Parallelization of the Particle Filter via Series Expansions

Olov Rosén and Alexander Medvedev

***Abstract*— Parallelizability of an algorithm is nowadays a highly desirable property as computer hardware is becoming increasingly parallel. In this paper, a formulation of the particle filtering algorithm, suitable for parallel or distributed computing, is proposed. From the particle set, a series expansion is fitted to the posterior probability density function. The global information expressed by the particles can in this way be compressed to a few informative coefficients that can be efficiently communicated between the local processing units. Experiments on a shared-memory multicore processor using up to 8 cores show that linear speedup in the number of used cores is achieved.**

I. Introduction

Nonlinear non-Gaussian filtering problems arise in numerous signal processing, system modeling, and control applications. To mention a few, communications, radar and sonar technology as well as satellite navigation are engineering domains where nonlinear filtering is frequently applied. More recently it has successfully expanded to biomedical applications [1]. In general this is a computationally costly problem to solve. Even though nonlinear filtering methods are capable of providing far better estimation accuracy in many well-modeled applications, linear (or linearized) models are often resorted to, due to the heavy computational burden associated with optimal nonlinear filtering.

Consider state estimation in a discrete-time nonlinear state-space model with additive process noise

$$\mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{v}_t, \quad (1)$$

$$\mathbf{y}_t = g_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t), \quad (2)$$

where $\mathbf{x}_t \in \mathbb{R}^{n_x}$ is the scalar state, $\mathbf{u}_t \in \mathbb{R}^{n_u}$ is the deterministic (control) input, $\mathbf{y}_t \in \mathbb{R}^{n_y}$ is the measurement, $\mathbf{v}_t \in \mathbb{R}^{n_x}$ and $\mathbf{e}_t \in \mathbb{R}^{n_y}$ are mutually independent white noise sequences with known probability density functions (PDFs) $p_v(\mathbf{v})$ and $p_e(\mathbf{e})$, respectively. The state and measurement functions $f_t(\cdot,\cdot)$ and $g_t(\cdot,\cdot,\cdot)$ are arbitrary known functions, and t denotes discrete time.

The filtering problem in hand is to provide an estimate $\hat{\mathbf{x}}_t$ of the (hidden) state $\mathbf{x}_t$, given the measurements $\mathbf{Y}_t = \{\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_t\}$. A powerful and flexible method to solve the problem is the particle filter (PF) but it comes with the well known drawback of a high computational cost.

The authors are partially supported by the project "Computationally Demanding Real-Time Applications on Multicore Platforms" funded by Swedish Foundation for Strategic Research.

O. Rosén and A. Medvedev are with the Department of Information Technology, Uppsala University, Uppsala SE-751 05, Sweden {olov.rosen, alexander.medvedev}@it.uu.se

Due to a wide spectrum of reasons, such as power consumption and heat dissipation concerns, the computational power of hardware is today increased by adding more parallel processors on a single unit, rather than increasing the performance of a single core processor. To improve the execution time of a program on parallel hardware it must hence be parallelized.

Parallelization of the PF has for a long time been seen as a way of improving its real-time feasibility[2], [3] . In [4], for instance, four different parallel particle filters: the globally distributed particle filter [5], sampling with proportional allocation and sampling with non-proportional allocation [6], as well as the Gaussian particle filter (GPF) [7] are evaluated. It is found that the GPF is the most suitable option for a fast and scalable implementation.

The GPF makes the approximation

$$p(\mathbf{x}_t|\mathbf{Y}_t) \approx \gamma(\mathbf{x}_t, \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t),$$

where $\gamma(\mathbf{x}, \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ denotes a multivariate Gaussian PDF with the mean $\boldsymbol{\mu}_t \in \mathbb{R}^D$ and the covariance $\boldsymbol{\Sigma}_t \in \mathbb{R}^{D\times D}$. By this approximation, the information contained in the particle set can be compressed to a few informative coefficients (the mean and covariance), and hence efficiently communicated among the parallel processing units. The Gaussian approximation of a posterior is though a rather restrictive assumption that infringes the generality of the method.

In this paper, a method to fit a truncated series expansion

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{Y}_t) \approx \sum_{\mathbf{k}\in\mathbf{K}} a_{t+1}^{(\mathbf{k})}\phi_{\mathbf{k}}(\boldsymbol{x}_{t+1}), \quad (3)$$

to the particle set for some set of basis functions $\Phi = \{\phi_{\mathbf{k}}(\boldsymbol{x})\}_{\mathbf{k}\in\mathbb{N}^D}$, where $\mathbf{K}$ is the set of coefficients that shall be included in the expansion, is suggested. In a sense, it can be seen as an extension of the GPF, as it reduces to the GPF with Φ chosen to be the Hermitian basis, and only the first basis function used in the approximation, i.e. $\mathbf{K} = \mathbf{0}$. The method enjoys the favorable parallelization properties of the Gaussian particle filter, as only some few coefficients $\{a^{(\mathbf{k})}\}_{\mathbf{k}\in\mathbf{K}}$ have to be communicated, but abolishes the restriction of the posterior distribution being Gaussian.

