Complexity Analysis of the Marginalized Particle Filter

Rickard Karlsson, Thomas Schön and Fredrik Gustafsson, *Member IEEE*

***Abstract*— In this paper the computational complexity of the marginalized particle filter is analyzed and a general method to perform this analysis is given. The key is the introduction of the *equivalent flop* measure. In an extensive Monte Carlo simulation different computational aspects are studied and compared with the derived theoretical results.**

***Index Terms*— Nonlinear estimation, Marginalized particle filter, Kalman filter, Complexity analysis, Equivalent Flop**

I. Introduction

In *particle filter* (PF) applications, knowledge of the computational complexity is often of paramount importance. In this paper the computational complexity issues that arise in the use of the *marginalized particle filter* (MPF), also called the Rao-Blackwellized particle filter are studied. The MPF is a clever combination of the standard PF, [10], and the *Kalman filter* (KF), [12], which can be used when the model contains a linear substructure, subject to Gaussian noise. It is a well known fact that in some cases it is possible to obtain better estimates, i.e., estimates with reduced variance, using the MPF instead of using the standard PF [8]. By now quite a lot has been written about the MPF, see e.g., [1, 2, 5–7, 15]. However, to the best of the authors knowledge, nothing has yet been written about complexity issues. In this article, expressions for the complexity, $\mathcal{C}(p, k, N)$, are derived, where p, k represent the states estimated using the PF and the KF, respectively and N denotes the number of particles. A general method to analyze the computational complexity of the MPF will be provided. The method is illustrated using a common tracking model, but can be applied to a much broader class of models. For more details of the proposed method, the reader is referred to [13].

II. The Marginalized Particle Filter (MPF)

Many nonlinear estimation problems can be handled using the particle filter. A general state-space model

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

has both nonlinear dynamics, f, and nonlinear measurements, h. The noise processes w_t and e_t have known probability density functions. If the state-space model contains a linear-Gaussian substructure, this can be exploited to obtain better estimates using the MPF. In this article the case with linear-Gaussian dynamics,

$$x_{t+1} = A_t x_t + w_t, \qquad w_t \in \mathcal{N}(0, Q_t), \tag{2a}$$
$$y_t = h(x_t^n) + C_t x_t^l + e_t, \tag{2b}$$

is discussed. In this context the state variable $x_t \in \mathbb{R}^m$ is

$$x_t = \begin{bmatrix} x_t^l \\ x_t^n \end{bmatrix}, \tag{3}$$

where $x_t^l \in \mathbb{R}^l$ denote the linear states and $x_t^n \in \mathbb{R}^n$ denotes the nonlinear states. Furthermore, $\mathbb{X}_t^n = \{x_i^n\}_{i=0}^t$ and $\mathbb{Y}_t = \{y_i\}_{i=0}^t$. Using Bayes' theorem,

$$p(\mathbb{X}_t^n, x_t^l | \mathbb{Y}_t) = p(x_t^l | \mathbb{X}_t^n, \mathbb{Y}_t) p(\mathbb{X}_t^n | \mathbb{Y}_t), \tag{4}$$

where $p(\mathbb{X}_t^n | \mathbb{Y}_t)$ is given by the PF and $x_t^l | \mathbb{X}_t^n$ is linear-Gaussian, i.e., $p(x_t^l | \mathbb{X}_t^n, \mathbb{Y}_t)$ is given by the KF. This marginalization idea is certainly not new [1, 4, 5, 7, 8, 14, 15]. The state vector x_t can be partitioned into two parts, $x_t^p \in \mathbb{R}^p$ and $x_t^k \in \mathbb{R}^k$, which are estimated using the PF and the KF respectively. Furthermore, $p \in [n, n+l]$, $k \in [0, l]$ and for the general partitioning case $p - n$ states can be selected from l possibilities.

