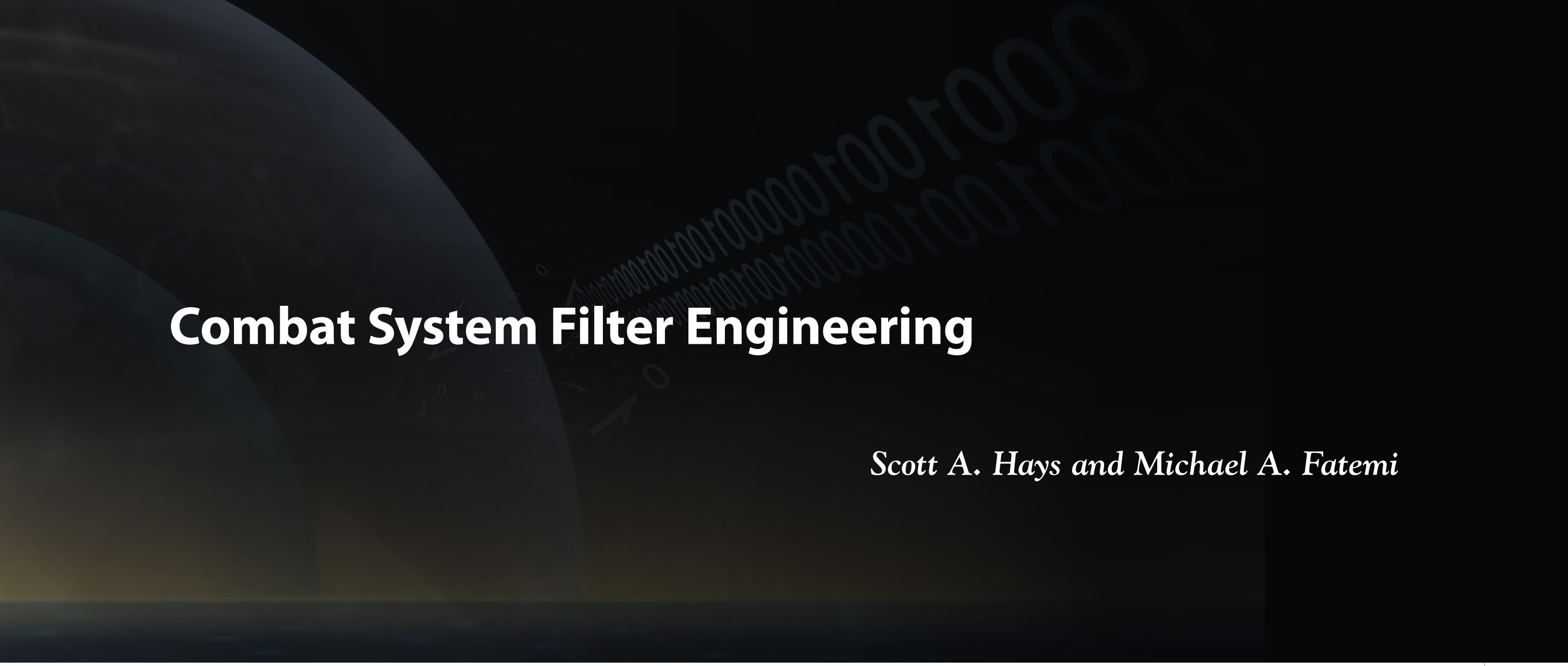


Combat System Filter Engineering

Scott A. Hays and Michael A. Fatemi

ABSTRACT

To develop a new combat system, the designer has to determine the optimal filter type, filter application point, and dynamic model assumption. The definition of an optimal filter will change depending on the filter's purpose and can change drastically depending on its application within the plan–detect–control–engage sequence. Ideally, the designer could apply a single computationally efficient filter algorithm that adapts in real time to threat maneuver, system bias, and measurement noise level while maintaining an accurate estimate with a high level of confidence. In practice, however, several different filters (often of different types) are applied to a single combat system in separate parts of the plan–detect–control–engage sequence to ensure the best results for problems that include track consistency, association, filter errors, and correlation. There are several key factors in developing a robust filter with the flexibility for the technology upgrades that are required to keep up with threat evolution. This article describes a design methodology to provide robustness in the face of dynamic threat behavior, lack of a priori knowledge, and threat evolution.

INTRODUCTION

Air and missile defense systems can be generalized as combat systems with architecture that allows the fire control loop to be integrated into the plan–detect–control–engage sequence. Air and missile defense combat systems usually have tracking sensors (plan–detect sequences), a decision component (control sequence), a control component (control sequence), and a guided missile (engage sequence). Optimal conditions required for the plan–detect–control–engage sequence can differ based on needs; a specific example of optimal conditions can be a very accurate position estimate and tight covariance to ensure that the tracking sensor can associate the different system tracks. A decision component needs to accurately extrapolate a track state into the future, which requires an accurate dynamic model and state estimate. Additionally, some components in the plan–detect–control–engage sequence could pass tracks to the next sequence, which changes the track picture. The new track picture results from filtered track estimate inputs rather than direct measurements. Early combat systems were forced to rely on simple filters such as alpha-beta filters, alpha-beta-gamma filters, discrete Kalman filters, and lookup-based filters; these filters provided computationally efficient algorithms but little system robustness. Increased computation capability allows for further expansion of filter types into nonlinear filters—i.e., extended Kalman filters (EKFs), unscented Kalman filters (UKFs), and, to an extent, particle filters

(PFs). This article introduces each of these nonlinear filters and provides a background for application of advanced filter algorithms.

NONLINEAR FILTERS

The increase in threat capability necessitates an increase in the sensor and tracking platform's ability to accurately predict future motion. Nonlinear filters provide a capability that extends beyond the linear assumptions to ensure system robustness. The expansion of filters from the linear to nonlinear realm inherently produces nonoptimal estimation, unlike that produced by the Kalman filter, which is an optimal estimator for linear dynamic systems and linear measurement models. Nonlinear filters use different methodologies (Taylor series expansion, unscented transformation [UT], Monte Carlo–based sampling approaches, and dynamic constrained optimization) to characterize and approximate nonlinear motion.

In this article, the linear/nonlinear dynamic and measurement equations are defined as:

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1}, \tag{1}$$

$$y_k = C_k x_k + v_k, \tag{2}$$

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}, \text{ and} \tag{3}$$

$$y_k = h(x_k) + v_k, \tag{4}$$

where x_k is the state; u_k is the control input; w_k is a model-bounded uncertainty (also referred to as state or model process noise); A_k is the state transition matrix; B_k is the control input matrix; y_k is the measurement; C_k is the observation matrix; v_k is a measurement-bounded uncertainty (also referred to as measurement process noise); f() is the nonlinear state transition function; h() is the nonlinear measurement function; R_w is the covariance associated with model-bounded uncertainty assuming the noise follows a Gaussian distribution with a zero mean; R_v is the covariance associated with measurement-bounded uncertainty assuming the noise follows a Gaussian distribution with a zero mean; and n_x or N refers to the dimension of the state vector.