The problem of fitting a series expansion to a random sample (i.e. particle set in this case) has been studied in [8], [9], [10]. A useful property of the series expansion estimator is that it possesses a convergence rate that is independent of the dimension of the problem. This is in contrast with most other non-parametric estimators whose convergence rate severely deteriorates with increasing dimension. Therefore,

the series estimator constitutes an appealing option for high-dimensional problems. Further, the series expansion method as well exhibits beneficial interpolation properties so that less particles are required to give an approximation to the posterior for a given accuracy.

Modifications of the GPF, such as the Gaussian sum particle filter (GSPF) [11], allow non-Gaussian posteriors to be approximated by a Gaussian mixture. However the mixands are required to be refitted frequently if the filter should operate near optimality [12]. This poses a severe obstacle to an efficient parallelization, as the refitting requires information access to the global posterior distribution and hence parallelizes poorly.

To concretize the proposed method, the Hermite basis is particularly studied in this paper. There is however no principal difference to the method if other orthogonal basis functions are to be employed.

Another approach, investigated in [13], is to employ the series expansion description of the probability density function throughout the whole recursion cycle. The coefficients of the expansion is then propagated over the iterations without turning to the particle description of the PDF:s. This method have shown very plausible results on the parallelizability, but has the drawback that the state have to be confined to a restricted domain, which can be undesirable for some applications. Using the particle filter in combination with the series expansion representation the flexibility of the particle filter, and the plausible parallelization properties of the series expansion method are combined.

The idea of the method suggested here is to fit a series expansion to the particle set and communicate the coefficients among the processing units. In Sec. II and Sec. III, the notation and background material are briefly presented. In Sec. **??** the problem of fitting a series expansion to the particle set is discussed, and an analysis of the fit is provided. The proposed method is explained in Sec. IV , followed in Sec. V by its parallelization. Experiments establishing the estimation accuracy and speedup obtained on a shared-memory multicore processor are described in Sec. VI.

II. Notation

For brevity, the input u_t is omitted in the description of the method. As u_t is a deterministic signal, it is straightforward to incorporate it into the development.

The writing $x \sim p_X(\cdot)$ means that x is a realization of the random variable X with the PDF $p_X(x)$.

III. Background

A. Recursive estimation

Recursive Bayesian estimation solves the estimation problem by computing the PDF $p(x_{t+1}|Y_t)$. Given an initial PDF $p(x_0|Y_{-1})$, the PDF $p(x_{t+1}|Y_t)$ is recursively propagated via the update-prediction equations as

$$\begin{aligned} p(x_t|Y_t) &= \frac{p(y_t|x_t)p(x_t|Y_{t-1})}{p(y_t|Y_{t-1})} \\ p(x_{t+1}|Y_t) &= \int_{\mathbb{R}} p(x_{t+1}|x_t)p(x_t|Y_t)dx_t. \end{aligned}$$

B. The PF algorithm

The PF solves the recursive estimation problem by providing a weighted sample $\{x_{t+1}^{(i)}, w_t^{(i)}\}_{i=1}^N$ from the PDF $p(x_{t+1}|Y_t)$. Then the desired information about x_{t+1}, such as e.g. the minimum mean square error or the maximum likelihood point estimate of the state, can be extracted from the sample. The notation $x_t^{(i)}$ is interpreted as the i-th particle at time step t with $w_t^{(i)}$ as the corresponding weight. The particles are propagated and the weights are updated via the relationships

$$x_{t+1}^{(i)} = f_t(x_t^{(i)}) + v_t^{(i)}, \quad (4)$$

$$w_{t+1}^{(i)} = w_t^{(i)} p(y_{t+1}|x_{t+1}^{(i)}), \quad (5)$$

$i = 1, 2, ..., N$, where $v_t^{(i)} \sim p_v(v)$ and the weights are normalized to sum up to one at the end of the iteration.

The algorithm also contains a third step, the resampling, used to avoid depletion of efficiency in the particle set [14]. Resampling creates a new set of particles $\{x_{t+1}^{(i)}, w_t^{(i)}\}_{i=1}^N$ by making a draw from $p(x_{t+1}|Y_t)$. In general, it is not possible to sample directly from $p(x_{t+1}|Y_t)$, but different methods exist to achieve this goal. A technique used for this purpouse is importance sampling, see e.g. [15], [16]. A weighted sample $\{x_{t+1}^{(i)}, w_t^{(i)}\}_{i=1}^N$ is then obtained by sampling from some easy-to-sample proposal distribution $x_{t+1}^{(i)} \sim \pi(x_{t+1})$, and computing the corresponding weight via

$$w_t^{(i)} \propto \frac{p(x_{t+1}^{(i)}|Y_t)}{\pi(x_{t+1}^{(i)})}. \quad (6)$$

To achieve good efficiency, it is desirable that q is as "similar" to p as possible as well as $\pi(x)$ must satisfy the condition $p(x_{t+1}|Y_t) > 0 \rightarrow \pi(x_{t+1}) > 0$, except at a zero measure of points.

