Kalman Filtering and Neural Networks, Edited by Simon Haykin
Copyright © 2001 John Wiley & Sons, Inc.
ISBNs: 0-471-36998-5 (Hardback); 0-471-22154-6 (Electronic)

7

THE UNSCENTED KALMAN FILTER

Eric A. Wan and Rudolph van der Merwe
Department of Electrical and Computer Engineering, Oregon Graduate Institute of Science and Technology, Beaverton, Oregon, U.S.A.

7.1 INTRODUCTION

In this book, the *extended Kalman filter* (EKF) has been used as the standard technique for performing recursive nonlinear estimation. The EKF algorithm, however, provides only an *approximation* to optimal nonlinear estimation. In this chapter, we point out the underlying assumptions and flaws in the EKF, and present an alternative filter with performance superior to that of the EKF. This algorithm, referred to as the *unscented Kalman filter* (UKF), was first proposed by Julier et al. [1–3], and further developed by Wan and van der Merwe [4–7].

The basic difference between the EKF and UKF stems from the manner in which Gaussian random variables (GRV) are represented for propagating through system dynamics. In the EKF, the state distribution is approximated by a GRV, which is then propagated analytically through the first-order linearization of the nonlinear system. This can introduce large errors in the true posterior mean and covariance of the transformed GRV, which may lead to suboptimal performance and sometimes divergence of the filter. The UKF address this problem by using a deterministic sampling approach. The state distribution is again approximated by a GRV, but is now represented using a minimal set of carefully chosen sample points. These sample points completely capture the true mean and covariance of the GRV, and, when propagated through the *true* nonlinear system, captures the posterior mean and covariance accurately to second order (Taylor series expansion) for *any* nonlinearity. The EKF, in contrast, only achieves first-order accuracy. No explicit Jacobian or Hessian calculations are necessary for the UKF. Remarkably, the computational complexity of the UKF is the same order as that of the EKF.

Julier and Uhlman demonstrated the substantial performance gains of the UKF in the context of state estimation for nonlinear control. A number of theoretical results were also derived. This chapter reviews this work, and presents extensions to a broader class of nonlinear estimation problems, including nonlinear system identification, training of neural networks, and dual estimation problems. Additional material includes the development of an unscented Kalman *smoother* (UKS), specification of efficient recursive *square-root* implementations, and a novel use of the UKF to improve *particle filters* [6].

In presenting the UKF, we shall cover a number of application areas of nonlinear estimation in which the EKF has been applied. General application areas may be divided into *state estimation*, *parameter estimation* (e.g., learning the weights of a neural network), and *dual estimation* (e.g., the expectation–maximization (EM) algorithm). Each of these areas place specific requirements on the UKF or EKF, and will be developed in turn. An overview of the framework for these areas is briefly reviewed next.

State Estimation The basic framework for the EKF involves estimation of the state of a discrete-time nonlinear dynamical system,

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k), \tag{7.1}$$

$$\mathbf{y}_k = \mathbf{H}(\mathbf{x}_k, \mathbf{n}_k), \tag{7.2}$$

where $\mathbf{x}_k$ represents the unobserved state of the system, $\mathbf{u}_k$ is a known exogeneous input, and $\mathbf{y}_k$ is the observed measurement signal. The *process* *noise* $\mathbf{v}_k$ drives the dynamic system, and the *observation noise* is given by $\mathbf{n}_k$. Note that we are not assuming additivity of the noise sources. The system dynamical model $\mathbf{F}$ and $\mathbf{H}$ are assumed known. A simple block diagram of this system is shown in Figure 7.1. In state estimation, the EKF is the standard method of choice to achieve a recursive (approximate) maximum-likelihood estimate of the state $\mathbf{x}_k$. For completeness, we shall review the EKF and its underlying assumptions in Section 7.2 to help motivate the presentation of the UKF for state estimation in Section 7.3.