Terminology used in this article is as follows: $x_{k1|k2}$ is the state estimate at time k_1 given measurements up to and including time k_2 (where $k_2 \le k_1$); $P_{k1|k2}$ is the estimated covariance matrix at time k_1 given measurements up to and including time k_2 (where $k_2 \le k_1$); y_{k1} is the measurement at time k_1; z_{k1} is the innovation (or measurement residual) at time k_1; and K_{k1} is the gain matrix at time k_1. The initial estimation period in this article is referred to as the a priori estimate, but in the literature this is also referred to as the predictive step or time update state. The update estimation period is referred to as the a posteriori estimate, but in the literature this is also referred to as the corrective step, update step, or measurement update step.

EXTENDED KALMAN FILTER

The EKF is an extension of the Kalman filter to represent nonlinear motion by using a Taylor series expansion to linearize about a local point (Figure 1). The EKF is based on local linearization of the nonlinear equations of motion around the prior state estimate. In the EKF, to represent the covariance prediction, a partial derivative matrix (Jacobian) of the state estimate matrix is computed and evaluated at each time step based on the current predicted states. More information on EKF

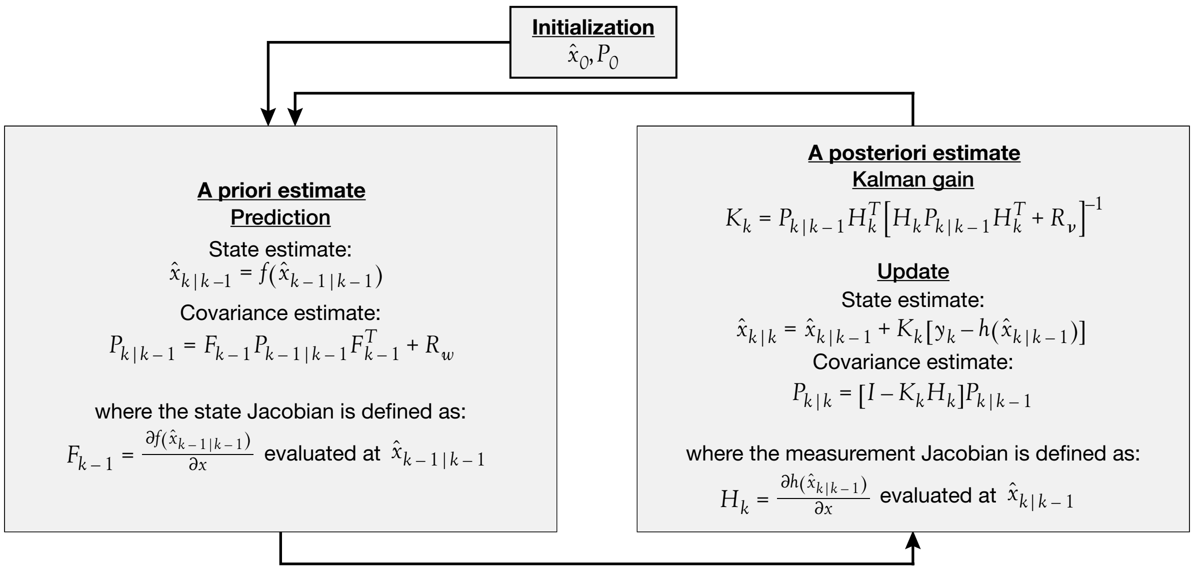

Initialization
$\hat{x}_0, P_0$
A priori estimate
Prediction
State estimate:
$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1})$
Covariance estimate:
$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F^T_{k-1} + R_w$
where the state Jacobian is defined as:
$F_{k-1} = \frac{\partial f(\hat{x}_{k-1|k-1})}{\partial x}$ evaluated at $\hat{x}_{k-1|k-1}$
A posteriori estimate
Kalman gain
$K_k = P_{k|k-1}H^T_k\left[H_kP_{k|k-1}H^T_k + R_v\right]^{-1}$
Update
State estimate:
$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k[y_k - h(\hat{x}_{k|k-1})]$
Covariance estimate:
$P_{k|k} = [I - K_kH_k]P_{k|k-1}$
where the measurement Jacobian is defined as:
$H_k = \frac{\partial h(\hat{x}_{k|k-1})}{\partial x}$ evaluated at $\hat{x}_{k|k-1}$

Figure 1. EKF flow diagram.

