
Proceedings Paper:
Single-Liertz, T, Kim, J and Richardson, R (2015) Nonlinear projection filter with parallel algorithm and parallel sensors. In: 2015 54th IEEE Conference on Decision and Control (CDC). 54th Conference on Decision and Control, 15-18 Dec 2015, Osaka. IEEE, pp. 2432-2437. ISBN: 978-1-4799-7884-7.

https://doi.org/10.1109/CDC.2015.7402572

Nonlinear projection filter with parallel algorithm and parallel sensors *

Tama Single-Liertz[1], Jongrae Kim[1], and Robert Richardson[1]

***Abstract*— Over the past few decades, the computational power has been increasing rapidly. With advances of the parallel computation architectures it provides new opportunities for solving the optimal estimation problem in real-time. In addition, sensor miniaturization technology enables us to acquire multiple measurements at low cost. Kolmogorov's forward equation is the governing equation of the nonlinear estimation problem. The nonlinear projection filter presented in the late 90's is an *almost* exact solution of the nonlinear estimation problem, which solves the governing equation using Galerkin's method. The filter requires high-dimensional integration in several steps and the complexity of the filter increases exponentially with the dimension of systems. The current parallel computation speed with the usage of many sensors at the same time make it feasible to implement the filter efficiently for practical systems with some mild dimension sizes. On-line or off-line multi-dimensional integration is to be performed over the parallel computation using the Monte-Carlo integration method and random samples for the state update are obtained more efficiently based on the multiple sensor measurements. A few simplifications of the filter are also derived to reduce the computational cost. The methods are verified with two numerical examples and one experimental example.**

I. Introduction

Optimal estimation has been one of the most important research topics in control engineering since the optimal estimator for linear time-invariant systems, Kalman filter, was developed [1]. The filter has been successfully applied to many dynamical systems. The major successful applications of Kalman filter are, on the other hand, for nonlinear systems in Aerospace Engineering such as the navigation problem of spacecraft for the Apollo mission [2] and attitude estimation problem for satellite [3]. The derivation of extended Kalman filter for nonlinear systems was the beginning of a plethora applications for the filter. As the extended Kalman filter relies on the accuracy of the first-order derivatives of nonlinear systems, the estimation could have some convergence issues depending on the initialisation error. In order to circumvent the divergence problem, the unscented Kalman filter was proposed [4]. Using a set of samples during the state propagation step, the unscented Kalman filter achieves the accuracy up to the 3rd-order in Taylor series expansion [5].

These filters are not, however, nonlinear filters and they suffer when the measurement are non-Gaussian and/or nonlinearities in the system becomes significant. To tackle the nonlinear estimation problem directly, solving the Chapman-Kolmogorov equation and exploitations of Bayes' rule are considered [6]. These are the main focus of particle filters for nonlinear and non-gaussian systems. Particle filters estimate the posterior probability density function using a set of samples in the state space. In a special case, the Chapman-Kolmogorov equation becomes the Kolmogorov forward equation, also known as the Fokker-Plank equation [7]. The nonlinear projection filter presented in [8] solves the equation by assuming the solution as the sum of basis functions. The filter provides an elegant form of the solution for the Fokker-Plank equation. It requires multi-dimensional integration in several steps of the filter implementation and the number of basis function increases very fast as the system dimension increases. These are the main obstacles to any practical usage up to now. The most recent application was a 2D target-tracking problem in [9].

To resolve the practicality of the filter, new implementation methods are to be proposed, which are based on *parallel computation and parallel sensors*. The parallel computational power has been increased tremendously by multi-core technology in CPU (Central Processing Unit) and GPU (Graphical Processing Unit) [10], and FPGA (Field Programmable Gate Array). These are the perfect platform to implement Monte-Carlo integration, which is naturally parallel. In addition, sensor technology has been advanced rapidly in terms of the miniaturising the size of the sensors and the minimised power consumption. This enables us to accumulate many sensors and use them at the same time, i.e. massively parallel sensor usage.

This paper is organised as follows: firstly, a summary of the nonlinear projection filter with compact expressions is presented; secondly, the filter implementation is improved using Monte-Carlo integration and multiple sensors; thirdly, two numerical and one experimental examples are presented, demonstrating the effectiveness of the proposed algorithms; finally conclusions and future works are presented.

II. Nonlinear projection filter

A nonlinear stochastic differential equation is given by

$$d\mathbf{x} = \mathbf{f}(\mathbf{x})dt + G(\mathbf{x})d\beta \tag{1}$$

for $t \geq t_0$, where $\mathbf{x}$ is an n-dimensional state vector in $\mathbb{R}^n$, which is the n-dimensional real number space, n is a positive integer, $\beta(t)$ is a q-dimensional Brownian motion in $\mathbb{R}^q$, whose covariance matrix, i.e. $\mathrm{E}(\beta\beta^T)$, is equal to $Q(t)dt$, q is a positive integer, $E(\cdot)$ is the expectation, $(\cdot)^T$ is the transpose, $\mathbf{f}(\cdot)$ is an n-dimensional nonlinear function,

*This work was supported by School of Mechanical Engineering, University of Leeds, Leeds LS2 9JT UK.