C. Hermite basis functions

Any set of linearly independent basis functions can be utilized in (3). For instance, the Hermite functions, the Laguerre functions and the Fourier functions constitute suitable basis options for the state variable supported in $x_t \in (-\infty, \infty)$, $x_t \in [0, \infty)$, $x_t \in [a, b]$, for some $a, b \in \mathbb{R}$, $a < b$, respectively. To concretize the method, the Hermite basis is particularly studied, and exemplified in the results section.

The Hermitian functions possess plausible properties for PDF approximation, such as exponential decay. It can be shown [17], that if the target function exhibits exponential decay, the coefficients $\{a^{(k)}\}_{k=0}^K$ will also exhibit exponential decay, in the sense that they decrease faster than the reciprocal of any finite order polynomial. Hence a good fit can be expected

for a low truncation order K. The k-th Hermite function is given by

$$H_k(x) = \frac{(-1)^k}{\sqrt{2^k k! \sqrt{\pi}}} e^{x^2/2} \frac{\partial^k}{\partial x^k} e^{-x^2}$$

$k = 0, 1, ...$ For computational purposes, the recurrence relation

$$\begin{aligned} H_0(x) &= \pi^{-1/4} e^{-x^2/2},\ H_1(x) = \sqrt{2} x H_0(x) \\ H_k(x) &= \sqrt{\frac{2}{k}} x H_{k-1}(x) - \sqrt{\frac{k-1}{k}} H_{k-2}(x) \end{aligned}$$

$k = 2, 3, ...$, is often suitable to exploit.

D. Parallelization

The speedup one can expect form a parallel implementation can be analyzed as follows. Assume that a program consists of a portion $p_|$ that must be executed sequentially and a portion $p_{||} = 1 - p_|$ that can be run in parallel. Further assume that there will be M processors used to execute the parallel section and that the parallel execution introduces an overhead of $c(M)$. The speedup s, as a function of the number of processors, is given by

$$s(M) = \frac{t_1}{t_M} = \frac{1}{p_| + \frac{(1-p_|)}{M} + c(M)}. \tag{7}$$

From this expression, it can be seen that a program containing a sequential portion can never achieve a greater speedup than $s(\infty) = \frac{1}{p_|}$ (assuming that $c(M) = 0$). This result is known as Amdahl's law [18]. It is therefore of great importance that there are no large sequentially executed portions in a parallel implementation. The overhead $c(M)$ depends to a large extent of the architecture on which the program is executed and reflects such factors as e.g. amount of communication and number of synchronization points. Obviously, it is also of great importance to restrain $c(M)$ as much as possible to achieve efficient utilization of the computational resources and, thereby, good scalability of the algorithm.

IV. Particle filtering method

In this section the suggested method is presented. Shortly it can be described as follows. Assume that the particle set $S_{t|t} = \{w_t^{(i)}, x_t^{(i)}\}_{i=1}^N$ is given. The particles are propagated via (4) to yield $S_{t+1|t} = \{w_t^{(i)}, x_{t+1}^{(i)}\}_{i=1}^N$. To the weighted set $S_{t+1|t}$ a series expansion is fitted

$$p(x_{t+1}|Y_t) \approx \sum_{k=0}^{K} a_{t+1}^{(k)} \phi_k(x_{t+1}).$$

From the fitted PDF, a new set of particles are drawn by importance sampling. The weights are then updated via (5) to yield $S_{t+1|t+1} = \{w_{t+1}^{(i)}, x_{t+1}^{(i)}\}_{i=1}^N$ and thus complete one iteration of the algorithm. The details of fitting the series expansion and sampling from it are discussed in the following subsections.

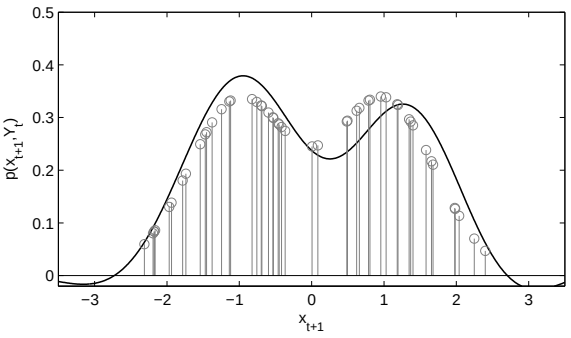

Fig. 1. A set of 50 weighted particles (gray stems) and the fitted series expansion (black solid line) using the first seven Hermite functions.

