A NEW EVALUATION PLATFORM FOR NAVIGATION SYSTEMS

Thomas Hanefeld Sejerøe * **Niels Kjølstad poulsen** **
Ole Ravn *

* *Ørsted•DTU, Automation,*
The Technical University of Denmark,
Building 326, DK-2800 Kgs. Lyngby, Denmark
*E-mail:*991707@student.dtu.dk, or@oersted.dtu.dk

** *Informatics and Mathematical Modelling,*
The Technical University of Denmark,
Building 321, DK-2800 Kgs. Lyngby, Denmark
*E-mail:*nkp@imm.dtu.dk

Abstract: The KALMTOOL 2 toolbox is a set of MATLAB tools for state estimation for nonlinear systems. The toolbox contains functions for extended Kalman filtering as well as for two new filters called the DD1 filter and the DD2 filter. It also contains function for Uncented Kalman filters as well as three versions of particle filters. The toolbox requires MATLAB ver. 6, but no additional toolboxes are required.

Keywords: State estimation, Kalman filtering, nonlinear systems Autonomous robots

1. INTRODUCTION

In this paper a newly developed platform for evaluation of estimation algorithms, Kalmtool 2, will be described. The purpose of the platform is to make evaluation of different algorithms for solving nonlinear state estimation problems and to enable a comparison with new methods.

During the work it was found that the extended Kalman filter was somewhat inconvenient to use in some of our applications. A small modification of the application sometimes had serious implications on the EKF implementation. Moreover, it was often difficult to implement. Our problem was that the EKF requires a linearization of the system model. Sometimes this is easy to find but sometimes it can be pretty hard. In any case, it makes things inflexible. If a small change is made in the model, one has to work out a new set of derivatives. This is particularly inconvenient in model calibration where certain model parameters are temporarily included in the state vector and estimated simultaneously with the actual states.

Since it was suggested, the extended Kalman filter (EKF) has undoubtly been the dominating technique for nonlinear state estimation. Nevertheless, the EKF is known to have several drawbacks. These are mainly due to the Taylor linearization of the nonlinear transformations around the current state estimate. The linearization requires that Jacobians of state transition and observation equations are derived, which is often a quite complex task. Moreover, sometimes there are points in which the Jacobians are not defined. In addition to the difficulties with implementation, convergence problems are often encountered due to the fact that the linearized models describe the system poorly.

Previous work include several toolboxes and other platform. Focus here is on comparision and transparency.

The paper is organized as follows: first overall design philosophy behind the platform is described. Next a description of the estimation algorithms are given including the extended Kalman filter, the Uncented Kalman filter and different types of particlefilters. Section 4 gives an extensive example study as well as a demonstration of the platform for comparing algorithms for navigation of a mobile robot. Finally conclusions and references are given.

2. THE PLATFORM

The design philosophy of the simulation platform has been to provide a tool, which is simple and transparent in its inner workings, yet powerful and reliable. With this in mind, ease of use in applications and with respect to development of new algorithms have been a high priority.

The platform itself is based on a series of functions written in Matlab, which are combined with a Simulink graphical interface for ease of contruction. As Simulink by design resembles the block-diagrams of control theory, construction of systems becomes very straightforward and all signal paths are visible.