[1]Tama Single-Liertz, Jongrae Kim and Robert Richardson are with Institute of Design, Robotics & Optimisation (iDRO), School of Mechanical Engineering University of Leeds, Leeds LS2 9JT, UK. mntrks,menjkim,r.c.richardson@leeds.ac.uk

and $G(\cdot)$ is an $n \times q$ matrix. In addition, noisy discrete measurements are obtained from the following nonlinear observation:

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{2}$$

for $k \geq 1$, where $\mathbf{y}_k$ is in $\mathbb{R}^m$, m is a positive integer, $\mathbf{v}_k$ is a white Gaussian noise independent of $d\beta$, whose covariance is R_k, and $\mathbf{h}(\cdot)$ is the m-dimensional measurement function.

Probability density function (pdf) conditioned by the measurement is given by

$$p(t,\mathbf{x}|\ Y_k) = \frac{p(t,\mathbf{x},Y_k)}{p(Y_k)},$$

where Y_k is the collection of all measurement up to $t_k \leq t$, i.e. $Y_k := \{\mathbf{y}_k|\ t_k \leq t\}$. $p(t,\mathbf{x}|\ Y_k)$ includes all possibly required information conditioned by all available measurements. The conditional pdf follows Kolmogorov's forward equation:

$$\frac{\partial p}{\partial t} = -\sum_{i=1}^{n} \frac{\partial (p f_i)}{\partial x_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 \left[p\left(GQG^T\right)_{i,j}\right]}{\partial x_i \partial x_j} \tag{3}$$

where x_i and f_i are i-th element of $\mathbf{x}$ and $\mathbf{f}(\mathbf{x})$, respectively, $(GQG^T)_{i,j}$ is i-th row and j-th column element of the matrix, GQG^T, and the initial condition is given by $p(t_0,\mathbf{x})$. Once the pdf is obtained, the first moment, for example, can be calculated as follows:

$$E(\mathbf{x}) = \int_\Omega \mathbf{x}\ p(\mathbf{x},t|\ Y_k) d\mathbf{x},$$

where Ω is a closed bounded subset of $\mathbb{R}^n$. Unlike in the Kalman filter, which tracks only first two moments, any moments can be calculated from the pdf.

Solving the above partial differential equation, (3), is computationally demanding. The closed form solution of (3) is not available in general except some special cases. The nonlinear projection filter proposed in [8] is a method to solve (3) using Galerkin's approximation, which is one of the common methods to solve partial differential equations [11]. Assume that the solution is a linear combination of basis functions, $\phi_\ell(\mathbf{x})$, for $\ell = 1,2,\ldots,N$, as follows:

$$p(t,\mathbf{x}|\ Y_k) \approx p_N(t,\mathbf{x}|\ Y_k) = \sum_{\ell=1}^{N} c_\ell(t)\phi_\ell(\mathbf{x}), \tag{4}$$

where the basis functions are orthogonal, i.e.,

$$\int_\Omega \phi_i(\mathbf{x})\ \phi_j(\mathbf{x}) d\mathbf{x} = \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{for } i \neq j \end{cases} \tag{5}$$

for $i,j = 1,2,3,\ldots,N-1,N$.

In the following, *propagation* and *update* parts of nonlinear projection filter are summarised. More details about nonlinear projection filter derivation can be found in [8].

A. Propagation

Substituting (4) into (3), projecting onto ϕ_q and integrating over Ω provide

$$\int_\Omega \left\{ \frac{\partial p_N}{\partial t} + \sum_{i=1}^{n} \frac{\partial (p_N f_i)}{\partial x_i} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 \left[p_N\left(GQG^T\right)_{i,j}\right]}{\partial x_i \partial x_j} \right\} \phi_q d\mathbf{x} = 0 \tag{6}$$

for $q = 1,2,\ldots,N$, which is the projection condition that $\mathbf{c}(t)$ must satisfy, where $\mathbf{c}(t) = [c_1(t),\ c_2(t),\ \ldots,\ c_N(t)]^T$. From the projection equation, (6), the differential equation for $\mathbf{c}(t)$ is obtained as follows:

$$\dot{\mathbf{c}}(t) = (\mathbf{A}_1 + \mathbf{A}_2)\mathbf{c}(t), \tag{7}$$

where the initial condition is given by

$$\mathbf{c}(t_0) = \int_\Omega p(t_0,\mathbf{x})\phi(\mathbf{x}) d\mathbf{x},$$

$p(t_0,\mathbf{x})$ is the pdf of the initial state $\mathbf{x}(t_0)$, $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}),\ \phi_2(\mathbf{x}),\ \ldots,\ \phi_N(\mathbf{x})]^T$, and $\mathbf{A}_1$ and $\mathbf{A}_2$ are the matrices, whose i-th row and j-th column element is given by

$$[\mathbf{A}_1]_{i,j} = -\sum_{k=1}^{n} \int_\Omega \frac{\partial [\phi_j f_k]}{\partial x_k} \phi_i d\mathbf{x},$$

$$[\mathbf{A}_2]_{i,j} = \frac{1}{2}\sum_{k=1}^{n}\sum_{\ell=1}^{n} \int_\Omega \frac{\partial^2 \left[\phi_j \left(GQG^T\right)_{k,\ell}\right]}{\partial x_k \partial x_\ell} \phi_i d\mathbf{x}.$$