It is interesting to consider which states to put in the nonlinear and the linear partition, respectively. Two relevant aspects with respect to this partitioning are how it will affect the computational complexity and the estimation performance. This will be discussed using the following model

$$x_{t+1}^p = A_t^p x_t^p + A_t^k x_t^k + w_t^p, \qquad w_t^p \in \mathcal{N}(0, Q_t^p), \tag{5a}$$
$$x_{t+1}^k = F_t^p x_t^p + F_t^k x_t^k + w_t^k, \qquad w_t^k \in \mathcal{N}(0, Q_t^k), \tag{5b}$$
$$y_t = h_t(x_t^p) + C_t x_t^k + e_t, \qquad e_t \in \mathcal{N}(0, R_t), \tag{5c}$$

where the noise is assumed to be independent. This is no restriction, since the case of dependent noise can be reduced to the case of independent noise using a Gram-Schmidt procedure [11]. In Alg. 1 the MPF is summarized for the model given in (5) (with $C_t = 0$, for the sake of brevity). For a detailed derivation (including the case $C_t \neq 0$), the reader is referred to [15].

Alg. 1 (Marginalized Particle Filter (MPF), $C_t = 0$):

1) Initialization: For $i = 1, \ldots, N$, 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 $t = 0$.

R. Karlsson, T. Schön and F. Gustafsson are with the Department of Electrical Engineering, Linköping University, Linköping, Sweden (e-mail: (rickard, schon, fredrik)@isy.liu.se, phone:+46 13 281000, fax: +46 13 282622).
EDICS: 2-ADPT Adaptive Systems and Filtering, 1-NPAR Parametric and Non-parametric Statistical Signal Processing

2) For $i = 1, \ldots, N$, evaluate the importance weights $q_t^{(i)} = p(y_t|\mathbb{X}_t^{p,(i)}, \mathbb{Y}_{t-1}) = \mathcal{N}(h_t(x_t^{p,(i)}), R_t)$ and normalize $\tilde{q}_t^{(i)} = \frac{q_t^{(i)}}{\sum_{j=1}^N q_t^{(j)}}$.
3) PF measurement update (resampling): Resample N particles with replacement according to,

$$\Pr(x_{t|t}^{p,(i)} = x_{t|t-1}^{p,(j)}) = \tilde{q}_t^{(j)}. \tag{6}$$

4) PF time update and KF update
   a) KF measurement update,

$$\hat{x}_{t|t}^{k,(i)} = \hat{x}_{t|t-1}^{k,(i)},\ P_{t|t} = P_{t|t-1}. \tag{7}$$

   b) PF time update (prediction): For $i = 1, \ldots, N$,

$$x_{t+1|t}^{p,(i)} \sim p(x_{t+1|t}^{p}|\mathbb{X}_t^{p,(i)}, \mathbb{Y}_t), \tag{8}$$

   where

$$p(x_{t+1}^{p,(i)}|\mathbb{X}_t^{p,(i)}, \mathbb{Y}_t) = \mathcal{N}(A_t x_t^{p,(i)} + A_t^k \hat{x}_{t|t}^{k,(i)},\ A_t^k P_{t|t}(A_t^k)^T + Q_t^p). \tag{9}$$

   c) KF time update,

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

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

III. Complexity Analysis

In this section the computational complexity of the MPF is discussed from a theoretical point of view, by giving the number of *floating-point operations* (flops) used in the algorithm. A flop is here defined as one addition, subtraction, multiplication, or division of two floating-point numbers. However, problems occur when the flop count is compared to the actual computation time. This is due to the fact that issues such as cache boundaries and locality of reference will significantly influence the computation time [3]. Moreover, there are certain steps in the algorithm that cannot easily be measured in flops, for instance the cost of generating a random number and the cost of evaluating a nonlinear function. Despite these drawbacks it is still possible to analyze the complexity using the computer to measure the absolute time that the different steps require. These can then be compared to the theoretical result obtained from counting flops. In the PF the computational complexity of 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. The proportionality coefficients are related to 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.