Parameter Estimation Parameter estimation, sometimes referred to as system identification or machine learning, involves determining a nonlinear mapping

$$\mathbf{y}_k = \mathbf{G}(\mathbf{x}_k, \mathbf{w}), \tag{7.3}$$

where $\mathbf{x}_k$ is the input, $\mathbf{y}_k$ is the output, and the nonlinear map $\mathbf{G}(\cdot)$ is parameterized by the vector $\mathbf{w}$. The nonlinear map, for example, may be a feedforward or recurrent neural network ($\mathbf{w}$ are the weights), with numerous applications in regression, classification, and dynamic modeling. Learning corresponds to estimating the parameters $\mathbf{w}$. Typically, a training set is provided with sample pairs consisting of known input and desired outputs, $\{\mathbf{x}_k, \mathbf{d}_k\}$. The error of the machine is defined as $\mathbf{e}_k = \mathbf{d}_k - \mathbf{G}(\mathbf{x}_k, \mathbf{w})$, and the goal of learning involves solving for the parameters $\mathbf{w}$ in order to minimize the expectation of some given function of the error.

While a number of optimization approaches exist (e.g., gradient descent using backpropagation), the EKF may be used to estimate the parameters by writing a new state-space representation,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{r}_k, \tag{7.4}$$
$$\mathbf{d}_k = \mathbf{G}(\mathbf{x}_k, \mathbf{w}_k) + \mathbf{e}_k, \tag{7.5}$$

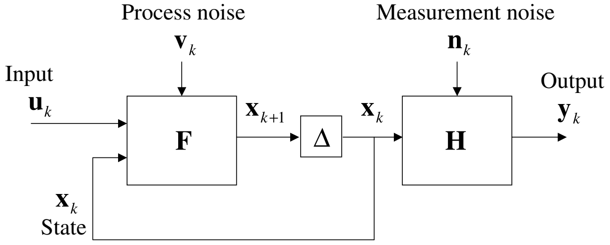


Figure 7.1 Discrete-time nonlinear dynamical system.

where the parameters $\mathbf{w}_k$ correspond to a stationary process with identity state transition matrix, driven by process noise $\mathbf{r}_k$ (the choice of variance determines convergence and tracking performance and will be discussed in further detail in Section 7.4). The output $\mathbf{d}_k$ corresponds to a nonlinear observation on $\mathbf{w}_k$. The EKF can then be applied directly as an efficient "second-order" technique for learning the parameters. The use of the EKF for training neural networks has been developed by Singhal and Wu [8] and Puskorious and Feldkamp [9], and is covered in Chapter 2 of this book. The use of the UKF in this role is developed in Section 7.4.

Dual Estimation A special case of machine learning arises when the input $\mathbf{x}_k$ is unobserved, and requires coupling both state estimation and parameter estimation. For these *dual estimation* problems, we again consider a discrete-time nonlinear dynamical system,

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k, \mathbf{w}), \tag{7.6}$$

$$\mathbf{y}_k = \mathbf{H}(\mathbf{x}_k, \mathbf{n}_k, \mathbf{w}), \tag{7.7}$$

where both the system states $\mathbf{x}_k$ and the set of model parameters $\mathbf{w}$ for the dynamical system must be simultaneously estimated from only the observed noisy signal $\mathbf{y}_k$. Example applications include adaptive nonlinear control, noise reduction (e.g., speech or image enhancement), determining the underlying price of financial time series, etc. A general theoretical and algorithmic framework for dual Kalman-based estimation has been presented in Chapter 5. An expectation–maximization approach has also been covered in Chapter 6. Approaches to dual estimation utilizing the UKF are developed in Section 7.5.

In the next section, we review optimal estimation to explain the basic assumptions and flaws with the EKF. This will motivate the use of the UKF as a method to amend these flaws. A detailed development of the UKF is given in Section 7.3. The remainder of the chapter will then be divided based on the application areas reviewed above. We conclude the chapter in Section 7.6 with the *unscented particle filter*, in which the UKF is used to improve sequential Monte-Carlo-based filtering methods. Appendix A provides a derivation of the accuracy of the UKF. Appendix B details an efficient *square-root* implementation of the UKF.