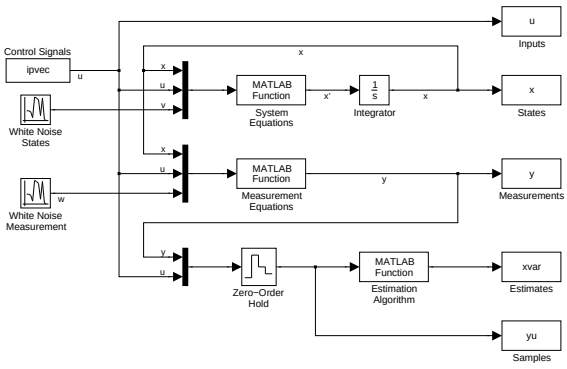

Fig. 1. *The Simulink layout of a continuous system.*

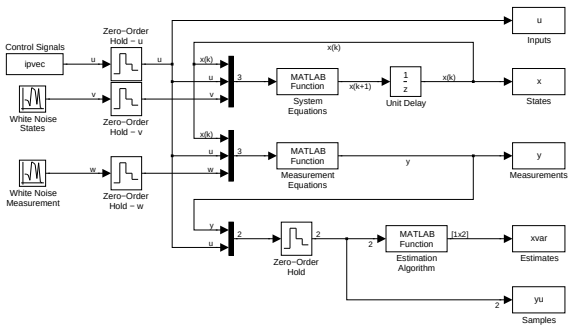

Fig. 2. *The Simulink layout of a discrete time systems*

The algorithms, which are implemented in the platform, are constructed in a model-independent manner. The models are defined externally as simple function files in Matlab, in accordance with the format used for solving differential equations numerically (see the documentation for Matlab). Depending on the application and the system at hand, continuous time step or fixed time step integration may be used, as well as discrete time systems.

Finally, the functions are not intended solely for use with Simulink. This means that any data set consisting of measurements and inputs may be filtered using the functions, given that a system and measurement description is provided. All in all, this provides a great deal of flexibility, ease of testing control algorithms, as well as testing various data filtering methods.

3. ESTIMATION ALGORITHMS

Consider a system in which the evolution of the state sequence $\{x_k \in \mathbb{R}^n,\ k \in \mathbb{N}\}$ is given by

$$x_{k+1} = f_k(x_k, u_k, v_k) \tag{1}$$

where f_k is a possible nonlinear function of the state, x_k, the input (control) signal, u_k and the process noise, v_k. The process noise is assumed to be a sequence $\{v_k \in \mathbb{R}^n\ k \in \mathbb{N}\}$ of i.i.d. stochastic vectors.

The objective is to estimate x_k from measurements

$$y_k = g_k(x_k, e_k) \qquad \in \mathbb{R}^m \tag{2}$$

where also g_k is a possible nonlinear function of the state and the measurement noise, e_k. The measurement noise is assumed to be a sequence, $\{e_k \in \mathbb{R}^m\ k \in \mathbb{N}\}$, of i.i.d. stochastic vectors. More specific we seek an estimate of x_k based on all available measurements (and known inputs) $Y_{0:k} = \{(y_i, u_i),\ i = 0, ..., k\}$.

The solution to this problem is embedded in the conditional degree of belief in the state, x_k given the data, $Y_{0:k}$. The problem is then (recursively) to determine the pdf. $p(x_k|y_{0:k})$. If the initial distribution, $p(x_0)$, is known then the solution can in principle be determined through the recursions:

$$p(x_k|Y_{0:k-1}) = \int_{\Omega_x} p(x_k|x_{k-1}) p(x_{k-1}|Y_{0:k-1}) dx_{k-1} \tag{3}$$

and

$$p(x_k|Y_{0:k}) = \frac{p(y_k|x_k)}{p(y_k|Y_{0:k-1})} p(x_k|Y_{0:k-1}) \tag{4}$$

These two recursions are related to the dynamic ((3)) and the inference ((4)) step, respectively and can only in special cases be solved analytically. In the linear Gaussian case the pdf. can be parameterized in terms of mean and variance and the recursions results in the well known Kalman filter. In that case (the linear Gaussian case with standard assumptions including $x_0 \in \mathbf{N}(\hat{x}_0, P_0)$) the system is assumed to be given by the recursions:

$$x_{k+1} = Ax_k + Bu_k + v_k \qquad v_k \in \mathbf{N}_{iid}(0, R_1)$$

$$y_k = Cx_k + e_k \qquad e_k \in \mathbf{N}_{iid}(0, R_2)$$

The Kalman filter is given by the prediction or the time updates

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{5}$$
$$P_{k+1|k} = AP_{k|k}A^T + R_1 \tag{6}$$