TABLE I

The EF complexity for the PF (upper) and KF time update (lower) in Alg. 1 († represents the case $k > 0$, ‡ represent operations not from matrix multiplications and additions, such as resampling, random number generation etc.).

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	$(k-1)p^2 + p^2$ †	
$T_1 := chol(M)$			$\frac{p^3}{3} + 2p^2$
$T_2 := randn(p, N)$			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	$(k-1)pN$ †	
$\hat{x}_{t+1\|t}^p := T_3 + T_4 + w$		$2pN$	
$inv_M := M^{-1}$			p^3
$L := F_t^k P_A inv_M$	$k^2p + kp^2$	$k^2p + p^2k - 2kp$	
$T_5 := F_t^k P_{t\|t}(F_t^k)^T$	$2k^3$	$2(k-1)k^2$	
$T_6 := L_t M_t L_t^T$	$2kp^2$	$2(p-1)pk$	
$P := T_5 + Q_t^k - T_6$		$2k^2$	
$T_7 := F^k x^k$	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	$(p+1)kN$	

Definition 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.

A. Nonlinear Measurements ($C_t = 0$)

In this section the case $C_t = 0$ in (5c) is discussed. The total complexity of Alg. 1 is given for each code line in Table I. For instance, the first instruction $P_{t|t}(A_t^k)^T$ corresponds to multiplying $P_{t|t} \in \mathbb{R}^{k\times k}$ with $(A_t^k)^T \in \mathbb{R}^{k\times p}$, which requires pk^2 multiplications and $(k-1)kp$ additions [9]. The total EF complexity is given by:

$$\begin{aligned}
\mathcal{C}(p,k,N) = 4pk^2 + 8kp^2 + \frac{4}{3}p^3 + 5k^3 - 5kp + 2p^2 + \\
(6kp + 4p^2 + 2k^2 + p - k + pc_3 + c_1 + c_2)N.
\end{aligned} \tag{11}$$

Above, the coefficient c_1 has been used for the calculation of the Gaussian likelihood, c_2 for the resampling and c_3 for the random number complexity. Note that, when $C_t = 0$ the same covariance matrix is used for all Kalman filters, which reduces the computational complexity.

The analysis provided above is general and the main steps, which will be discussed in the subsequent section are as follows:

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 $\mathcal{C}(p, k, N)$.

By requiring $\mathcal{C}(p + k, 0, N_{\text{PF}}) = \mathcal{C}(p, k, N(k))$, where N_{PF} corresponds to the number of particles used in the standard PF $N(k)$ can be solved for. This gives 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 $N(k)/N_{\text{PF}}$ is plotted for systems with $m = 3, \ldots, 9$ states. Hence, using Fig. 1 it is

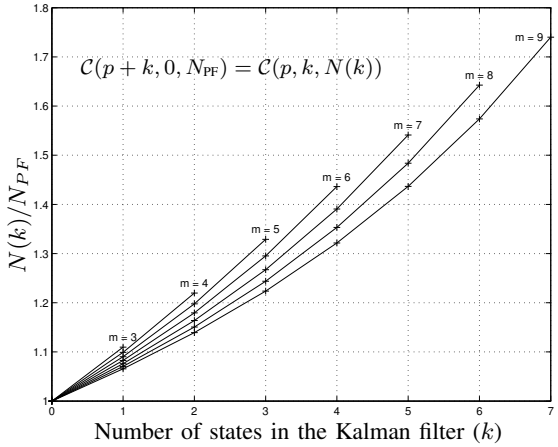

Fig. 1. The ratio $N(k)/N_{\text{PF}}$ for systems with $m = 3, \ldots, 9$ states and $C_t = 0$, $n = 2$ is shown. It is apparent the MPF can use more particles for a given computational complexity, when compared to the standard PF.

possible to directly find out how much there is to gain in using the MPF from a computational complexity point of view. The figure also shows that the computational complexity is always reduced when the MPF can be used instead of the standard PF. Furthermore, it is well-known that the quality of the estimates will improve or remain the same when the MPF is used [8].