7.2 OPTIMAL RECURSIVE ESTIMATION AND THE EKF

Given observations $\mathbf{y}_k$, the goal is to estimate the state $\mathbf{x}_k$. We make no assumptions about the nature of the system dynamics at this point. The optimal estimate in the minimum mean-squared error (MMSE) sense is given by the conditional mean:

$$\hat{\mathbf{x}}_k = \mathbb{E}[\mathbf{x}_k|\mathbf{Y}_0^k], \tag{7.8}$$

where $\mathbf{Y}_0^k$ is the sequence of observations up to time k. Evaluation of this expectation requires knowledge of the *a posteriori* density $p(\mathbf{x}_k|\mathbf{Y}_0^k)$.[1] Given this density, we can determine not only the MMSE estimator, but any "best" estimator under a specified performance criterion. The problem of determining the *a posteriori* density is in general referred to as the Bayesian approach, and can be evaluated recursively according to the following relations:

$$p(\mathbf{x}_k|\mathbf{Y}_0^k) = \frac{p(\mathbf{x}_k|\mathbf{Y}_0^{k-1})p(\mathbf{y}_k|\mathbf{x}_k)}{p(\mathbf{y}_k|\mathbf{Y}_0^{k-1})}, \tag{7.9}$$

where

$$p(\mathbf{x}_k|\mathbf{Y}_0^{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{Y}_0^{k-1})\,d\mathbf{x}_{k-1}, \tag{7.10}$$

and the normalizing constant $p(\mathbf{y}_k|\mathbf{Y}_0^k)$ is given by

$$p(\mathbf{y}_k|\mathbf{Y}_0^{k-1}) = \int p(\mathbf{x}_k|\mathbf{Y}_0^{k-1})p(\mathbf{y}_k|\mathbf{x}_k)\,d\mathbf{x}_k. \tag{7.11}$$

This recursion specifies the current state density as a function of the previous density and the most recent measurement data. The state-space model comes into play by specifying the state transition probability $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and measurement probability or likelihood, $p(\mathbf{y}_k|\mathbf{x}_x)$. Specifically, $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is determined by the process noise density $p(\mathbf{v}_k)$ with the state-update equation

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k). \tag{7.12}$$

For example, given an additive noise model with Gaussian density, $p(\mathbf{v}_k) = \mathcal{N}(0, \mathbf{R}^\mathrm{v})$, then $p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{F}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}), \mathbf{R}^\mathrm{v})$. Similarly,

[1]Note that we do not write the implicit dependence on the observed input $\mathbf{u}_k$, since it is not a random variable.

$p(\mathbf{y}_k|\mathbf{x}_x)$ is determined by the observation noise density $p(\mathbf{n}_k)$ and the measurement equation

$$\mathbf{y}_k = \mathbf{H}(\mathbf{x}_k, \mathbf{n}_k). \tag{7.13}$$

In principle, knowledge of these densities and the initial condition $p(\mathbf{x}_0|\mathbf{y}_0) = p(\mathbf{y}_0|\mathbf{x}_0)p(\mathbf{x}_0)/p(\mathbf{y}_0)$ determines $p(\mathbf{x}_k|\mathbf{Y}_0^k)$ for all k. Unfortunately, the multidimensional integration indicated by Eqs. (7.9)–(7.11) makes a closed-form solution intractable for most systems. The only general approach is to apply Monte Carlo sampling techniques that essentially convert integrals to finite sums, which converge to the true solution in the limit. The *particle filter* discussed in the last section of this chapter is an example of such an approach.