A. Fitting a series expansion to the particle set

Assume that the particle set $\{x_{t+1}^{(i)}, w_t^{(i)}\}_{i=1}^N$ is given, and it is sought to find an approximation

$$p(x_{t+1}|Y_t) \approx \sum_{k=0}^{K} a_{t+1}^{(k)} \phi_k\left(\frac{x_{t+1} - \mu_{t+1}}{\sigma_{t+1}}\right)$$

for some set of basis functions $\{\phi_k(x)\}_{k=0}^K$ and coefficients $\{a_{t+1}^{(k)}\}_{k=0}^K$. An illustration of a fitted expansion is provided in Fig. 1. The shift and scaling parameters μ_{t+1} and σ_{t+1} are inserted to increase the efficiency of the approximation. For simplicity of notation, the following variables are defined:

$$\begin{aligned} \boldsymbol{\varphi}_t(x) &:= \left[\ \phi_0\left(\tfrac{x-\mu_t}{\sigma_t}\right) \quad \phi_1\left(\tfrac{x-\mu_t}{\sigma_t}\right) \quad \cdots \quad \phi_K\left(\tfrac{x-\mu_t}{\sigma_t}\right)\ \right]^T \\ \mathbf{a}_t &:= \left[\ a_t^{(0)} \quad a_t^{(1)} \quad \cdots \quad a_t^{(K)}\ \right]^T. \end{aligned}$$

Making a least-squares fit, the criterion to minimize is

$$V(\mathbf{a}_{t+1}) = \int_{\mathbb{R}} [p(x_{t+1}|Y_t) - \mathbf{a}_{t+1}^T \boldsymbol{\varphi}_{t+1}(x_{t+1})]^2 dx_{t+1}.$$

By solving the equation

$$\frac{\partial V}{\partial \mathbf{a}_{t+1}} = 0,$$

it is found that the unique solution (since $\{\phi_k(x)\}_{k=0}^K$ are linearly independent) is given by

$$\mathbf{a}_{t+1} = \mathbf{A}^{-1} \mathbf{b}_{t+1},$$

where the positive definite Gramian matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$ and the vector $\mathbf{b}_{t+1} \in \mathbb{R}^K$ are

$$\begin{aligned} \mathbf{A} &= \int_{\mathbb{R}} \boldsymbol{\varphi}(x) \boldsymbol{\varphi}(x)^T dx, \\ \mathbf{b}_{t+1} &= \int_{\mathbb{R}} p(x_{t+1}|Y_t) \boldsymbol{\varphi}_{t+1}(x_{t+1}) dx_{t+1}. \end{aligned}$$

In the expression above, $\boldsymbol{\varphi}$ is the same as $\boldsymbol{\varphi}_t$, but taken with $\mu = 0$ and $\sigma = 1$.

The elements of $\mathbf{A}$ are independent of data and can be calculated offline from the known functions $\{\phi_k(x)\}_{k=0}^K$, to any given accuracy. The unknown vector $\mathbf{b}_{t+1}$ can be estimated from the particle set $\{x_{t+1}^{(i)}, w_t^{(i)}\}_{i=1}^N$ as

$$\mathbf{b}_{t+1} = \int_{\mathbb{R}} p(x_{t+1}|Y_t)\boldsymbol{\varphi}_{t+1}(x_{t+1})dx \quad (8)$$
$$= E[\boldsymbol{\varphi}_{t+1}(x_{t+1})] \quad (9)$$
$$\approx \sum_{i=1}^{N} w_t^{(i)}\boldsymbol{\varphi}_{t+1}(x_{t+1}^{(i)}). \quad (10)$$

It is well know that (10) is a consistent estimator of (9), and hence, by the bijective relation between $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a}$ is consistently estimated from

$$\hat{\mathbf{a}}_{t+1} = \mathbf{A}^{-1}\hat{\mathbf{b}}_{t+1}.$$

Note that if the Hermite basis is used, i.e. $\phi_k(x) = H_k(x)$, then $\mathbf{A} = \mathbf{I}_K$ by the orthonormality of the basis functions, where $\mathbf{I}_K$ is an identity matrix of order K. Further, the location and scaling parameters μ and σ are naturally chosen as the mean and standard deviation of the particle set, as if the distribution were truly Gaussian, a perfect fit would be achieved as $N \to \infty$.

B. Resampling from the fitted series expansion

When resampling, the particles should be drawn from a proposal distribution. For the Hermite functions, the first basis function is a scaled Gaussian PDF and it is hence natural to sample from a Gaussian distribution. The resampling is then performed, via importance sampling, from the proposal, i.e.

$$x_{t+1}^{(i)} \sim \gamma_{t+1}(x),$$

where $\gamma_t(x) := \gamma(x, \mu_t, \sigma_t)$, and computing the weights from

$$w_{t+1}^{(i)} = \frac{\hat{p}(x_{t+1}|Y_y)}{\gamma(x_{t+1}^{(i)})} = \frac{\mathbf{a}_{t+1}^T\boldsymbol{\varphi}_{t+1}(x_{t+1}^{(i)})}{\gamma(x_{t+1}^{(i)})}.$$