and the inference recursion

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_{k|k-1}) \tag{7}$$
$$P_{k|k} = P_{k|k-1} - K_k C P_{k|k-1} \tag{8}$$

where:

$$K_k = P_{k|k-1}C^T S_k^{-1}$$

and

$$\hat{y}_{k|k-1} = X\hat{x}_{k|k-1} \qquad S_k = CP_{k|k-1}C^T + R_2$$

In this case, the prediction in (3) results in (5) and can also be found as an application of calculus for linear operations on Gaussian vectors. The inference recursion in (7) emerge from (4) or as an application of the Projection Theorem.

The various filters differs in the way the handle the propagation of the distributions through the two non-linearities, f and g, and how the inference is carried out. The next three filters are all based on the projection Theorem.

3.1 The Extended Kalman filter

The Extended Kalman filter is as its name indicate based on an extension of the application of the Kalman filter to the nonlinear case. The Extended Kalman filter (EKF) is based on a standard Taylor expansion of the nonlinear functions and can be regarded as a local approximation. In general the approximation is best for small deviations from the point of linearization.

The basic idea is related to the problem of determine the distribution of z if

$$z = F(x)$$

and the distribution of x is known to be $\mathbf{N}(\hat{x}, P_x)$. The approximation is simply to use

$$z \in \mathbf{N}\left(F(\hat{x}), AP_xA^T\right)$$

where

$$A = \left.\frac{\partial}{\partial x}F\right|_{\hat{x}}$$

This approximation applies both to the process equation (and f) and the measurement equation (and g). In fact, the only changes with respect to (5)-(8) is

$$\hat{x}_{i+1|i} = f_i(\hat{x}_{i|i}, u_i, 0) \qquad \hat{y}_{i|i} = g_i(\hat{x}_{i|i}, 0)$$

The variance update, (6) and (8), are unchanged (except for the state dependent system matrices).

3.2 Divided difference filters

The divided difference filter exists in a first order version (DD1) and in a second order version (DD2) and is based on Stirlings interpolation formula (see (Nørgaard *et al.*, 2000) and Figure 3 for illustration).

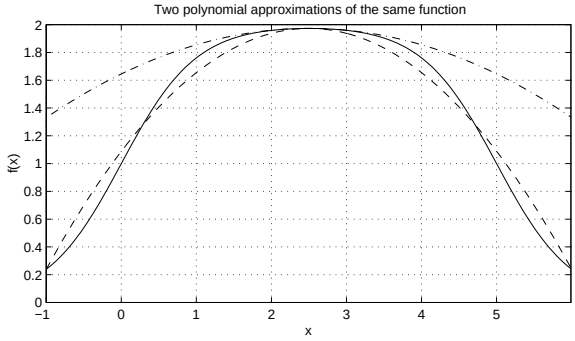

Fig. 3. Comparison of a second-order polynomial approximation obtained with the Taylor (dot-dashed) and the Stirling method (dashed)

Let again, x be a stochastic variable and $x \in \mathbf{N}\left(\hat{x}, S_x S_x^T\right)$. The approximation which takes the variation of w into account is

$$F(x) = F(\hat{x}) + \bar{\nabla}_z F(\hat{x})(x-\hat{x}) + \frac{1}{2}\bar{\nabla}_x^2 F(\hat{x})(x-\hat{x})^2 + \varepsilon$$

where

$$\bar{\nabla}_x F(\hat{x}) = \mathbf{Matr}_{ij}\left\{\frac{1}{2h}\left[F_i(\hat{x}+hS_{xj}) - F_i(\hat{x}-hS_{xj})\right]\right\}$$

$$\bar{\nabla}_x^2 F(\hat{x}) = \mathbf{Matr}_{ij}\left\{\frac{1}{h^2}\left[F_i(\hat{x}+hS_{xj}) + F_i(\hat{x}-hS_{xj}) - 2F_i(\hat{x})\right]\right\}$$

Here h is a scale parameter and S_{xj} is the j'th column in S_x. In the Gaussian case the choice $h^2 = 3$ is in some sense optimal (see (Nørgaard *et al.*, 2000)).