$\mathbf{A}_1$ can be written in a compact form as follows:

Lemma 2.1:

$$\mathbf{A}_1 = -\int_\Omega \phi \left[\mathbf{f}^T (\nabla\phi)^T + \left(\nabla^T \mathbf{f}\right)\phi^T\right] d\mathbf{x}, \tag{9}$$

where

$$\nabla := \left[\frac{\partial}{\partial x_1},\quad \frac{\partial}{\partial x_2},\quad \ldots,\quad \frac{\partial}{\partial x_n}\right]^T, \tag{10}$$

and $\nabla\phi$ is $N \times n$ Jacobian matrix, whose i-th row and j-th column element is given by

$$[\nabla\phi]_{i,j} = \frac{\partial \phi_i}{\partial x_j} \tag{11}$$

for $i = 1,2,\ldots,N$, and $j = 1,2,\ldots,n$.

Proof: As the summation and integration are commutable,

$$[\mathbf{A}_1]_{i,j} = -\int_\Omega \sum_{k=1}^{n} \frac{\partial [\phi_j f_k]}{\partial x_k} \phi_i d\mathbf{x}$$

and the summation is written in vector dot products,

$$[\mathbf{A}_1]_{i,j} = -\int_\Omega \left[\phi_i\ \left(\mathbf{f}^T\ \nabla\phi_j\right) + \left(\nabla^T \mathbf{f}\right)\phi_i\phi_j\right] d\mathbf{x}.$$

And, $\mathbf{A}_1$ is constructed as

$$\mathbf{A}_1 = -\int_\Omega \left\{\phi\ \mathbf{f}^T \left[\nabla\phi_1,\quad \nabla\phi_2,\quad \ldots,\quad \nabla\phi_N\right] + \left(\nabla^T \mathbf{f}\right)\phi\phi^T\right\} d\mathbf{x}.$$

■

And, $\mathbf{A}_2$ can be also written in a compact form as follows:

Lemma 2.2:

$$\mathbf{A}_2 = \mathbf{A}_{21} + \mathbf{A}_{22}, \tag{12}$$

where

$$\mathbf{A}_{21} = \frac{1}{2}\int_\Omega \phi\left(\lambda \circ \phi^T\right) d\mathbf{x} \tag{13a}$$

$$\mathbf{A}_{22} = \frac{1}{2}\int_\Omega \gamma\ \phi\phi^T d\mathbf{x}, \tag{13b}$$

λ is an operator defined by $\lambda := \mathbf{1}^T\left(GQG^T\right) \odot H\mathbf{1}$, the operator applies to each term of ϕ in $\lambda \circ \phi$, and $\gamma := \mathbf{1}^T H \odot \left(GQG^T\right)\mathbf{1}$, H is the Hessian matrix equal to $\nabla\nabla^T$, $\odot$ is the Hardamard product, i.e., an element-wise multiplication of two same dimensional matrices, and $\mathbf{1}$ is the column vector whose elements are all 1 with an appropriate dimension.

Proof: As the summation and integration are commutable,

$$[\mathbf{A}_2]_{i,j} = \frac{1}{2}\int_\Omega \sum_{k=1}^{n}\sum_{\ell=1}^{n} \frac{\partial^2\left[\phi_j\left(GQG^T\right)_{k,\ell}\right]}{\partial x_k \partial x_\ell}\phi_i d\mathbf{x},$$

where

$$\sum_{k=1}^{n}\sum_{\ell=1}^{n} \frac{\partial^2\left[\phi_j\left(GQG^T\right)_{k,\ell}\right]}{\partial x_i \partial x_j} = \mathbf{1}^T(\nabla\nabla^T) \odot (\phi_j GQG^T)\mathbf{1}.$$

In addition,

$$\begin{aligned}
&\mathbf{1}^T(\nabla\nabla^T) \odot (\phi_j GQG^T)\mathbf{1} \\
&\quad = \mathbf{1}^T\left\{(H\phi_j) \odot \left(GQG^T\right) + \left[H \odot \left(GQG^T\right)\right]\phi_j\right\}\mathbf{1} \\
&\quad = \mathbf{1}^T\left[\left(GQG^T\right) \odot H\right]\phi_j\mathbf{1} + \mathbf{1}^T\left[H \odot \left(GQG^T\right)\right]\phi_j\mathbf{1} \\
&\quad = \lambda \circ \phi_j + \gamma\phi_j
\end{aligned}$$

Therefore,

$$\mathbf{A}_2 = \frac{1}{2}\int_\Omega \phi(\lambda \circ \phi^T) + \phi\gamma\phi^T d\mathbf{x}. \qquad \blacksquare$$

Remark 2.3: $\mathbf{A}_{22}$ is symmetric and it is equal to zero if G and Q are constant matrices.