Note that during the recursion, the PDF is propagated as

$$p(x_t|Y_t) \to p(x_{t+1}|Y_t) \to p(x_{t+1}|Y_{t+1})$$

via the prediction and update equations (4), (5). The resampling can be carried out at any step of the recursion. Within the proposed formulation, the resampling is specified to occur at the second step, i.e. by making a draw from $p(x_{t+1}|Y_t)$. This is motivated by the fact that the PDF $p(x_{t+1}|Y_t)$ is typically smoother than $p(x_t|Y_t)$ and, therefore, is more suitable to approximate with a series expansion of low order. This is can be intuitively understood by that $p(x_{t+1}|Y_t)$ is a prediction and, hence, subject to greater uncertainty (higher degree of smoothness) than $p(x_t|Y_t)$, or more technically by that $p(x_{t+1}|Y_t)$ is the outcome of a convolution of two PDFs, whereas $p(x_t|Y_t)$ is a product.

C. Summarizing the method

The algorithm comprised of the prediction (P), resampling (R) and update (U) steps is summarized in Alg. 1.

Algorithm 1 Particle filtering algorithm

(P) $x_{t+1}^{(i)} = f_t(x_t^{(i)}) + v_t^{(i)}$, where $v_t^{(i)} \sim p_v(v)$

(R) $\hat{\mathbf{a}}_{t+1} = \frac{1}{N}\mathbf{A}^{-1}\sum_{i=1}^{N}\boldsymbol{\varphi}_{t+1}(x_{t+1}^{(i)})$
$x_{t+1}^{(i)} \sim \gamma_t(x)$
$w_t^{(i)} = \hat{\mathbf{a}}_{t+1}^T\boldsymbol{\varphi}_{t+1}(x_{t+1}^{(i)})/\gamma_{t+1}(x_{t+1}^{(i)})$

(U) $w_t^{(i)} = w_{t-1}^{(i)}p(y_t|x_t^{(i)})$

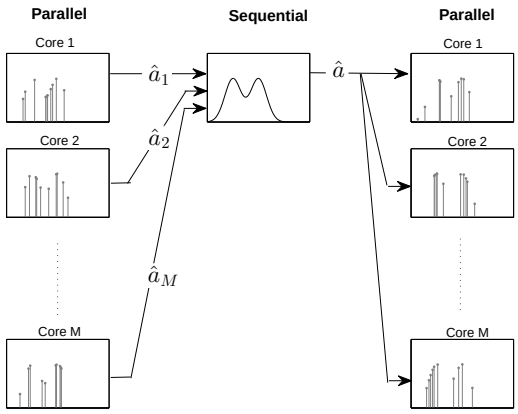

Fig. 2. Illustration of the work flow for parallel execution of the algorithm.

V. Parallelization

The method is by design straightforward to parallelize. The execution steps are illustrated in Fig. 2. The key concept is as described, to propagate and update the particle set locally at each processing unit (first step). From the local particle set each processor will then compute the local estimate $\hat{\mathbf{a}}^{(m)}$ of $\mathbf{a}$, which is communicated to one processor that forms the global estimate $\hat{\mathbf{a}}$ (second step). With the aid of the global estimate of $\mathbf{a}$, each processing unit resamples, i.e. creates a new local set of particles by making a draw from $\hat{p}(x_{t+1}|Y_t) = \mathbf{a}_{t+1}^T\boldsymbol{\varphi}_{t+1}(x)$, (third step). Pseudo code for a parallelization is given in Alg. 2, where N_m, $m = 1, 2, ..M$ are disjoint subsets $\{1, 2, .., N\}$ of cardinality N/M (assumed to be integer).

Algorithm 2 Parallel implementation

Parallel (processor m do for $i \in N_m$)

(R) $x_t^{(i)} \sim \gamma_t(x)$
$w_{t-1}^{(i)} = \hat{\mathbf{a}}_t^T\boldsymbol{\varphi}_t(x_t^{(i)})/\gamma_t(x_t^{(i)})$

(U) $w_t^{(i)} = w_{t-1}^{(i)}p(y_t|x_t^{(i)})$

(P) $x_{t+1}^{(i)} = f_t(x_t^{(i)}) + v_t^{(i)}$, where $v_t^{(i)} \sim p_v(v)$
$\hat{\mathbf{a}}_{t+1}^{(m)} = \hat{\mathbf{a}}_{t+1}^{(m)} + \boldsymbol{\varphi}_{t+1}(x_{t+1}^{(i)})$

Sequentially (one processor)

$\hat{\mathbf{a}}_{t+1} = \frac{1}{M}\mathbf{A}^{-1}\sum_{m=1}^{M}\hat{\mathbf{a}}_{t+1}^{(m)}$