Introduce the notation

$$F_p^+ = F(\hat{x}+hS_{x,p}) \qquad F_p^- = F(\hat{x}-hS_{x,p}) \qquad F^0 = F(\hat{x})$$

For the DD2 filter the approximation is then

$$\hat{z} = \frac{h^2-n_x}{h^2}F^0 + \frac{1}{2h^2}\sum_{p=1}^{n_x} F_p^+ + F_p^-$$

and

$$P_z = \frac{1}{4h^2}\sum_{i=1}^{n_x}(F_p^+ - F_p^-)(F_p^+ - F_p^-)^T + \frac{h^2-1}{4h^2}\sum_{i=1}^{n_x}(F_p^+ + F_p^- - 2F^0)(F_p^+ + F_p^- - 2F^0)^T$$

For the first order filter (DD1) only the first terms in the approximations are used.

In the divided difference filters (DD1 and DD2) the propagation of mean and variance is determined through the approximations mentioned above. The inference is based on the Projection Theorem.

3.3 The Unscented kalman filter

The Uncented filter is based on the (uncentedi) transformation of a stochastic variable, x, through a nonlinear function, $F(x)$ (see (Julier and Uhlmann, 2004)). Assuming again the mean of x is $\hat{x}$ and the variance matrix is $P_x = S_x S_x^T$, then the sigma points are defined as:

$$\begin{aligned} x^{(1)} &= \hat{x} & w_0 &= \frac{\kappa}{n_x+\kappa} \\ x^{(i)} &= \hat{x} + \sqrt{(n_x+\kappa)}S_{xi} & w_i &= \frac{\kappa}{2(n_x+\kappa)} \quad i = 1, \ldots n_x \\ x^{(j+n_x)} &= \hat{x} - \sqrt{(n_x+\kappa)}S_{x,j} & w_{j+n_x} &= \frac{\kappa}{2(n_x+\kappa)} \\ & & & \quad j = 1, \ldots, n_x \end{aligned}$$

Here κ is a scaling parameter and w_i is the weight associated with a point and

$$\sum_{i=0}^{2n_x} w_i = 1$$

Each sigma point is propagated through the nonlinear function

$$z^{(i)} = F(x^{(i)}) \qquad i = 0, \ldots 2n_x$$

and the approximation is then

$$\hat{z} = \sum_{i=0}^{2n_x} w_i z^{(i)}$$

and

$$P_z = \sum_{i=0}^{2n_x} w_i (z^{(i)} - \hat{z})(z^{(i)} - \hat{z})^T$$

The standard UKF is based on the approximation mentioned above and the Projection Theorem. In the scaled version of UKF the weight is chosen in a slightly different manner (see (Julier, 2002) or (Wan and van der Merwe, 2000) for details).

3.4 Particle filters

Particle filters comes in several versions and implementations (see e.g. (Arulampalam *et al.*, 2002) or (van der Merwe *et al.*, 2000)). In the most basic version (Exp. PF) implemented in the platform the nonlinearities are dealt with by propagating a swarm of particle through the nonlinearities. Again assuming $x \in \mathbf{N}(\hat{x}, P_x)$ a number (N) of particles are generated

$$x^{(i)} \leftarrow \mathbf{N}(\hat{x}, P_x) \qquad i = 1, \ldots N$$

and propagated through the nonlinear function

$$z^{(i)} = F(x^{(i)})$$

The approximation is then simply

$$\hat{z} = \sum_{i=1}^{N} z^{(i)} \qquad P_z = \sum_{i=1}^{N} (z^{(i)} - \hat{z})(z^{(i)} - \hat{z})^T$$

In the most basic version (Exp. PF) the inference is based on the Projection Theorem and the nonlinearities are handled with the method mentioned above.