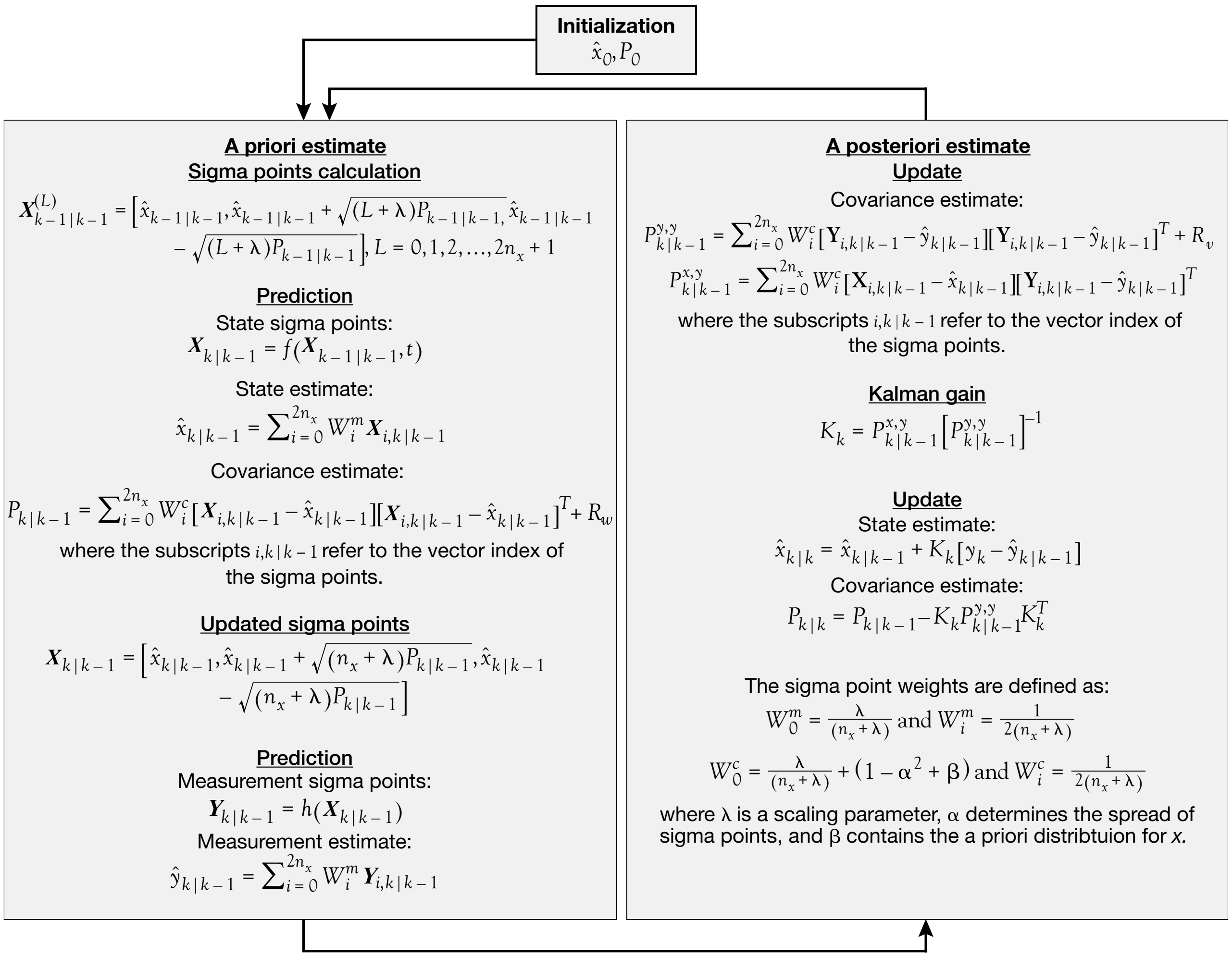
algorithms, derivation, and advances in research can be found in Refs. 1–5.

The EKF has limitations due to the calculation of the Jacobian matrix and the local linearization approximation. The EKF requires an accurate representation of the nonlinear equations of motion and an accurate estimate of the partial derivative state equations. For highly nonlinear systems, the equations of motion can be difficult to describe to a high degree, and the construction of the Jacobian matrix can be difficult. Errors within equations of motion, errors within Jacobian construction, or a local/global motion mismatch can result in divergence in the EKF error propagation, resulting in an unstable filter. The errors that can occur while deriving the EKF can lead to computational complexity and overall poor performance when tracking challenging targets.

UNSCENTED KALMAN FILTER

The UT is a method for calculating the statistics of a random variable that undergoes a nonlinear transformation and builds on the principle that it is easier to approximate a probability distribution than an arbitrary nonlinear function. The UKF uses the UT, creating a related statistic set (mean and covariance information) for a predetermined number of transformed points (Figure 2). The UKF approximates a probability distribution by a set of sample points; this results in a filtering method that does not require state linearization or calculation of a Jacobian matrix. State dynamic linearization and calculation of Jacobian matrices can be computationally intensive and require more accurate state estimation.

The number of transformed points (sigma points) is 2n$_x$ + 1 where n$_x$ is the dimension of the state vector. The weights for the UT can be selected to provide conditions that are bundled closer to the mean or approximate higher moments of the nonlinear function. In a Gaussian distribution, the weights are predetermined. UKF can use the nonlinear dynamic equations in the update process and can be tuned to provide robustness to reduce instability issues. More information on UKF algorithms derivation and advances in research can be found in Refs. 6–12.



Initialization

$\hat{x}_0, P_0$

A priori estimate

Sigma points calculation

$$\boldsymbol{X}^{(L)}_{k-1|k-1} = \left[\hat{x}_{k-1|k-1}, \hat{x}_{k-1|k-1} + \sqrt{(L+\lambda)P_{k-1|k-1}}, \hat{x}_{k-1|k-1} - \sqrt{(L+\lambda)P_{k-1|k-1}}\right], L = 0,1,2,\ldots,2n_x+1$$

Prediction

State sigma points:

$$\boldsymbol{X}_{k|k-1} = f(\boldsymbol{X}_{k-1|k-1}, t)$$

State estimate:

$$\hat{x}_{k|k-1} = \sum_{i=0}^{2n_x} W_i^m \boldsymbol{X}_{i,k|k-1}$$

Covariance estimate:

$$P_{k|k-1} = \sum_{i=0}^{2n_x} W_i^c [\boldsymbol{X}_{i,k|k-1} - \hat{x}_{k|k-1}][\boldsymbol{X}_{i,k|k-1} - \hat{x}_{k|k-1}]^T + R_w$$

where the subscripts $i,k|k-1$ refer to the vector index of the sigma points.

Updated sigma points

$$\boldsymbol{X}_{k|k-1} = \left[\hat{x}_{k|k-1}, \hat{x}_{k|k-1} + \sqrt{(n_x+\lambda)P_{k|k-1}}, \hat{x}_{k|k-1} - \sqrt{(n_x+\lambda)P_{k|k-1}}\right]$$

Prediction

Measurement sigma points:

$$\boldsymbol{Y}_{k|k-1} = h(\boldsymbol{X}_{k|k-1})$$

Measurement estimate:

$$\hat{y}_{k|k-1} = \sum_{i=0}^{2n_x} W_i^m \boldsymbol{Y}_{i,k|k-1}$$

A posteriori estimate

Update

Covariance estimate:

$$P^{y,y}_{k|k-1} = \sum_{i=0}^{2n_x} W_i^c [\mathbf{Y}_{i,k|k-1} - \hat{y}_{k|k-1}][\mathbf{Y}_{i,k|k-1} - \hat{y}_{k|k-1}]^T + R_v$$

$$P^{x,y}_{k|k-1} = \sum_{i=0}^{2n_x} W_i^c [\mathbf{X}_{i,k|k-1} - \hat{x}_{k|k-1}][\mathbf{Y}_{i,k|k-1} - \hat{y}_{k|k-1}]^T$$

where the subscripts $i,k|k-1$ refer to the vector index of the sigma points.

Kalman gain

$$K_k = P^{x,y}_{k|k-1}\left[P^{y,y}_{k|k-1}\right]^{-1}$$

Update

State estimate:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k[y_k - \hat{y}_{k|k-1}]$$

Covariance estimate:

$$P_{k|k} = P_{k|k-1} - K_k P^{y,y}_{k|k-1} K_k^T$$

The sigma point weights are defined as:

$$W_0^m = \frac{\lambda}{(n_x+\lambda)} \text{ and } W_i^m = \frac{1}{2(n_x+\lambda)}$$

$$W_0^c = \frac{\lambda}{(n_x+\lambda)} + (1 - \alpha^2 + \beta) \text{ and } W_i^c = \frac{1}{2(n_x+\lambda)}$$

where λ is a scaling parameter, α determines the spread of sigma points, and β contains the a priori distribtuion for *x*.