Remark 2.4: $\mathbf{f}$ and G in (1) are not function of time, $\mathbf{A}_1$ and $\mathbf{A}_2$ in (9) and (12) are constant matrices and they are calculated off-line and stored a priori.

B. Update

Whenever the measurement is available, the conditional probability density function is updated followed by Bayes' rule [6]:

$$p(t_k^+, \mathbf{x}|Y_k) = \frac{p(\mathbf{y}_k|\mathbf{x})\ p(t_k^-, \mathbf{x}|Y_{k-1})}{\int_\Omega p(\mathbf{y}_k|\xi)p(t_k^-, \xi|Y_{k-1})d\xi}, \tag{14}$$

where t_k^+ and t_k^- are the instances just after and before the k-th measurement is available, respectively, $p(\mathbf{y}_k|\mathbf{x})$ is the sensor model, which could be given by the normal distribution,

$$p(\mathbf{y}_k|\mathbf{x}) = \frac{e^{-\frac{1}{2}[\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k)]^T R_k^{-1}[\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k)]}}{\sqrt{(2\pi)^m |R_k|}},$$

$p(t_k^-, \mathbf{x}|Y_{k-1})$ is the pdf propagated from the previous step. The sensor model does not necessarily have the Gaussian distribution. Other distribution can be easily incorporated with the update equation. Substituting $p(t, \mathbf{x}|Y_k) \approx p_N(t, \mathbf{x}|Y_k)$ into (14) the following update equation for $\mathbf{c}(t)$ is obtained [8]:

$$\mathbf{c}(t_k^+) = \left[Y(\mathbf{y}_k)\ \mathbf{c}(t_k^-)\right] / \left[\upsilon(\mathbf{y}_k)^T \mathbf{c}(t_k^-)\right] \tag{15}$$

where $Y(\mathbf{y}_k) = \int_\Omega p(\mathbf{y}_k|\mathbf{x})\phi\phi^T d\mathbf{x}$, $\upsilon(\mathbf{y}_k) = \int_\Omega p(\mathbf{y}_k|\mathbf{x})\phi d\mathbf{x}$.

III. Filter Implementation

A discrete implementation of (7) can be done by

$$\mathbf{c}(t_{k+1}) = \Phi(t_{k+1}, t_k)\mathbf{c}(t_k), \tag{16}$$

where $\Phi(t_{k+1}, t_k)$ is the state transition matrix given by

$$\Phi(t_{k+1}, t_k) = e^{(\mathbf{A}_1 + \mathbf{A}_2)(t_{k+1} - t_k)}. \tag{17}$$

In [8], it was recommended to use Discrete Cosine Transformations (DCT) algorithm to perform the integrals to obtain $\mathbf{A}_1$ and $\mathbf{A}_2$. Although there is some research in multi-dimensional implementation of DCT [12], [13], to the best knowledge of the authors, the DCT algorithm implementation for general multi-dimensional cases is not available. Instead, Monte-Carlo integration is proposed for all multi-dimensional integrations required. Monte-Carlo integration is very simple and powerful for integrating complex functions and it does not cause exponential increase of the computational cost from *the curse of dimensionality*.

Algorithm 3.1: (Calculation $\mathbf{A}_1$ and $\mathbf{A}_2$)

1) Set the number of samples, N_s, the tolerance, ε_A, the iteration number, $k = 1$, and $\mathbf{A}_1^0$ and $\mathbf{A}_2^0$ to be zero matrices.
2) Generate N_s random samples of $\mathbf{x}^i$ uniformly distributed in Ω, where $i = 1, 2, \ldots, N_\mathrm{s}$.
3) Calculate

$$\begin{aligned}
\mathbf{A}_1^k &= -\frac{V_\Omega}{kN_\mathrm{s}}\sum_{i=1}^{N_\mathrm{s}} \phi(\mathbf{x}^i)\Big\{\mathbf{f}^T(\mathbf{x}^i)\left[\nabla\phi(\mathbf{x}^i)\right]^T \\
&\qquad + \left[\nabla^T\mathbf{f}(\mathbf{x}^i)\right]\phi^T(\mathbf{x}^i)\Big\} + \frac{k-1}{k}\mathbf{A}_1^{k-1}, \\
\mathbf{A}_2^k &= \frac{V_\Omega}{kN_\mathrm{s}}\sum_{i=1}^{N_\mathrm{s}}\left[\mathbf{A}_{21}(\mathbf{x}^i) + \mathbf{A}_{22}(\mathbf{x}^i)\right] + \frac{k-1}{k}\mathbf{A}_2^{k-1},
\end{aligned}$$

   where V_Ω is the volume of Ω.
4) If $\|\mathbf{A}_1^k - \mathbf{A}_1^{k-1}\| \le \varepsilon_A$ and $\|\mathbf{A}_2^k - \mathbf{A}_2^{k-1}\| \le \varepsilon_A$, then stop. Otherwise, go to step 2).