In the generic particle filters (Gen. PF) the inference is not based on the Projection Theorem, but is carried out by applying 4 directly. That results in weights associated with each of the particles. In this version the particle are only initially generated as described above. After the inference step the particles are resampled from a distribution characterized by the weights. In the last version (MH. PF) implemented here on this platform, the resampling is performed by means of the Metropolis-Hastings algorithm.

4. EXAMPLE STUDY

The versatility of the simulation framework is most evident when implementing a number of examples. For the purpose of this demonstration, a continuous time differential equation system and a discrete time difference equation system are selected. The continuous time system is a very simple model of a dead-reckoning guidance for a small mobile robot. The discrete time system is an academic example of a nonlinear system (though in a simplified form), which have been used previously as a benchmark for testing filter algorithms (Netto *et al.*, 1978))

The model of the small mobile robot (unicycle type) is given by the set of equations stated below and is only slightly nonlinear. The equations yield a position in a two dimensional space as well as a heading. An input signal consisting of the velocity, γ, and turnrate, ω, is used as a way of introducing an external source of disturbance. The observation equations are purely linear, and the noise is modelled as being additive, zero mean, Gaussian distributed in both sets of equations.

$$\begin{aligned} \dot{x} &= \gamma \cos(\theta) + v_1 \\ \dot{y} &= \gamma \sin(\theta) + v_2 \\ \dot{\theta} &= \omega + v_3 \end{aligned} \tag{9}$$

The process noise sources, v_n, as well as the observation noise sources, w_n, are specified by $N(0, 0.01)$.

$$\begin{aligned} z_1 &= x + w_1 \\ z_2 &= y + w_2 \\ z_3 &= \theta + w_3 \end{aligned} \tag{10}$$

An example of a simulation using the Divided Difference (2nd order) as estimator can be seen in figure 4. The integral of the control signal, ω, is seen below the path traced by the robot.

In order to compare a range of techniques implemented in the framework, accuracy results are given in table 1. Attempting to find a fair estimate of the accuracy, 100 runs were made with each algorithm and the average values were found. The particle filters all used 200 particles per time update. The table contains the "worst case" values for the three states.

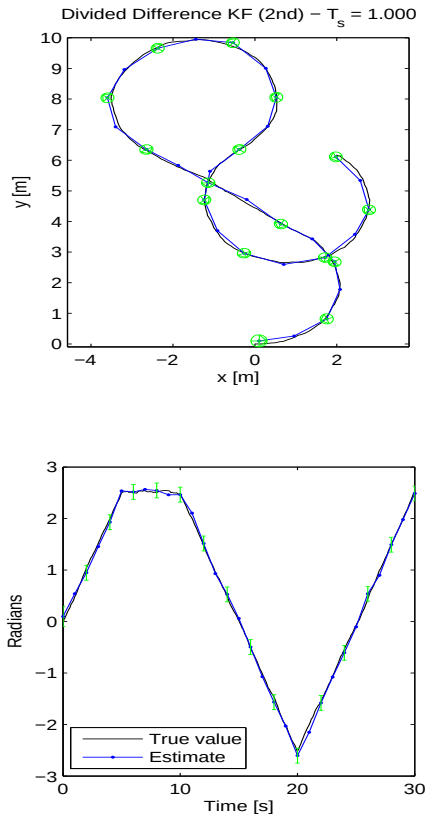

Fig. 4. *A path traced by the small unicycle robot. The estimation routine employed is the 2nd order Divided Difference filter with a sampling frequency of 1 Hz. At every other estimated state, the 95% confidence intervals are drawn as ellipses or bars respectively. The estimate is at no point outside the confidence intervals.*