The location and spread parameters μ and σ should be computed from the mean and covariance of the global particle set. This can be simply done in parallel by letting

processor m compute

$$\begin{aligned}\mu_{t+1}^{(m)} &= \sum_{i\in N(m)} w_t^{(i,m)} x_{t+1}^{(i,m)},\\ \rho_{t+1}^{(m)} &= \sum_{i\in N(m)} w_t^{(i,m)} [x_{t+1}^{(i,m)}]^2\end{aligned}$$

from the local particle set and forming the global estimates in the sequential section as

$$\begin{aligned}\mu_{t+1} &= \frac{1}{M}\sum_{m=1}^{M} \mu_{t+1}^{(m)},\\ \sigma_{t+1}^2 &= E[x_{t+1}^2] - E[x_{t+1}]^2 = \frac{1}{M}\sum_{m=1}^{M} \rho^{(m)} - \mu_{t+1}^2.\end{aligned}$$

A. Parallelization properties analysis

For the suggested parallelization scheme, the majority of the work is carried out in the parallel section, and only a small amount of communication and sequential processing is required, which property is the key characteristic for a scalable parallelization, as discussed in Sec. III. Assume that the system is linear and that the process and measurement noise are uniformly distributed, which presents a worst case scenario in the sense that it gives the least amount of parallel executed floating point instructions (flops) relative the sequentially executed flops. The flops yielding the sequential work $q_|$ and the parallel work $q_{||}$ can be counted as

$$q_|(M,K) \approx (M-1)(K+1), \quad (11)$$

$$q_{||}(N,K) \approx (2+2f_r+f_e+4K)N + 2(K+1), \quad (12)$$

where f_r and f_e are the number of flops required to generate a random number and to evaluate the exponential function, respectively. The communication required is to transfer the local estimates $\mathbf{a}^{(m)}$, $\mu^{(m)}$, and $\rho^{(m)}$, $m = 1, 2, ..., M$ which results in a total communication of

$$\kappa(K,M) = (K+3)(M-1)$$

elements per iteration.

VI. Experiments

The main goal the proposed method is to achieve parallelizability of the PF. As pointed out before, a method that can compete with the parallel properties of the method is the GPF. For comparison, the method described above is hence tested against the GPF, at an example that highlights the benefits of the former of not being restricted to a Gaussian posterior. The estimation accuracy and the parallelizability are investigated in the following subsections. Merely for brievity, the proposed method will be referred to as the Hermitian Particle Filter (HPF) in this section.

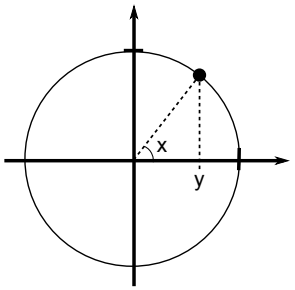

Fig. 3. The system, with state x and measurement y.

A. System model

To illustrate the method the simple, but non-linear system (illustrated in Fig. 3)

$$\begin{aligned}x_{t+1} &= x_t + u_t + v_t\\ y_t &= \cos x_t + e_t\end{aligned}$$

is studied, where $x_t \in \mathbb{R}$, is the state $y_t \in \mathbb{R}$ is the measurement,

$$u_t = \begin{cases}0 & \text{if } t \le 50\\ 0.1 & \text{if } t > 50\end{cases}$$

is the input signal, and $v_t \in \mathbb{R}$ and $e_t \in \mathbb{R}$ are mutually independent, normally distributed, white, zero mean noise sequences, with variances $\sigma_v^2 = 0.1$ and $\sigma_e^2 = 0.5$ respectively. The system was simulated for $t = 0, 1, 2..., T$, where $T = 100$.

B. Estimation accuracy

In Fig. 4 and Fig. 5, the true PDF $p(x_t|Y_t)$ and the approximated PDFs $p^{HPF}(x_{t+1}|Y_t)$ and $p^{GPF}(x_{t+1}|Y_t)$ obtained from the HPF (with $K = 7$) and GPF, respectively, are shown for time instants $t = 40$ and $t = 60$. The "true" PDF has been obtained by executing a regular bootstrapping particle filter with 10^6 particles and applying a kernel density approximation method [19] to the obtained particle set.

During the first fifty time steps $0 \le t \le 50$, the input signal is equal to zero, and the object is just moving randomly, driven by the process noise. From the measurements it is then not possible to determine if the object is moving clockwise or anti-clockwise, which results in a multimodal PDF. Obviously the GPF can not capture this multimodality and therefore gives a poor estimate of the true PDF. Starting from $t > 50$, the input signal becomes equal 0.1 and drives the object clockwise around the circle. Then it becomes possible to rule out one of the modes as unlikely. However, even though the true PDF becomes more unimodal and fairly Gaussian, the GPF fails to capture the true mean due to accumulated errors from the previous time steps.