Figure 2. UKF flow diagram.

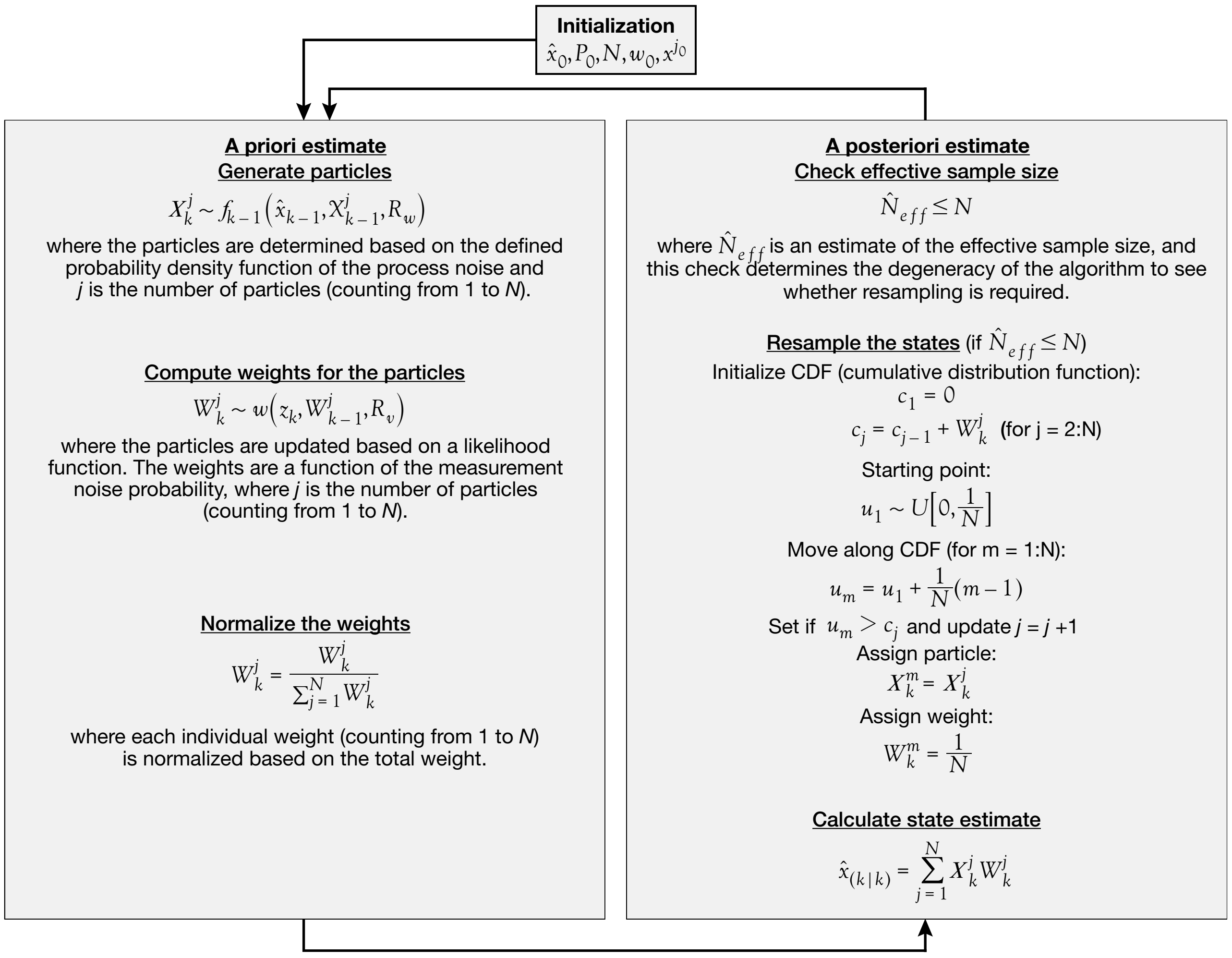
PARTICLE FILTERS

PFs are designed to address a wide range of nonlinear and non-Gaussian state estimation problems. PFs are recursive Bayesian filters (based on Bayes's rule), which utilize weighted particles sampled from the probability density function (PDF). The single most important technique in particle filtering is the selection of the distribution function parameters. The weighted particles are randomly generated state estimates of a user-defined number N. The PF has different implementations based on simplicity, the determination of whether the algorithm requires resampling, and the resampling method. The PF algorithm and flow diagram are based on the details provided in Ref. 13 and will detail a generic algorithm that uses resampling to help reduce the effects of degeneracy. The PF algorithm (Figure 3) randomly generates particles, computes weights for the particles, normalizes the weights, resamples the states, and calculates the a posteriori state estimate.

The PF can produce more accurate estimates for nonlinear systems because it does not rely on linearized covariance updates. It has been shown that the PF estimation error is lower than that of EKF but at a higher computational cost. The limitations of PFs are in the implementation and computational burden; in many cases, producing more accurate estimates than the EKF/UKF will require a large number of particles (N).

IMM FILTERS

The IMM technique is considered a class of the Bayesian–Markovian model. An IMM estimator is a suboptimal filter scheme that can estimate the state of a dynamic system with several behavior modes. The main feature of the IMM algorithm is an adaptive state estimator that favors the filter dynamics, among a number of filter models, that matches best with the target dynamics.[14] The IMM state estimator has been shown to accurately track maneuvering targets provided that the filter dynamics capture the wide range of maneuvering target dynamics. The IMM technique works through internal collaboration or interaction of all of the filters rather than independent filter analysis and selection. An IMM estimator can contain linear filters (Kalman



Initialization

$\hat{x}_0, P_0, N, w_0, x^{j_0}$

A priori estimate
Generate particles

$$X_k^j \sim f_{k-1}\left(\hat{x}_{k-1}, X_{k-1}^j, R_w\right)$$

where the particles are determined based on the defined probability density function of the process noise and *j* is the number of particles (counting from 1 to *N*).

Compute weights for the particles

$$W_k^j \sim w\left(z_k, W_{k-1}^j, R_v\right)$$

where the particles are updated based on a likelihood function. The weights are a function of the measurement noise probability, where *j* is the number of particles (counting from 1 to *N*).

Normalize the weights

$$W_k^j = \frac{W_k^j}{\sum_{j=1}^N W_k^j}$$

where each individual weight (counting from 1 to *N*) is normalized based on the total weight.