Algorithm	Max. RMSE	Max. Var. Est.	Time
C.D. EKF	0.06799	0.005717	1.000
Std. UKF	0.07399	0.006779	2.678
Scl. UKF	0.07331	0.006693	3.673
DD1	0.07251	0.006779	2.640
DD2	0.07177	0.006781	2.658
Exp. PF	0.07591	0.006479	16.86
Gen. PF	0.09698	0.039829	17.82
PF (MH)	0.08960	0.058690	18.53

Table 1. *Small mobile robot, worst value of mean estimate (x,y,θ) and maximum mean variance estimate of 100 Monte Carlo simulations. The table is split into Kalman filter variants (top) and particle filters (bottom). The particle filters all used 200 particles.*

Also listed in the table is the computational burden of each algorithm. The latter is given as a relative number compared to the runtime of a continuous-discrete extended Kalman filter (C.D. EKF). The times are relative, as other processor speeds and types will yield different absolute results. Furthermore, the algorithms and their runtimes may well benefit from numerical optimizations in application specific implementations. The algorithms used a fixed step integration (Matlab, Dormand-Prince, order 5) to solve equation 9. The standard Unscented Kalman filter (Std. UKF) performs very well, while it's scaled version gives a lower mean RMSE and a slightly lower mean variance estimates. The DD1 and DD2 both give low mean RMSE and consistent variance estimates - in this case, the second order parts of the DD2 does not yield much.

The second example is a nonlinear equation with a linear and noisy measurement. First, the process equation, x_{k+1} is listed, next the measurement equation, y_k.

$$\begin{aligned} x_{k+1} &= \frac{1}{2}x_k + \frac{25x_k}{1+x_k^2} + 8\cos(1.2k) + v_n \\ y_k &= x_k + w_k; \end{aligned} \tag{11}$$

Note that, both the noise sources, v_k and w_k, are zero mean Gaussian white noise with variances of 10.0 and 1.0 respectively. As was the case with the small robot model, a Monte Carlo series of simulations was made with a variety of estimation algorithms. Two examples of the appearance of a simulation can be found in figure 5.

The result of the Monte Carlo simulation can be seen in table 2. The Kalman filter type algorithms were simulated 1000 times and the means of the root mean square errors (RMSE) were found as well as the means of the variance estimates. The Particle Filter types were simulated 100 times with 200 particles per time update in all filters.

Algorithm	Mean RMSE	Mean Var. Est.	Time
C.D. EKF	0.9573	0.9206	1.000
Std. UKF	0.9472	0.9238	0.126
Scl. UKF	0.9503	0.9247	0.151
DD1	0.9417	0.9221	0.133
DD2	0.9260	0.9238	0.137
Exp. PF	0.9513	0.9165	2.067
Gen. PF	4.2326	31.595	5.917
PF (MH)	4.0238	28.165	8.543

Table 2. *Table of results for a Monte Carlo series of simulations on the discrete nonlinear and noisy system.*

Finally, in order to compare the precision of the three particle filters as a function of the number of particles per time update, a Monte Carlo series of simulations was made using the benchmark system. The results can be seen in figure 6. The series consisted of 100 runs per particle count, from 2 to 256 particles in increasing steps. The algorithms converge rather quickly as the particle count increases.