B. Mixed Nonlinear/Linear Measurements ($C_t \neq 0$)

It is now assumed that $C_t \neq 0$ in (5c), which implies that the Riccati recursions have to be evaluated for each particle. This results in a significant increase in the computational complexity. Hence, different covariance matrices have to be used for each Kalman filter, implying that (11) has to be modified. For details the reader is referred to [13], but approximately the complexity is given by

$$\mathcal{C}(p,k,N) = (6kp + 4p^2 + 2k^2 + p - k + pc_3 + c_1 + c_2 + 4pk^2 + 8kp^2 + \frac{4}{3}p^3 + 5k^3 - 5kp + 2p^2 + k^3)N. \quad (12)$$

The problem with the increased complexity in (12) might be reduced simply by moving one or more linear states from x_t^k to x_t^p. In Fig. 2 the ratio $N(k)/N_{PF}$ is plotted for systems with $m = 3, \ldots, 9$ states. For systems with few states the MPF is more efficient than the standard PF. However, for systems with more states, where most of the states are marginalized the standard PF becomes more efficient than the MPF. The reason is the increased complexity in the Kalman filters due to the increased dimension in the Riccati recursions. For example; according to Fig. 2 a system with 9 states, where 7 are marginalized, $N(k) < N_{PF}$.

IV. Target Tracking Example

The general method for analyzing the computational complexity presented in the previous section is illustrated using

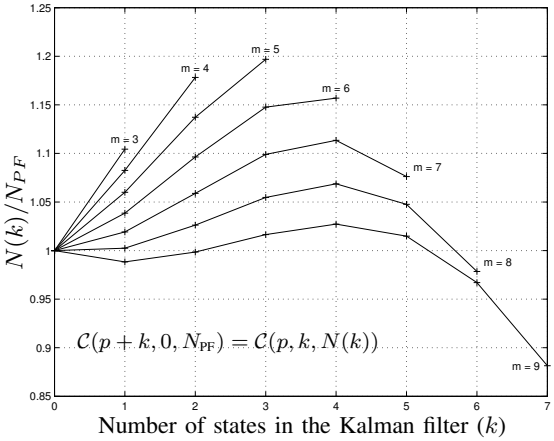

Fig. 2. The ratio $N(k)/N_{\text{PF}}$ for systems with $m = 3, \ldots, 9$ states and $C_t \neq 0$, $n = 2$ is shown. For systems with high state dimension and many marginalized states the standard PF can use more particles than the MPF.

a common tracking model. The problem of estimating the position and velocity of an aircraft is studied using

$$x_{t+1} = \begin{bmatrix} 1 & 0 & T & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & T & 0 & T^2/2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_t + w_t, \quad (13a)$$

$$y_t = \begin{bmatrix} \sqrt{p_x^2 + p_y^2} \\ \arctan\left(p_y/p_x\right) \end{bmatrix} + e_t \quad (13b)$$

where $Q = \text{Cov}[w] = \text{diag}[1 \;\; 1 \;\; 1 \;\; 1 \;\; 0.01 \;\; 0.01], R = \text{Cov}[e] = \text{diag}[100 \;\; 10^{-6}]$ and the state vector is $x_t = [p_x \; p_y \; v_x \; v_y \; a_x \; a_y]^T$, i.e., position, velocity and acceleration. The measurement equation gives the range and azimuth from the radar system.

In the subsequent section a numerical study of the computational complexity is given, where the theoretical expressions previously derived are validated. Furthermore the MPF will be analyzed in an extensive *Monte Carlo* (MC) simulation using the model described in (13). The main purpose of this simulation is to illustrate the implications of the results derived in this paper. In the simulations one state trajectory with different noise realizations have been used. The purpose of the simulations presented here is to show that using marginalization the computational complexity is significantly reduced and the quality of the estimates is improved.