Monte-Carlo integration is *embarrassingly parallel* and can be easily implemented on parallel computational architecture including parallel computation nodes and GPU [10].

Proof of step 3): Monte-Carlo integration with uniform samples of a scalar function, $w(\mathbf{x})$ is given by

$$W_1^1 = \int_\Omega w(\mathbf{x})d\mathbf{x} \approx \frac{V_\Omega}{N_\mathrm{s}}\sum_{i=1}^{N_\mathrm{s}} w(\mathbf{x}^i)$$

If this calculation is repeated $(k-1)$-times, then

$$W_1^{k-1} = \frac{V_\Omega}{(k-1)N_\mathrm{s}} \sum_{i=1}^{(k-1)N_\mathrm{s}} w(\mathbf{x}^i).$$

Add N_s samples at k-th step,

$$\begin{aligned} W_0^k &= \frac{V_\Omega}{kN_\mathrm{s}} \sum_{i=1}^{kN_\mathrm{s}} w(\mathbf{x}^i) \\ &= \frac{V_\Omega}{kN_\mathrm{s}} \left[\sum_{i=(k-1)N_\mathrm{s}+1}^{kN_\mathrm{s}} w(\mathbf{x}^i) + \sum_{i=1}^{(k-1)N_\mathrm{s}} w(\mathbf{x}^i) \right] \\ &= \frac{V_\Omega}{kN_\mathrm{s}} \sum_{i=(k-1)N_\mathrm{s}+1}^{kN_\mathrm{s}} w(\mathbf{x}^i) + \frac{k-1}{k} W_1^{k-1}. \end{aligned}$$

■

Instead of a single sensor, N_ms number of multiple sensors can be deployed, where each sensor provides a measurement $\mathbf{y}_k^i$ for k-th step for $i = 1, 2, \ldots, N_\mathrm{ms}$, and $\mathbf{y}_k^i$ could be empty if i-th sensor measurement is not available. The sensor model for each is given by $p^i(\mathbf{y}_k^i|\mathbf{x})$. Assume that each sensor measurement is independent of each other, then the combined sensor measurement model becomes

$$p(\mathbf{y}_k^1, \mathbf{y}_k^2, \ldots, \mathbf{y}_k^{N_\mathrm{ms}}|\mathbf{x}) = \prod_{i=1}^{N_\mathrm{ms}} p^i(\mathbf{y}_k^i|\mathbf{x}). \tag{18}$$

Each $p^i(\mathbf{y}_k^i|\mathbf{x})$ is less than or equal to 1, and the product quickly approaches zero. If the above is implemented directly, the result of the multiplication might be always zero for most of the time as underflow in the calculation occurs. In order to avoid the underflow, the calculation is implemented in the following way:

Algorithm 3.2: (Calculation of multiple sensor likelihood)

1) Take $\log(\cdot)$ of (18) and set

$$p_\mathrm{ms} = \sum_{i=1}^{N_\mathrm{ms}} \log\left[p^i(\mathbf{y}_k^i|\mathbf{x})\right]$$

2) $p(\mathbf{y}_k^1, \mathbf{y}_k^2, \ldots, \mathbf{y}_k^{N_\mathrm{ms}}|\mathbf{x}) = e^{p_\mathrm{ms}}$

In *Update* step, (14) and (15), some improvements in terms of computation are possible.

Lemma 3.3: The update equation, (15), is equivalent to

$$\mathbf{c}(t_{k+1}^+) = \alpha\ Y(\mathbf{y}_{k+1})\ \mathbf{c}(t_{k+1}^-) \tag{19}$$

where

$$\alpha = \frac{1}{\mathbf{c}^T(t_{k+1}^-)\ Y^T(\mathbf{y}_{k+1})\phi_I}, \tag{20}$$

$$\phi_I := \int_\Omega \phi d\mathbf{x}, \tag{21}$$

Proof: The integration of the conditional pdf over Ω is equal to one,

$$\int_\Omega p_N(t_{k+1}^+) d\mathbf{x} = \int_\Omega \mathbf{c}^T(t_{k+1}^+)\phi(\mathbf{x}) d\mathbf{x} = 1,$$

and

$$\alpha\ \mathbf{c}^T(t_{k+1}^-)\ Y^T(\mathbf{y}_{k+1}) \int_\Omega \phi d\mathbf{x} = 1.$$

■

Note that (21) is to be calculated a priori and stored.