A posteriori estimate
Check effective sample size

$$\hat{N}_{eff} \leq N$$

where $\hat{N}_{eff}$ is an estimate of the effective sample size, and this check determines the degeneracy of the algorithm to see whether resampling is required.

Resample the states (if $\hat{N}_{eff} \leq N$)

Initialize CDF (cumulative distribution function):

$$c_1 = 0$$

$$c_j = c_{j-1} + W_k^j \quad \text{(for j = 2:N)}$$

Starting point:

$$u_1 \sim U\left[0, \frac{1}{N}\right]$$

Move along CDF (for m = 1:N):

$$u_m = u_1 + \frac{1}{N}(m-1)$$

Set if $u_m > c_j$ and update $j = j + 1$

Assign particle:

$$X_k^m = X_k^j$$

Assign weight:

$$W_k^m = \frac{1}{N}$$

Calculate state estimate

$$\hat{x}_{(k|k)} = \sum_{j=1}^{N} X_k^j W_k^j$$

Figure 3. Generic PF flow diagram.[13]

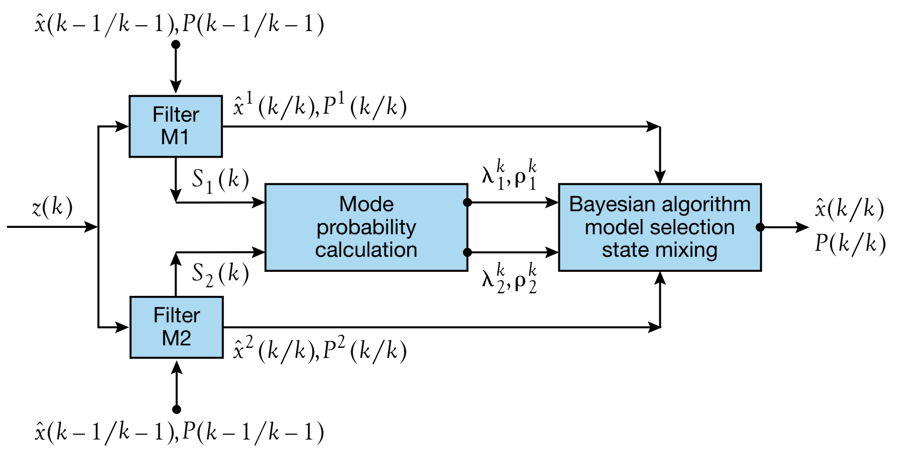

$\hat{x}(k-1/k-1), P(k-1/k-1)$
Filter M1
$\hat{x}^1(k/k), P^1(k/k)$
$S_1(k)$
$z(k)$
Mode probability calculation
λ_1^k, ρ_1^k
Bayesian algorithm model selection state mixing
$\hat{x}(k/k)$
$P(k/k)$
λ_2^k, ρ_2^k
$S_2(k)$
Filter M2
$\hat{x}^2(k/k), P^2(k/k)$
$\hat{x}(k-1/k-1), P(k-1/k-1)$

Figure 4. Example IMM architecture.[14]

filter), nonlinear filters (UKF, EKF, etc.), combinations of linear and nonlinear filters, and combinations of different system modes (dynamic models).

IMM Construction

In the multiple model approach (Figure 4), several different filter dynamics models are assumed, and all models are driven by the same measurement model. The main objective of such a methodology is to determine the dynamic model mismatch due to the changes of the system driving force. The detection of model mismatch allows the adaptive estimator to combine state estimates based on model probabilities, reducing the overall state estimation error.

IMM Selection Logic

An IMM can exercise several different selection logic paths; this article will focus on a singular method to provide a more detailed introduction. The model selection logic using the expectation and maximization (EM) technique for a two-model system (this methodology can be extended to a higher number of multiple models) is explored here and shown in Figure 5. The EM technique is developed with the use of Gaussian noise properties.[15] The following expresses the output PDF of two model linear system outputs:

$$f\{y(k)\} = \mu_1(k)N\left(y(k); m_{y_1}, \Sigma_{y_1}(k)\right) + \mu_2(k)N\left(y(k); m_{y_2}, \Sigma_{y_2}(k)\right). \tag{5}$$

The expectation and covariance of the output signal can be taken to obtain

$$\begin{aligned} m_y(k) &= \hat{y}(k) = C(k)\hat{x}(k/k) \\ \Sigma_{\tilde{y}}(k) &= \operatorname{var}(\tilde{y}(k)) = \operatorname{var}(y(k) - \hat{y}(k)) \\ &= C(k)\Sigma_{\tilde{x}}(k)C'(k) \\ f\{v(k)\} &= N(v(k); 0, \Sigma(k)) \text{ Noise PDF} \end{aligned}, \tag{6}$$

where Σ is the measurement noise covariance matrix. Applying Bayes's rule,

$$\begin{aligned} f(y/z) &= \frac{f(y)f(z/y)}{f(z)} = \frac{f(y)f(z-y)}{f(z)} = \frac{f(y)f(v)}{f(z)} \\ &= \underbrace{\frac{\{\mu_1 N(y; m_{y_1}, \Sigma_{y_1})\}\{N(v; 0, \Sigma)\}}{f(z)}}_{g(y)} + \underbrace{\frac{\{\mu_2 N(y; m_{y_2}, \Sigma_{y_2})\}\{N(v; 0, \Sigma)\}}{f(z)}}_{h(y)}. \\ &\equiv g(y) + h(y) \end{aligned} \tag{7}$$

The following expresses the mixture of output PDF from each filter:

$$f(y/z) = \sum_j \mu_j \phi_j(y), \text{ note that } \sum_j \mu_j = 1. \tag{8}$$

References 14–16 show that there exists a binary identification variable θ_j such that one and only one $\theta_j = 1$ exists, and then in that case the joint distribution $f\{y, \theta\}$ is:

$$f(y,\Theta) = \prod_j [\mu_j \phi_j(y)]^{\theta_j}, \Theta = [\theta_1\, \theta_2 \cdots \theta_n], \quad (9)$$

The model probability is represented by the variable μ. θ is the indication of whether a model is in effect and is limited to having a value of 0 or 1. The algorithm then considers a binary hypothesis receiver where there exists a likelihood ratio test. Therefore, to make a binary representation of which model the sample output comes from, define a variable ε such that

$$\varepsilon = \begin{cases} 0 & \text{If } f(y) \text{ is sampled from PDF} \quad g(y) \\ 1 & \text{If } f(y) \text{ is sampled from PDF} \quad h(y) \end{cases}. \quad (10)$$