A. Numerical Complexity Analysis

The model (13) has 2 nonlinear state variables and 4 linear state variables, implying $k \in [0, 4]$, $p \in [2, 6]$. Two cases are now studied, the full PF, where all states are estimated using the PF and the completely marginalized PF, where all linear states are marginalized out and estimated using the KF. Requiring the same computational complexity, i.e.,

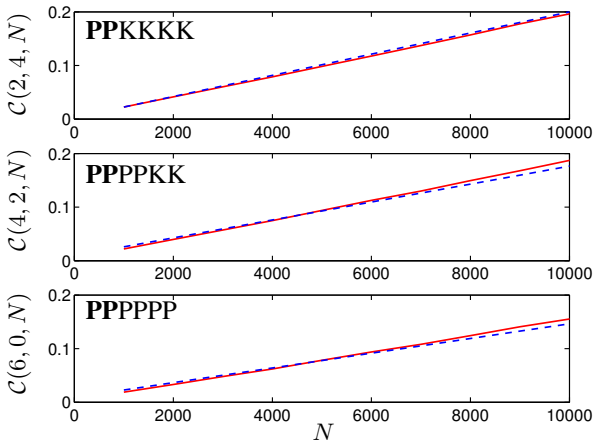

Fig. 3. Using a constant number of particles the times predicted from the theoretical results are shown by the dashed line. The solid line corresponds to the actual time measured using MATLAB. If a certain state variable is estimated using the PF this is indicated with a P, and if the KF is used this is indicated using a K.

$\mathcal{C}(6,0,N_{PF}) = \mathcal{C}(2,4,N_{MPF})$, gives

$$N_{\text{PF}} = \underbrace{\left(1 - \frac{4c_3 + 56}{c_1 + c_2 + 6c_3 + 150}\right)}_{<1} N_{\text{MPF}}. \qquad (14)$$

From (14) it is clear that for a given computational complexity more particles can be used in the MPF than in the standard PF. Expression (14) is a specific instance of what has been plotted in Fig. 1, where the curve corresponds to $m = 6$, $k = 4$. In order to quantify this statement numerical values for the three constants c_1, c_2 and c_3 are needed. They are estimated by analyzing the actual computational time consumed by various parts of the MPF 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. The problem is to relate them to the time consumed for a single flop. For simpler hardware implementations one flop would have a constant execution time. However, in order to do this on a normal desktop computer running MATLAB, the EF estimation has to be considered, since flop count does not entirely reflect the actual computational time. This is due to memory caching, pipelining, efficient computational routines which are problem size dependent and memory swapping. For the tracking example from (13) the estimated coefficients are $c_1 = 445, c_2 = 487$ and $c_3 = 125$ (on a Sun Blade 100 with 640 MB memory).

By comparing the EF complexity given by (11) to the actual computational time measured in MATLAB it is clear that the predictions of the computational complexity based on theoretical considerations are quite good indeed. The result is given in Fig. 3. The small error is mainly due to the fact that it is quite hard to predict the time used for matrix operations, as previously discussed.

B. Simulation - Constant Time

Using a constant time the number of particles that can be used is computed. The study is performed by first running the full PF and measure the time consumed by the algorithm. A MC simulation, using $N = 2000$ 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 MPF. To find the number of particles needed a search method is implemented and MC simulations are used to get a stable estimate. In Table II the number of particles (N), the *root mean square error* (RMSE) and simulation times are shown for the different marginalization cases. RMSE is defined as $\left(\frac{1}{T_f}\sum_{i=1}^{T_f}\frac{1}{N_{MC}}\sum_{j=1}^{N_{MC}}\|x_i^{true} - \hat{x}_i^{(j)}\|_2^2\right)^{1/2}$, where T_f is the number of time samples and $N_{MC} = 100$ is the number of MC simulations used. From Table II it is clear that the

TABLE II
RESULTS FROM THE CONSTANT TIME SIMULATION.