In addition, consider $Y(\mathbf{y}_{k+1})$ for a multiple sensor case

$$Y(\mathbf{y}_{k+1}) = \int_\Omega \left[\prod_{i=1}^{s} p_i(\mathbf{y}_{k+1}^i|\mathbf{x}) \right] \phi\phi^T d\mathbf{x}, \tag{22}$$

Lemma 3.4: The Monte-Carlo integration is obtained by

$$Y(\mathbf{y}_{k+1}) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \phi(\mathbf{x}^i)\phi^T(\mathbf{x}^i).$$

where N_s random samples of $\mathbf{x}^i$ in Ω is drawn from the sensor likelihood pdf as follows:

$$\mathbf{x}^i \sim \left[\prod_{i=1}^{s} p_i(\mathbf{y}_{k+1}^i|\mathbf{x}) \right].$$

Proof: By the property of Monte-Carlo integration,

$$\begin{aligned} Y(\mathbf{y}_{k+1}) &\approx \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{\left[\prod_{i=1}^{s} p_i(\mathbf{y}_{k+1}^i|\mathbf{x}^i)\right] \phi(\mathbf{x}^i)\phi^T(\mathbf{x}^i)}{\mathrm{pdf}(\mathbf{x}^\mathbf{i})} \\ &= \frac{1}{N_s} \sum_{i=1}^{N_s} \phi(\mathbf{x}^i)\phi^T(\mathbf{x}^i). \end{aligned}$$

■

In this Monte-Carlo integration, the samples are now more efficiently used as they are concentrated around regions where the sensor likelihood is higher.

For the n-dimensional case, Ω is defined as a hyper-rectangle,

$$\Omega = \{\mathbf{x}|\mathbf{x} \in [b_1, a_1] \times [b_2, a_2] \times \ldots [b_n, a_n]\}, \tag{23}$$

where $x_i \in [b_i, a_i]$, x_i is the i-th element of $\mathbf{x}$, and $b_i < a_i$ for $i = 1, 2, \ldots, n$. For each x_i,

$$\psi_p(x_i) = \begin{cases} \dfrac{1}{\sqrt{b_i - a_i}} & \text{for } p = 1, \\ \sqrt{\dfrac{2}{b_i - a_i}} \cos\left[\dfrac{(p-1)\pi}{b_i - a_i}(x_i - a_i)\right] & \text{for } 2 \le p \le N_b, \end{cases}$$

and the basis functions are generated by multiplication of ψ_p functions as follows:

$$\phi_l(\mathbf{x}) = \prod_{q=1}^{n} \psi_{I_q}(x_q), \tag{24}$$

for $\ell = 1, 2, \ldots, N$, where $N = N_b^n$, I_q is an index such that

$$\ell = I_1 + N_b[I_2 - 1] + N_b^2[I_3 - 1] + \ldots + N_b^{n-1}[I_n - 1],$$

and I_q is in $[1, N_b]$.

IV. Examples

A. First-order System

A first-order nonlinear system is given by [8]

$$\begin{aligned} dx_t &= \sin(x_t)dt + d\beta_t, \\ y_k^i &= x(t_k) + v_k^i \end{aligned}$$

with $Q = 0.5$ and $R^i = 0.5$ for $i = 1, 2, \ldots, 10$, i.e., 10 identical sensors provide the measurement at every 0.1s interval. The sample space, Ω, is defined by $[-6.5, 6.5]\pi$, the number of

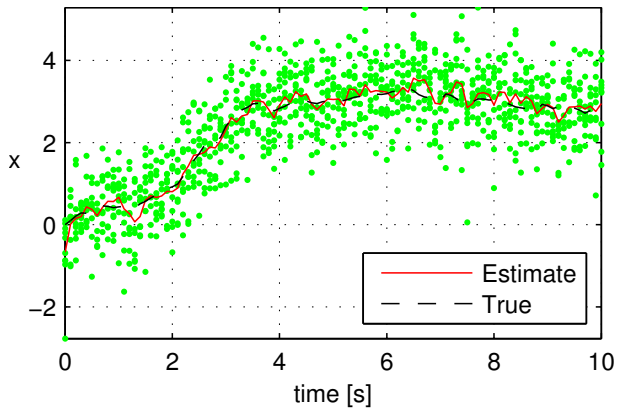

Fig. 1. First-order example, where sensor measurements are green dots; the estimated mean value is in red solid line; and the true state is in dashed line.

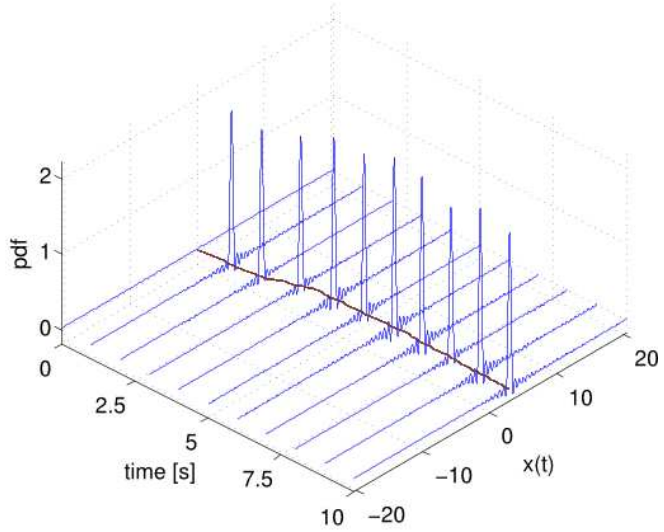

Fig. 2. Evolution of pdf at 1s interval with true state (red line)

basis functions is equal to 128. Fig. 1 shows the estimated state trajectory compared to the true state trajectory, where the estimate is calculated using the pdf obtained. Fig. 2 shows the evolution of the approximate probability density function $p_N(t,\mathbf{x}|\ Y_k)$, at 1s interval, along with the true state trajectory indicated by the solid red line. Initially, it is a uniform distribution over Ω as no information is available.