In other words, ε is a variable that indicates whether the PDF sample is coming from the first model output or the second model output. In that case, let $\theta_g = 1 - \varepsilon$ and $\theta_h = \varepsilon$ (Ref. 15), and then

$$\begin{aligned} f(y,\varepsilon/z) &= [g(y)]^{\theta_g}[h(y)]^{\theta_h} \\ &= [g(y)]^{1-\varepsilon}[h(y)]^{\varepsilon}. \end{aligned} \quad (11)$$

Dempster, Laird, and Rubin[17] developed the EM technique to obtain the maximum likelihood estimates of the desired information using incomplete information about the data. The maximum likelihood technique requires that information be available in the form of a continuous and differentiable probability distribution. Maximizing the differentiable probability distribution produces the maximum likelihood estimates. The EM technique requires the identification of the complete data and the incomplete data. Define the set (y,ε) as the complete data set since the parameter ε provides the information about the origin of the output—that is, whether the data are coming from the model M_1 or M_2. Therefore, ε and y together complete the information about the target model. Taking the log of the preceding equation,[17]

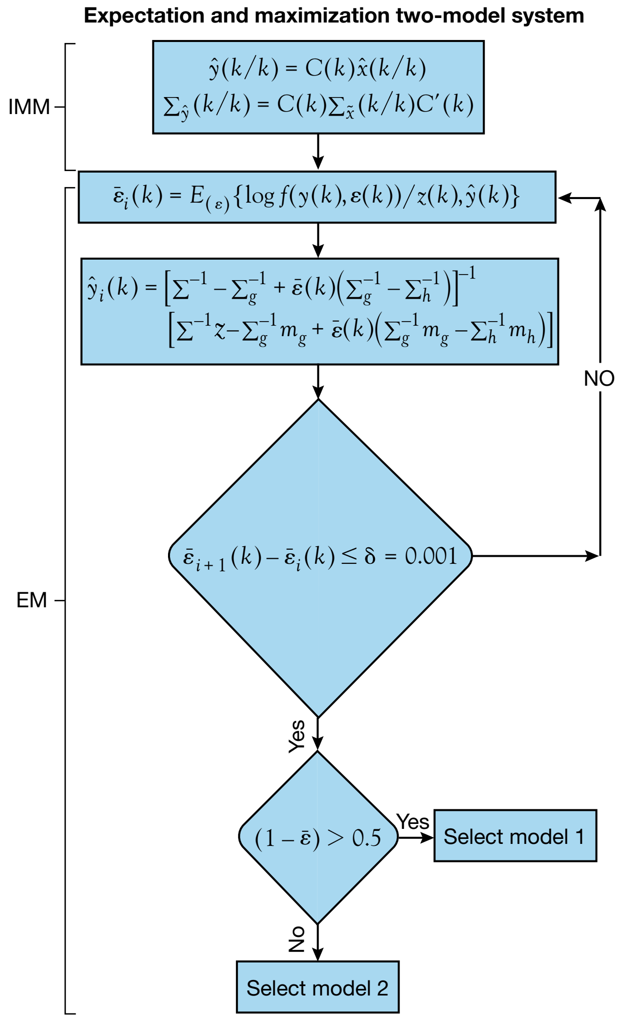

Expectation and maximization two-model system
IMM
EM
$\hat{y}(k/k) = C(k)\hat{x}(k/k)$
$\Sigma_{\hat{y}}(k/k) = C(k)\Sigma_{\tilde{x}}(k/k)C'(k)$
$\bar{\varepsilon}_i(k) = E_{(\varepsilon)}\{\log f(y(k),\varepsilon(k))/z(k),\hat{y}(k)\}$
$\hat{y}_i(k) = \left[\Sigma^{-1} - \Sigma_g^{-1} + \bar{\varepsilon}(k)\left(\Sigma_g^{-1} - \Sigma_h^{-1}\right)\right]^{-1} \left[\Sigma^{-1}z - \Sigma_g^{-1}m_g + \bar{\varepsilon}(k)\left(\Sigma_g^{-1}m_g - \Sigma_h^{-1}m_h\right)\right]$
$\bar{\varepsilon}_{i+1}(k) - \bar{\varepsilon}_i(k) \le \delta = 0.001$
NO
Yes
$(1 - \bar{\varepsilon}) > 0.5$
Yes
Select model 1
No
Select model 2

Figure 5. IMM-EM algorithm.

$$\log[f(y,\varepsilon/z)] = (1-\varepsilon)\log g(y) + \varepsilon \log h(y); \quad (12)$$

using the Gaussian properties,

$$\log g(y) = \log \mu_g + \log\frac{1}{(2\Pi)^{\frac{n}{2}}\,|\Sigma_g|^{\frac{n}{2}}} - \tfrac{1}{2}(y - m_g)'\Sigma_g^{-1}(y - m_g) + \log\frac{1}{(2\Pi)^{\frac{n}{2}}\,|\Sigma|^{\frac{n}{2}}} - \tfrac{1}{2}(z - y)'\Sigma^{-1}(z - y) - \log f(z). \quad (13)$$

Taking the gradient of the equation with respect to y produces

$$\nabla_y\{\log g(y)\} = -\Sigma_g^{-1}(y - m_g) - \Sigma^{-1}(z - y). \quad (14)$$

Substituting for the appropriate terms in $\log[f(y,\varepsilon/z)]$ and then taking the gradient with respect to y gives

$$\begin{aligned} \nabla_y\{\log[f(y,\varepsilon/z)]\} = (1-\varepsilon)&\left[-\Sigma_g^{-1}(y - m_g) - \Sigma^{-1}(z - y)\right] + \\ (\varepsilon)&\left[-\Sigma_h^{-1}(y - m_h) - \Sigma^{-1}(z - y)\right] \end{aligned}. \quad (15)$$

The EM algorithm maximization step (M-step) requires setting the preceding equation to zero and solving for the output signal y. This ends in an estimate of $(\hat{y})$ for the output signal y of

$$\hat{y} = \left[\Sigma^{-1} - \Sigma_g^{-1} + \boldsymbol{\varepsilon}\left(\Sigma_g^{-1} - \Sigma_h^{-1}\right)\right]^{-1}\left[\Sigma^{-1}z - \Sigma_g^{-1}m_g + \boldsymbol{\varepsilon}\left(\Sigma_g^{-1}m_g - \Sigma_h^{-1}m_h\right)\right]. \tag{16}$$

The preceding equation represents the desired relationship between the parameter optimization set and the missing data set. Therefore, the most general-form representation of the IMM-EM technique is

$$\hat{y}^{(p+1)} = \arg\max_{y} E_{(\boldsymbol{\varepsilon})}\{\log f(y,\boldsymbol{\varepsilon})/z,\hat{y}^{(p)}\}, \tag{17}$$

where E is the expectation and arg max is the maximization. The EM technique as described in Refs. 17 and 18 requires the iterative process to continue until the convergence of the optimized parameter set. Note that the model identification probability $\boldsymbol{\varepsilon}$ or the missing data is a discrete, random variable whose value is in range of discrete values $0 \le \boldsymbol{\varepsilon} \le 1$. Therefore, monitor $\boldsymbol{\varepsilon}$ until its convergence toward a value between zero and one. Once $\boldsymbol{\varepsilon}$ has stabilized to a value, then declare the end of iterations.