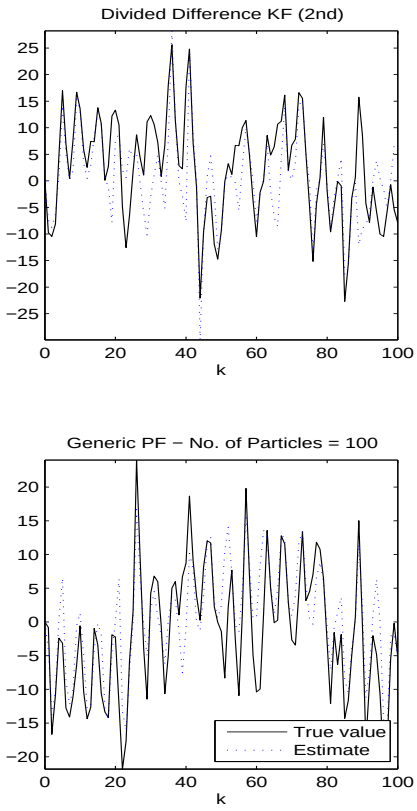

Fig. 5. *Two examples of the highly nonlinear and noisy system given in equation 11. The topmost is the 2nd order Divided Difference filter, while the bottommost is a generic Particle Filter.*

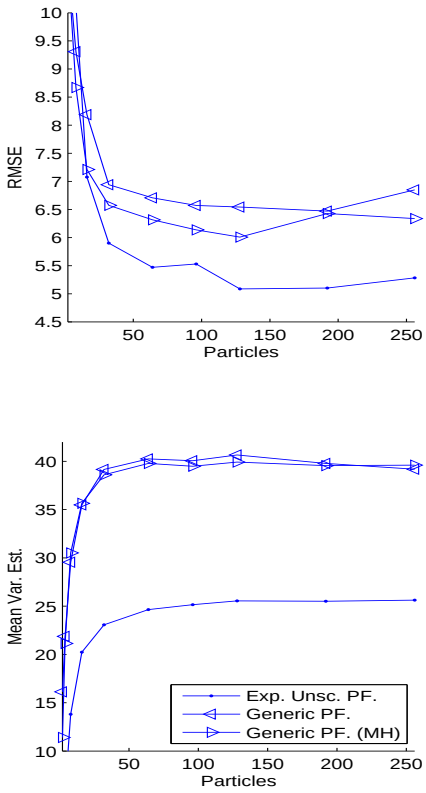

Fig. 6. *Two graphs depicting the effect of varying the particle count per update on the benchmark system. The mean RMSE and mean variance estimates are seen to converge rather quickly to relatively stationary values at around 100 particles per time update.*

5. CONCLUSION

In this paper we have presented the toolbox KALMTOOL ver. 2 which a set of MATLAB tools for state estimation for nonlinear systems. It contains functions for extended Kalman filtering as well as for the two new filters the DD1 filter and the DD2 filter. It also contains functions for Unscented (standard and scaled) Kalman filter as well as three versions of particle filters.

REFERENCES

Arulampalam, M.S., S. Maskell, N. Gordon and T. Clapp (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing* **50**(2), 174–188.

Julier, S. (2002). The scaled unscented transformation. *Proceedings of the American Control Conference* pp. 4555–4559.

Julier, Simon and Jeffrey Uhlmann (2004). Unscented filtering and nonlinear estimation. *Proceeding of the IEEE* **92**(3), 401–422.

Netto, A.M.L., L. Gimeno and M.J. Mendes (1978). A new spline algorithm for non-linear filtering of discrete time systems. *Proceedings of the 4th IFAC Symposium on Identification and System Parameter Estimation, Tbilisi, U.S.S.R.* pp. 2123–2130.

Nørgaard, Magnus, Niels K. Poulsen and Ole Ravn (2000). New development in state estimation for nonlinear systems. *Automatica* **36**, 1627–1638.

Nørgaard, Magnus, Niels Kjølstad Poulsen and Ole Ravn (2003). Kalmtool for use with matlab. In: *13th IFAC Symposium on System Identification, SYSID03, Rotterdam*. IFAC. Rotterdam. pp. 1490–1495.

van der Merwe, R., A. Doucet, N. de Freitas and E. Wan (2000). The unscented particle filter. Technical report. Cambridge University Engineering Department.

Wan, Eric and Rudolph van der Merwe (2000). The unscented kalman filter for nonlinear estimation. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*. pp. 153–158.