B. Van der Pol Oscillator

A modified Van der Pol oscillator is used to test the algorithm for a second-order system. The system dynamics is as follows [8]:

$$d\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}-x_2\\0.2(x_1^2-1)x_2+x_1\end{bmatrix} + d\beta,$$
$$y_k^i = x_2(t_k) + v_k^i,$$

where Q is the 2×2 diagonal matrix whose diagonal term is equal to 0.05, the measurement noise variance, R^i, is equal to 0.02 for $i=1,2,\ldots,10$, and the number of basis functions is 16. Fig. 3 shows the estimated and true state trajectories

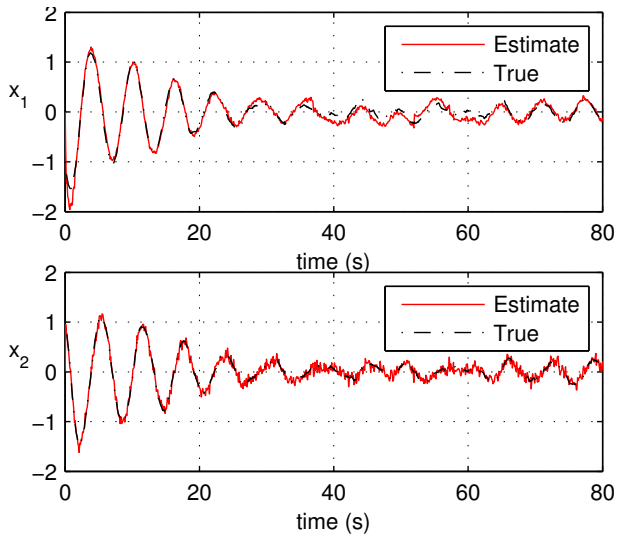

Fig. 3. Second-order example, where the estimated mean values are in read solid lines and the true are in dashed lines.

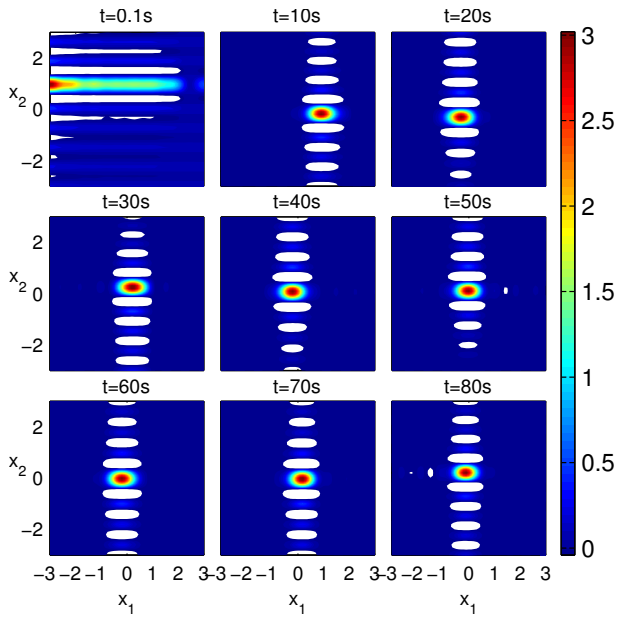

Fig. 4. Evolution of pdf at 10s intervals

for x_1, and for x_2. Fig. 4 shows the evolution of approximate probability density function at $10s$ intervals. At t=0s, the pdf is uniform over Ω and $E(x_1)=0$ and $E(x_2)=0$, while the true states are: $x_1(0)=-1$ and $x_2(0)=1$. As the pdf at $t=0.1$s shown in Fig. 4, $E(x_1)$ and $E(x_2)$ converge to the true value reasonably close. The white areas in Fig. 4 represent sections where the probability density function value is below zero, which is an unavoidable effect caused by the finite number of the basis functions. More basis functions will reduce the size of these undesirable regions with additional computations. The integration of whole region is equal to one as the pdf is normalized. Any moment of the states are the results of integration over the whole sample space, Ω, and they are less affected by these negative pdf values. It should

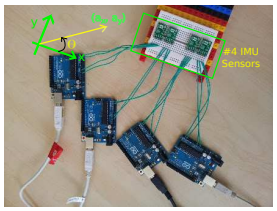

Fig. 5. Four IMU sensor configuration, where x and y axes are indicated.

be very careful in using the pdf integration over a subsection of Ω. There will be some resolution limit in terms of the size of subsection area over Ω to calculate the probability.

