce the computational time.

References

Anderson, B. and Moore, J. (1979). *Optimal Filtering*. Information and system science series. Prentice Hall, Englewood Cliffs, New Jersey.

Andrieu, C. and Doucet, A. (2002). Particle filtering for partially observed Gaussian state space models. *Journal of the Royal Statistical Society*, 64(4):827–836.

Andrieu, C. and Godsill, S. (2000). A particle filter for model based audio source separation. In *International Workshop on Independent Component Analysis and Blind Signal Separation (ICA 2000)*, Helsinki, Finland.

Bergman, N. (1999). *Recursive Bayesian Estimation: Navigation and Tracking Applications*. PhD thesis, Linköping University. Dissertation No. 579.

Boyd, S. and Vandenberghe, L. (2001). Convex optimization.

Casella, G. and Robert, C. (1996). Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94.

Chen, R. and Liu, J. (2000). Mixture Kalman filters. *Journal of the Royal Statistical Society*, 62(3):493–508.

Doucet, A. (1998). On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR.310, Signal Processing Group, Department of Engineering, University of Cambridge.

Doucet, A., de Freitas, N., and Gordon, N., editors (2001a). *Sequential Monte Carlo Methods in Practice*. Springer Verlag.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*.

Doucet, A., Gordon, N., and Krishnamurthy, V. (2001b). Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624.

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

Gustafsson, F. (2000). *Adaptive Filtering and Change Detection*. John Wiley & Sons.

Hol, J. (2004). Resampling in particle filters. LiTH-ISY-EX-ET-0283-2004, Automatic control and communications systems, Linköping university.

Jazwinski, A. (1970). *Stochastic processes and filtering theory*. Mathematics in science and engineering. Academic Press, New York.

Kailath, T., Sayed, A., and Hassibi, B. (2000). *Linear Estimation*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, New Jersey.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans. AMSE, J. Basic Engineering*, 82:35–45.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. Springer, New York, USA.

Nordlund, P.-J. (2002). *Sequential Monte Carlo Filters and Integrated Navigation*. Licentiate thesis, Linköping university. Thesis No. 945.

Ripley, B. (1987). *Stochastic Simulation*. John Wiley.

Schön, T., Gustafsson, F., and Nordlund, P.-J. (2003). Marginalized particle filters for nonlinear state-space models. Technical Report LiTH-ISY-R-2548, Department of Electrical Engineering, Linköping University.

Schön, T. (2003). *On Computational Methods for Nonlinear Estimation*. Licentiate thesis, Linköping university. Thesis No. 1047.

Smith, A. and Gelfand, A. (1992). Bayesian statistics without tears: A sampling-resampling perspective. *The American Statistician*, 46(2):84–88.