C. Execution time and speedup

To evaluate the performance of the method in terms of the speedup obtained when executed on a shared memory multicore computer, the method was implemented in c++

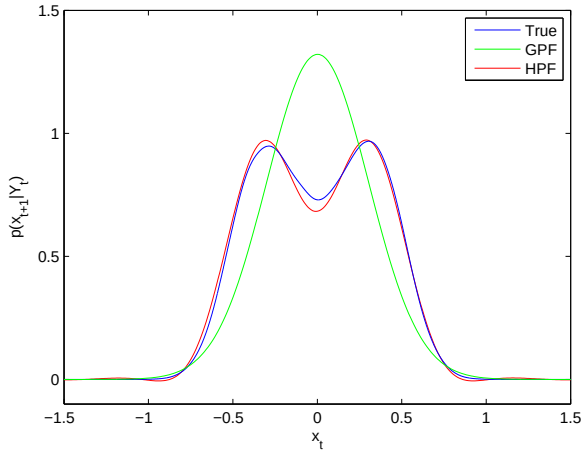

Fig. 4. The true PDF $p(x_{t+1}|Y_t)$ and estimated PDFs $p^{GPF}(x_{t+1}|Y_t)$ and $p^{HPF}(x_{t+1}|Y_t)$ from the Gaussian and Hermitian particle filters respectively at time step $t = 40$.

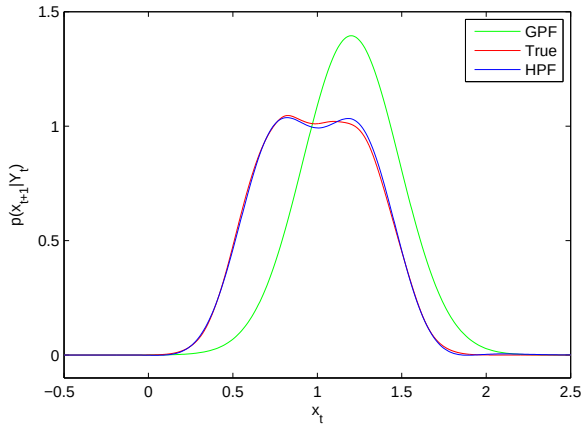

Fig. 5. The true PDF $p(x_{t+1}|Y_t)$ and estimated PDFs $p^{GPF}(x_{t+1}|Y_t)$, $p^{HPF}(x_{t+1}|Y_t)$ from the Gaussian and Hermitian particle filters respectively at time step $t = 60$.

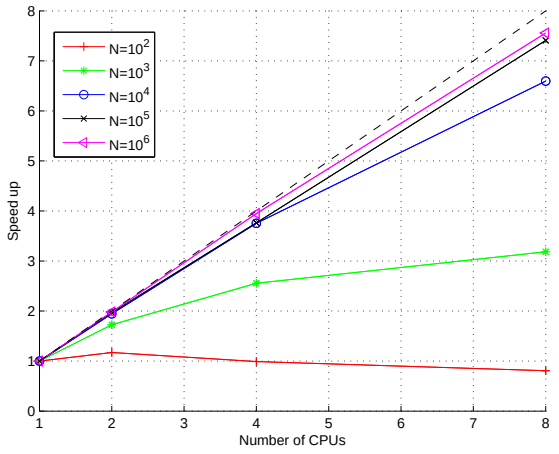

Fig. 6. Speedup curves for execution on a shared-memory multicore processor.

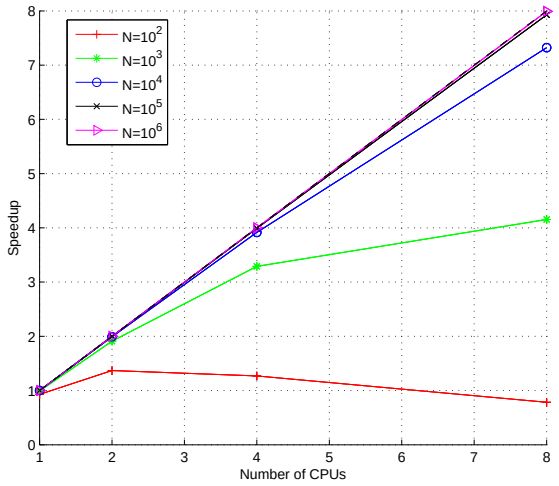

Fig. 7. Theoretical speedup curves computed from (13).

and executed on a AMD Opteron 6220 processor (3.3 GHz, 8 cores, 16 MB cache). Compilation was performed using the pgi compiler, with full optimization for execution speed. OpenMP [20] was used for parallelization. In Fig. 6 the speedup is shown for different problem sizes.