The algorithm requires the target state combination model outputs

$$\begin{gathered}\hat{x}(k/k) = \sum_{j=1}^{r} \hat{x}^j(k/k)\mu_j(k) \\ \Sigma_{\tilde{x}}(k/k) = \sum_{j}^{r} \mu_j(k)\left\{\Sigma_{\tilde{x}^j}(k/k) + [\hat{x}^j(k/k) - \hat{x}(k/k)][\hat{x}^j(k/k) - \hat{x}(k/k)]'\right\}\end{gathered}, \tag{18}$$

where μ is the model probability.

ANALYSIS EXAMPLE UKF IMM

IMM Example Setup

As a simple example, the tracking of an object in a ballistic path is explored. The sensor model will provide an east-north-up position measurement of the object and will be operating at a high rate. The filter approach chosen for the example will be a highlighted application of an advanced algorithm, with the implementation of an IMM that contains three separate UKFs. The three filters assessed are a constant velocity UKF, a constant acceleration UKF, and a constant jerk UKF. All three filters are initialized to the position and velocity covariance estimate provided from the sensor. The constant velocity filter is tuned to track an object with a constant acceleration that experiences a slight variation in acceleration. The constant acceleration UKF is tuned to track an object with a constant acceleration that undergoes a slight variation in acceleration. The constant jerk filter is tuned to track an object with a constant velocity that experiences a slight variation in acceleration or jerk. Generally, the tuning of Kalman filters is achieved by a priori knowledge of the process covariance matrix and the measurement noise covariance matrix that are appropriate for the target characteristics. The output of the example (Figure 6) will show the position error, velocity error, filter performance metrics, and UKF probability (mixing probability). Upper bound and lower bound are computed relative to the number of Monte Carlo runs as defined in Ref. 14, with 25 Monte Carlo runs and three degrees of freedom.

Performance Metrics

If target truth is available, then the normalized estimation error squared (NEES) is defined as

$$(X_t - X_e)'P^{-1}(X_t - X_e). \tag{19}$$

NEES has a chi-squared distribution depending on the number of Monte Carlo runs; it has an upper and lower limit. If NEES is maintained between these two limits, generally, the filter model performance is satisfactory. Normalized innovation squared (NIS) is defined as

$$\begin{gathered}\boldsymbol{\nu}'S^{-1}\boldsymbol{\nu} \\ S = H_{k+1}P_{k+1|k}H_{k+1}^T + R_{k+1}\end{gathered}, \tag{20}$$

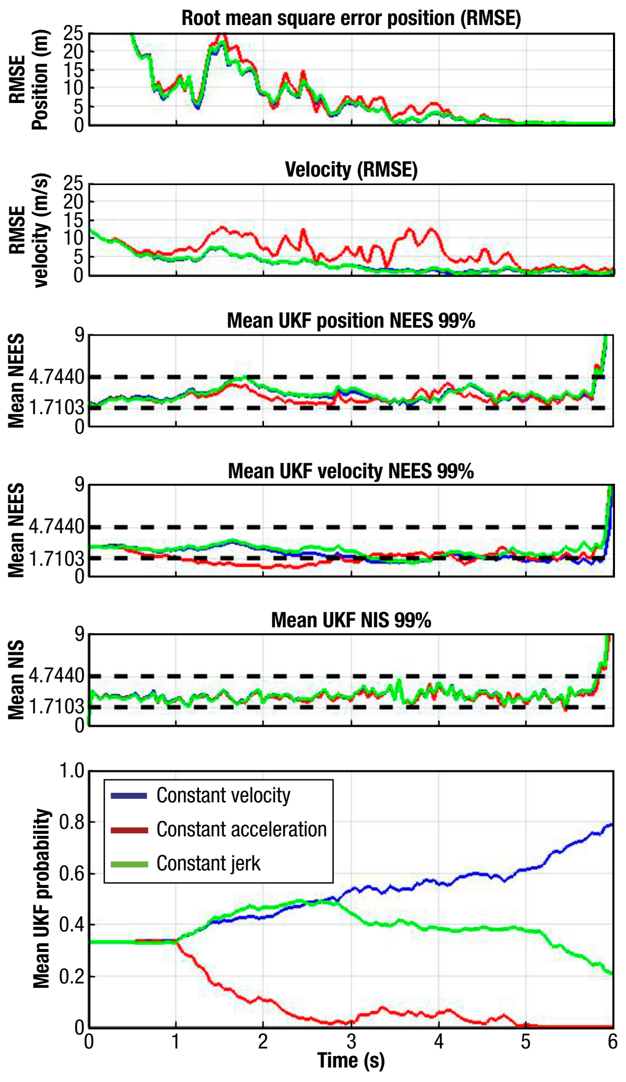


Figure 6. IMM UKF example results.

where H is a measurement vector, P is covariance matrix, and R is the measurement covariance matrix.

NIS also has a chi-squared distribution depending on the number of Monte Carlo runs; it has an upper and lower limit. If NIS is maintained between these two limits, generally, the filter model performance is satisfactory. Both NEES and NIS are best maintained between their respective upper and lower limits. The upper limit in the bounds of NIS and NEES is a measure of the ability of the estimate to be contained within the covariance. If the NIS/NEES values are lower than the upper bound, it will ensure that the covariance contains the values, which is a measure of the accuracy of the estimate of the error or innovation within the filter. The lower limit in the bounds of NIS and NEES is a measure of the ability of the filter to have a tight estimate of the covariance, meaning that the covariance is not arbitrarily large to ensure that it contains the error or innovation. If the NIS and NEES values are within the bounds, they will add robustness to the filter while ensuring that the covariance estimate is accurately bounded.

Accumulate N innovations $S_1 = \sum_1^N \nu$; accumulate N innovation squared $S_2 = \sum_1^N \nu\, \nu'$; compute N sample innovation squared $S_3 = (S_1)(S_1)'$; compute the innovation covariance $S = \frac{S_2 - \frac{S_3}{N}}{N}$; compute power density $\mu = \nu' S^{-1} \nu$; compute model Gaussian density function $\lambda = \frac{e^{-\frac{1}{2}(\mu)}}{\left(\frac{1}{2}\pi\right)^{\frac{n}{2}} |S|^{\frac{1}{2}}}$, where n is the size of the innovation vector ν; compute the normalized model probability $p_k = \frac{\lambda * p_{k-1}}{\sum_1^m (\lambda * p_{k-1})}$, where m is the number of models; and blend target state estimate $x = \sum_1^m (p_k^m * x^m)$.