C. Pointing estimation

Four inertial measurement unit (IMU) sensors are connected and kept stationary while they measure the accelerations, as shown in Fig. 5. In order to use the sensor measurements for a nonlinear estimation problem, a virtual pointing platform, where four sensors are attached, is assumed and its dynamics are given by

$$d\theta = d\beta,$$
$$y_k^i = \tan(\theta_k^i) + v_k^i,$$

where y_k^i is the k-th measurement of the i-th sensor, $Q = 0.05$, R for each sensor is 0.618, 0.645, 0.404, 0.146, respectively, $\Omega = [-85^\circ, 85^\circ]$, and the virtual sensor providing the measurement is constructed using the accelerometers as follows: $y_k^i = a_y^i / a_x^i$, where a_x^i and a_y^i are x and y directional acceleration measurements respectively, from i-th IMU sensor. Ideally, if the platform is in the perfectly perpendicular to the gravitational acceleration, a_x^i and a_y^i are equal to zero. It is more likely that the surface is slightly tilted from the horizontal and both accelerations are affected by noise. Hence, the measurement provided by the virtual sensor, y_k^i, would have very strong noise effects as shown in Fig. 6. All dots in Fig. 6 are the measurements at each instance, where each colour represents the measurement from each sensor. The measurements are available from around 5s, the true angle is zero and the estimated is indicated by the blue solid line, where the number of basis functions for the filter is 64. As shown by the measurements, the estimate of θ is a lot better than the individual measurements and it is also better than a direct average of the four sensor outputs (not indicated).

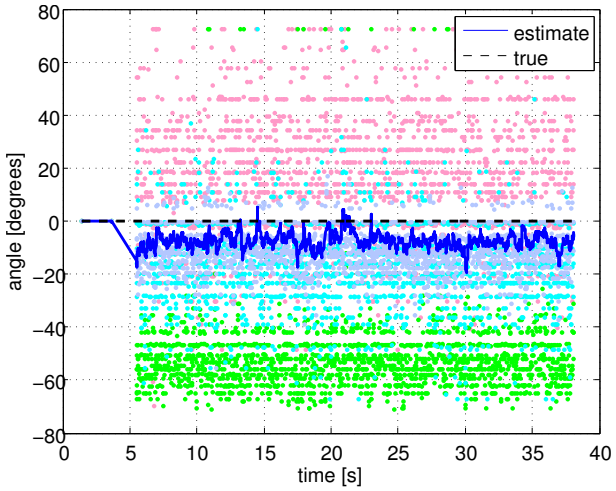

Fig. 6. Pointing angle (θ) estimation, where all dots are the measurements.

V. Conclusions & Future Works

Nonlinear projection filter was presented in the late 90's and the computational complexity was the main obstacle for any practical implementations of the filter. Current increasing computational power and parallel architectures allow the practical application of the exact nonlinear filter. Several ways to improve the computation of the filter are presented based on Monte-Carlo integration and the usage of multiple parallel sensors. The performance of the proposed methods are demonstrated by two numerical examples and one experimental example. Future works will include: 1) applying the filter for indoor navigation problem, whose state dimension could be up to twelve; 2) real-time re-positioning algorithm for the sampling space, Ω, while the states are progressed; and 3) implement the algorithms in GPU or FPGA.

Acknowledgement

This research is supported by School of Mechanical Engineering, University of Leeds, Leeds, UK.

References

[1] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960.

[2] M. S. Grewal and A. P. Andrews, "Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]," *Control Systems, IEEE*, vol. 30, pp. 69–78, June 2010.

[3] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, pp. 417–429, September-October 1982.

[4] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, pp. 401–422, Mar. 2004.

[5] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pp. 153–158, IEEE, Aug. 2000.

[6] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, pp. 174–188, Feb. 2002.

[7] N. G. Van Kampen, *Stochastic Processes in Physics and Chemistry, Third Edition (North-Holland Personal Library)*. North Holland, 3 ed., May 2007.

[8] R. Beard, J. Kenney, J. Gunther, J. Lawton, and W. Stirling, "Nonlinear Projection Filter Based on Galerkin Approximation," *Journal of Guidance, Control, and Dynamics*, vol. 22, pp. 258–266, Mar. 1999.

[9] G. Qian, K. Shafique, and P. Wang, "Fusion of nonlinear motion dynamics using fokker-planck equation and projection filter," in *Information Fusion (FUSION), 2014 17th International Conference on*, pp. 1–7, July 2014.

[10] NVIDIA, "CUDA parallel computing platform," Mar. 2015.

[11] L. Meirovitch, *Analytical Methods in Vibrations*. Macmillan Publishing Co., Inc., 1967.

[12] X. Chen, Q. Dai, and C. Li, "A fast algorithm for computing multidimensional DCT on certain small sizes," *Signal Processing, IEEE Transactions on*, vol. 51, pp. 213–220, Jan. 2003.

[13] S. Boussakta and H. O. Alshibami, "Fast algorithm for the 3-D DCT-II," *Signal Processing, IEEE Transactions on*, vol. 52, pp. 992–1001, Apr. 2004.