On the machine under consideration, empirical testing shows that $f_r \approx 10$, $f_e \approx 40$, are reasonable approximations for the terms in Eq. (12). It takes wo hundred cpu cycles to communicate an element to the RAM and about 1000 cpu cycles for overhead such as synchronization, thread startup etc. This results in the overhead term of $c(M, K) = 200\kappa(K, M) + 1000$. Inserting this into (7) yields a theoretical estimate of the speedup for $K = 7$

$$s(M, N) = M \frac{4M + 65N + 4}{1004M^2 - 504M + 65N + 8}. \qquad (13)$$

The theoretical speedup curves (13) are plotted in Fig. 7. Though being a bit optimistic, the curves resemble the experimentally obtained ones quite well. Exact figures can of course not be expected from (13), but it provides a guideline for the speedup to expect for a given problem size and number of processors employed.

The obtained speedup values compare well with the ones obtained for the GPF in [21]. For $N \geq 10^4$, close-to-linear speedup in the number of cores used is reached. For $N = 100$, the benefit of using parallelization is low, and actually a slow down can be observed for $M > 4$. This is due to the fact that the overhead $c(M)$ constitutes a disproportionally large part of the execution time. Using more than 10^4 particles is actually unnecessary in this particular example, as the estimation accuracy is not increasing above this value. However, for the speedup test, these number of particles are relevant. The speedup curves obtained are for hardware that executes on a relatively high clock frequency (3.3 GHz). In low power applications, such as e.g. communication, the

TABLE I
SINGLE CORE EXECUTION TIME FOR ONE ITERATION

N	10^2	10^3	10^4	10^5	10^6
T [s]	0.004	0.0042	0.001	0.12	0.52

clock frequencies are typically much lower, and hence a better scalability can be expected for smaller problem sizes. The small amount of communication makes the approach suitable for computer cluster execution which can be relevant for large problem sizes.

REFERENCES

[1] S. M. M. A. Rosen, O., "Nonlinear estimation of a parsimonious wiener model for the neuromuscular blockade in closed-loop anesthesia," *IFAC world congress*.

[2] M. Chitchian, A. S. van Amesfoort, A. Simonetto, T. Keviczky, and H. J. Sips, "Particle filters on multi-core processors," *Dept. Comput. Sci., Delft Univ. Technology, Delft, The Netherlands, Tech. Rep. PDS-2012-001,(Feb. 2012).*, 2012.

[3] K. Par and O. Tosun, "Parallelization of particle filter based localization and map matching algorithms on multicore/manycore architectures," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, June 2011, pp. 820–826.

[4] O. Rosen and A. Medvedev, "Efficient parallel implementation of state estimation algorithms on multicore platforms," *Control Systems Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –14, 2011.

[5] A. S. Bashi, V. P. Jilkov, X. R. Li, and H. Chen, "Distributed implementations of particle filters," in *Proc. of the Sixth Int. Conf. of Information Fusion*, 2003, pp. 1164–1171.

[6] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Processing*, vol. 53, pp. 2442–2450, 2004.

[7] J. H. Kotecha, P. M. Djuric, and S. Member, "Gaussian particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2592–2601, 2003.

[8] S. C. Schwartz, "Estimation of probability density by an orthogonal series," *The Annals of Mathematical Statistics*, vol. 38, no. 4, pp. 1261–1265, 08 1967.

[9] B. Silverman, *Density Estimation for Statistics and Data Analysis*, ser. Monographs on Statistics and Applied Probability. Taylor & Francis, 1986.

[10] G. S. Watson, "Density estimation by orthogonal series," *The Annals of Mathematical Statistics*, vol. 40, no. 4, pp. pp. 1496–1498, 1969. [Online]. Available: http://www.jstor.org/stable/2239615

[11] J. H. Kotecha and P. Djuric, "Gaussian sum particle filtering," *Signal Processing, IEEE Transactions on*, vol. 51, no. 10, pp. 2602–2612, Oct 2003.

[12] B. Anderson and J. Moore, *Optimal Filtering*, ser. Dover Books on Electrical Engineering. Dover Publications, 2012.

[13] O. Rosen and Medvedev, "Parallel recursive estimation, based on orthognal series expansions," in *American Control Conference (ACC), 2040 IEEE International Conference on*, Sep. 2010, pp. 440 –445.

[14] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, ser. Artech House Radar Library. Artech House, 2004.

[15] C. Robert and G. Casella, *Monte Carlo Statistical Methods*, ser. Springer Texts in Statistics. Springer, 2004.

[16] J. S. Liu, *Monte Carlo strategies in scientific computing*. springer, 2008.

[17] J. P. Boyd, "Asymptotic coefficients of hermite function series," *Journal of Computational Physics*, vol. 54, no. 3, pp. 382–410, 1984.

[18] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *AFIPS '67 (Spring): Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. New York, NY, USA: ACM, 1967, pp. 483–485.

[19] B. Silverman, *Density Estimation for Statistics and Data Analysis*, ser. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1986.

[20] "Open mp," http://www.cs.virginia.edu/stream/, Aug. 2010.

[21] O. Rosen and A. Medvedev, "Efficient parallel implementation of state estimation algorithms on multicore platforms," *Control Systems Technology, IEEE Transactions on*, vol. 21, no. 1, pp. 107 –120, jan. 2013.