Figure 7. Technique for computing model probability.[14]

A technique for computing model probability uses the multivariate Gaussian density function properties and accumulates 5–10 innovation vectors. Adaptive window size might be more appropriate for tracking targets that exhibit unpredictable characteristics. The technique completes the analysis shown in Figure 7.

IMM Results

A MATLAB simulation was developed to have an IMM container class managing three UKF models: constant velocity, constant acceleration, and constant jerk. The results track an object flying a ballistic profile within the atmosphere. The measurement model generates noisy position measurements that are applied to all filter models (Figure 6). The simulation ran at a 10-Hz data rate and summarizes data for 25 Monte Carlo runs.

The NEES bounds indicate satisfactory position estimate performance and somewhat degraded velocity estimate performance; this indicates the need to adjust the velocity process noise matrix. The NIS bounds indicate satisfactory tuning parameters of the measurement noise matrix for all UKF models. The model probability assignment indicates the constant velocity is the most probable model to match the target trajectory. This example demonstrates how different dynamic models can be set into a single IMM architecture and mixed to provide optimal performance. In expanded analysis, if the flight characteristics of the object changed, then the IMM would adapt and select the filters accordingly to mix the output states.

CONCLUDING REMARKS

This article has provided an introduction to nonlinear filters and IMM logic and extensive references for future applications. It highlights some of the challenges and considerations in developing tracking filters to counter increasing threat capability. The evolution is highlighted by the extension and application of advanced filter algorithms, which include nonlinear filters such as the UKF, EKF, and PF. The increase in computational capability allows for real-world application of multiple model filtering methods, which are most commonly implemented in the form of an IMM. The IMM model allows a system to determine dynamic model mismatch due to the changes of the system driving force (or threat variation) and results in an adaptive estimator by combining state estimates (based on model probabilities) that reduce the overall state estimation error. The development of multiple model filtering with nonlinear filters increases system robustness while allowing for future adaptations. IMM models can be updated to add additional filters or dynamic models, as threat capabilities increase, to provide increased performance for a system that was designed with prior knowledge. The ability of a system to evolve and adapt without a complete redesign provides a useful architecture that can be expanded as threats become more challenging.

REFERENCES

[1]L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," *IEEE Trans. Automat. Contr.*, vol. 24, no. 1, pp. 36–50, 1979.

[2]M. Hoshiya and E. Saito, "Structural identification by extended Kalman filter," *J. Eng. Mech.*, vol. 110, no. 12, pp. 1757–1770, 1984.

[3]Y. Song and J. W. Grizzle, "The extended Kalman filter as a local asymptotic observer for nonlinear discrete-time systems," in *Proc. 1992 American Control Conf.*, Chicago, IL, pp. 3365–3369, 1992.

[4]K. Reif, S. Gunther, E. Yaz, and R. Unbehauen, "Stochastic stability of the discrete-time extended Kalman filter," *IEEE Trans. Automat. Contr.*, vol. 44, no. 4, pp. 714–728, 1999.

[5]J. K. Sasiadek, Q. Wang, and M. B. Zeremba, "Fuzzy adaptive Kalman filtering for INS/GPS data fusion," in *Proc. 2000 IEEE Int. Symp. on Intelligent Control*, Rio Patras, Greece, pp. 181–186, 2000.

[6]S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *Proc. 1995 American Control Conf.*, Seattle, WA, pp. 1628–1632, 1995.

[7]S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, Proc. SPIE, Vol. 3068, I. Kadar, Ed., SPIE, Bellingham, WA, pp. 182–193, 1997.

[8]E. A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. Adaptive Systems for Signal Processing, Communications, and Control Symp.*, Lake Louise, Alberta, Canada, pp. 153–158, 2000.

[9]E. A. Wan and R. van der Merwe, "The square-root unscented Kalman filter for state and parameter-estimation," in *Proc. 2001 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Salt Lake City, UT, pp. 3461–3464, 2001.

[10]S. J. Julier, "The scaled unscented transformation," in *Proc. American Control Conf.*, Anchorage, AK, pp. 4555–4559, 2002.

[11]S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proc. IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[12]J. Stauch and M. Jah, "Unscented Schmidt-Kalman filter algorithm," *J. Guid. Control Dyn.*, vol. 38, no. 1, pp. 117–123, 2015.

[13]M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, 2002.

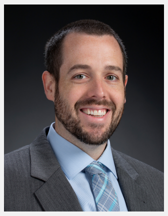
[14]Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, 1995.

[15]A. Singhal and D. E. Seborg, "Dynamic data rectification using the expectation and maximization algorithm," *AIChE J.*, vol. 46, no. 8, pp. 1556–1565, 2000.

[16]S. Lei, L. Weihua, and L. Zuoliang, "An effective IMM algorithm for maneuvering target tracking," in *Proc. 1996 3rd Int. Conf. Signal Processing*, Beijing, China, pp. 922–925, 1996.

[17]A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood estimation from incomplete data via the EM algorithm," *J. R. Stat. Soc. [Ser A]*, vol. 39, no. 1, pp. 1–38, 1977.

[18]G. W. Pulford and A. Logothetis, "An expectation-maximization tracking for multiple observations of a single target in clutter," in *Proc. IEEE Conf. on Decision and Control*, San Diego, CA, pp. 4997–5003, 1997.



Scott A. Hays, Air and Missile Defense Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Scott A. Hays is a Senior Professional Staff member and an assistant program manager in APL's Air and Missile Defense Sector. He has a BS in mechanical engineering and a BS, an MS, and a PhD in aerospace engineering, all from North Carolina State University. During his more than 8 years at APL, Scott has led and contributed to projects for a variety of new concepts and techniques in systems engineering, aerodynamic estimation, advanced filter application, test and evaluation of weapon systems, and propagation/future state estimation. His email address is scott.hays@jhuapl.edu.

Michael A. Fatemi, Air and Missile Defense Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Michael A. Fatemi is a Senior Professional Staff member in the Guidance, Navigation, and Control Group in APL's Air and Missile Defense Sector. He has a BA in engineering science from Cambridge University, an MS in electrical and computer science from Johns Hopkins University, and a PhD in electrical engineering from Catholic University of America. His primary responsibilities include analysis, modeling, and simulation development of guidance algorithms and control systems. His email address is michael.fatemi@jhuapl.edu.