	PPPPPP	**PP**KKPP	**PP**PPKK	**PP**KKKK
N	2000	2029	1974	2574
RMSE pos	7.10	5.81	5.76	5.60
RMSE vel	3.62	3.27	3.28	3.21
RMSE acc	0.52	0.47	0.45	0.44
Time	0.59	0.58	0.57	0.60

different MPFs can use more particles for a given time, which is in perfect correspondence with the theoretical result given in (14). From the study it is also concluded that the RMSE is decreasing when marginalization is used. This is also in accordance with theory, which states that the variance should decrease or remain unchanged when marginalization is used [8]. Furthermore, Table II verifies the theoretical results presented in Fig. 1. From this figure it is also clear that the complete marginalization ($m = 6, k = 4$) gives $N(k)/N_0 = 1.44$. Hence, the theoretically predicted number of particles is $2000 \times 1.44 = 2880$. This is in quite good agreement with the result reported in Table II, 2574.

C. Simulation - Constant Velocity RMSE

In this section it is studied what happens if a constant velocity RMSE is used. First the velocity RMSE for the full PF is found using a MC simulation. This value is then used as a target function in the search for the number of particles needed by the different MPFs. Table III clearly indicates that the MPF can obtain the same RMSE using fewer particles. The result is that using full marginalization only requires 14% of the computational resources as compared to the standard PF in this example.

TABLE III
RESULTS USING A CONSTANT VELOCITY RMSE.

	PPPPPP	**PP**KKPP	**PP**PPKK	**PP**KKKK
N	2393	864	943	264
RMSE pos	7.07	6.98	7.12	7.27
RMSE vel	3.58	3.60	3.65	3.61
RMSE acc	0.50	0.51	0.49	0.48
Time	0.73	0.26	0.28	0.10

V. CONCLUSION

The contribution in this paper is a systematic approach to analyze and partition the marginalized particle filter from a computational complexity point of view. The method is general and can be applied to a large class of problems. To illustrate the idea, a common target tracking problem is analyzed in detail. The complexity analysis is performed theoretically by counting the number of flops and using the equivalent flop measure to account for complex algorithmic parts such as random number generation and resampling. In an extensive Monte Carlo simulation different performance aspects are shown, and the theoretical results are illustrated and validated.

ACKNOWLEDGMENT

This work was supported by VINNOVA's Center of Excellence ISIS (Information Systems for Industrial Control and Supervision), in particular the partner NIRA Dynamics, and by the Swedish Research Council (VR). The authors would also like to thank the reviewers for their constructive comments.

REFERENCES

[1] C. Andrieu and A. Doucet. Particle filtering for partially observed Gaussian state space models. *Journal of the Royal Statistical Society*, 64(4):827–836, 2002.
[2] C. Andrieu and S.J. Godsill. A particle filter for model based audio source separation. In *ICA 2000*, Helsinki, Finland, Jun. 2000.
[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.
[4] G. Casella and C.P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.
[5] R. Chen and J.S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society*, 62(3):493–508, 2000.
[6] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.
[7] A. Doucet, S.J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.
[8] A. Doucet, N. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, 2001.
[9] G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 3 edition, 1996.
[10] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. A novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.
[11] T. Kailath, A.H. Sayed, and B. Hassibi. *Linear Estimation*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, New Jersey, 2000.
[12] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. AMSE, J. Basic Engineering*, 82:35–45, 1960.
[13] R. Karlsson, T. Schön, and F. Gustafsson. Complexity analysis of the marginalized particle filter. Technical Report LiTH-ISY-R-2611, Department of Electrical Engineering, Linköping University, 2004.
[14] P-J. Nordlund. *Sequential Monte Carlo Filters and Integrated Navigation*. Licentiate thesis, Linköping university, 2002. Thesis No. 945.
[15] T. Schön, F. Gustafsson, and P-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *Accepted for publication in IEEE Transactions on Signal Processing*, 2004.