If we make the basic assumption that all densities remain Gaussian, then the Bayesian recursion can be greatly simplified. In this case, only the conditional mean $\hat{\mathbf{x}}_k = \mathbb{E}[\mathbf{x}_k|\mathbf{Y}_0^k]$ and covariance $\mathbf{P}_{\mathbf{x}_k}$ need to be evaluated. It is straightforward to show that this leads to the recursive estimation

$$\hat{\mathbf{x}}_k = (\text{prediction of } \mathbf{x}_k) + \mathcal{K}_k[\mathbf{y}_k - (\text{prediction of } \mathbf{y}_k)], \tag{7.14}$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathcal{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k} \mathcal{K}_k^T. \tag{7.15}$$

While this is a *linear* recursion, we have not assumed linearity of the model. The optimal terms in this recursion are given by

$$\hat{\mathbf{x}}_k^- = \mathbb{E}[\mathbf{F}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{v}_{k-1})], \tag{7.16}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k}^{-1}, \tag{7.17}$$

$$\hat{\mathbf{y}}_k^- = \mathbb{E}[\mathbf{H}(\mathbf{x}_k^-, \mathbf{n}_k)], \tag{7.18}$$

where the optimal prediction (i.e., prior mean) of $\mathbf{x}_k$ is written as $\hat{\mathbf{x}}_k^-$, and corresponds to the expectation of a nonlinear function of the random variables $\mathbf{x}_{k-1}$ and $\mathbf{v}_{k-1}$ (with a similar interpretation for the optimal prediction $\hat{\mathbf{y}}_k^-$). The optimal gain term $\mathcal{K}_k$ is expressed as a function of posterior covariance matrices (with $\tilde{\mathbf{y}}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k^-$). Note that evaluation of the covariance terms also require taking expectations of a nonlinear function of the prior state variable. $\mathbf{P}_{\mathbf{x}_k}^-$ is the prediction of the covariance of $\mathbf{x}_k$, and $\mathbf{P}_{\tilde{\mathbf{y}}_k}$ is the covariance of $\tilde{\mathbf{y}}_k$.

The celebrated Kalman filter [10] calculates all terms in these equations exactly in the linear case, and can be viewed as an efficient method for analytically propagating a GRV through linear system dynamics. For nonlinear models, however, the EKF *approximates* the optimal terms as

$$\hat{\mathbf{x}}_k^- \approx \mathbf{F}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \bar{\mathbf{v}}), \tag{7.19}$$

$$\mathcal{K}_k \approx \hat{\mathbf{P}}_{\mathbf{x}_k\mathbf{y}_k}\hat{\mathbf{P}}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}^{-1}, \tag{7.20}$$

$$\hat{\mathbf{y}}_k^- \approx \mathbf{H}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}}), \tag{7.21}$$

where predictions are approximated simply as functions of the prior *mean* value (no expectation taken).[2] The covariances are determined by linearizing the dynamical equations ($\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{B}_u\mathbf{u}_k + \mathbf{B}\mathbf{v}_k$, $\mathbf{y}_k \approx \mathbf{C}\mathbf{x}_\mathbf{k} + \mathbf{D}\mathbf{n}_k$), and then determining the posterior covariance matrices analytically for the linear system. In other words, in the EKF, the state distribution is approximated by a GRV, which is then propagated analytically through the "first-order" linearization of the nonlinear system. The explicit equations for the EKF are given in Table 7.1. As such, the EKF

Table 7.1 Extended Kalman filter (EKF) equations

Initialize with

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \tag{7.22}$$

$$\mathbf{P}_{\mathbf{x}_0} = \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]. \tag{7.23}$$

For $k \in \{1, \ldots, \infty\}$, the time-update equations of the extended Kalman filter are

$$\hat{\mathbf{x}}_k^- = \mathbf{F}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \bar{\mathbf{v}}), \tag{7.24}$$

$$\mathbf{P}_{\mathbf{x}_k}^- = \mathbf{A}_{k-1}\mathbf{P}_{\mathbf{x}_{k-1}}\mathbf{A}_{k-1}^T + \mathbf{B}_k\mathbf{R}^\mathbf{v}\mathbf{B}_k^T, \tag{7.25}$$

and the measurement-update equations are

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k}^-\mathbf{C}_k^T(\mathbf{C}_k\mathbf{P}_{\mathbf{x}_k}^-\mathbf{C}_k^T + \mathbf{D}_k\mathbf{R}^\mathbf{n}\mathbf{D}_k^T)^{-1}, \tag{7.26}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k[\mathbf{y}_k - \mathbf{H}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}})], \tag{7.27}$$

$$\mathbf{P}_{\mathbf{x}_k} = (\mathbf{I} - \mathcal{K}_k\mathbf{C}_k)\mathbf{P}_{\mathbf{x}_k}^-, \tag{7.28}$$

where

$$\begin{aligned} \mathbf{A}_k &\triangleq \left.\frac{\partial \mathbf{F}(\mathbf{x}, \mathbf{u}_k, \bar{\mathbf{v}})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k}, & \mathbf{B}_k &\triangleq \left.\frac{\partial \mathbf{F}(\hat{\mathbf{x}}_k^-, \mathbf{u}_k, \mathbf{v})}{\partial \mathbf{v}}\right|_{\bar{\mathbf{v}}}, \\ \mathbf{C}_k &\triangleq \left.\frac{\partial \mathbf{H}(\mathbf{x}, \bar{\mathbf{n}})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k}, & \mathbf{D}_k &\triangleq \left.\frac{\partial \mathbf{H}(\hat{\mathbf{x}}_k^-, \mathbf{n})}{\partial \mathbf{n}}\right|_{\bar{\mathbf{n}}}, \end{aligned} \tag{7.29}$$

and where $\mathbf{R}^\mathbf{v}$ and $\mathbf{R}^\mathbf{n}$ are the covariances of $\mathbf{v}_k$ and $\mathbf{n}_k$, respectively.

[2]The noise means are denoted by $\bar{\mathbf{n}} = \mathbb{E}[\mathbf{n}]$ and $\bar{\mathbf{v}} = \mathbb{E}[\mathbf{v}]$, and are usually assumed to equal zero.

can be viewed as providing "first-order" approximations to the optimal terms.[3] These approximations, however, can introduce large errors in the true posterior mean and covariance of the transformed (Gaussian) random variable, which may lead to suboptimal performance and sometimes divergence of the filter.[4] It is these "flaws" that will be addressed in the next section using the UKF.

7.3 THE UNSCENTED KALMAN FILTER

The UKF addresses the approximation issues of the EKF. The state distribution is again represented by a GRV, but is now specified using a minimal set of carefully chosen sample points. These sample points completely capture the true mean and covariance of the GRV, and when propagated through the *true* nonlinear system, capture the posterior mean and covariance accurately to the second order (Taylor series expansion) for *any* nonlinearity. To elaborate on this, we begin by explaining the *unscented transformation*.

Unscented Transformation The unscented transformation (UT) is a method for calculating the statistics of a random variable which undergoes a nonlinear transformation [3]. Consider propagating a random variable $\mathbf{x}$ (dimension L) through a nonlinear function, $\mathbf{y} = f(\mathbf{x})$. Assume $\mathbf{x}$ has mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P_x}$. To calculate the statistics of $\mathbf{y}$, we form a matrix $\boldsymbol{\mathcal{X}}$ of $2L + 1$ *sigma* vectors $\mathcal{X}_i$ according to the following:

$$\begin{aligned}
\mathcal{X}_0 &= \bar{\mathbf{x}}, \\
\mathcal{X}_i &= \bar{\mathbf{x}} + \left(\sqrt{(L+\lambda)\mathbf{P_x}}\right)_i, \qquad i = 1, \ldots, L, \\
\mathcal{X}_i &= \bar{\mathbf{x}} - \left(\sqrt{(L+\lambda)\mathbf{P_x}}\right)_{i-L}, \qquad i = L+1, \ldots, 2L,
\end{aligned} \tag{7.30}$$

[3]While "second-order" versions of the EKF exist, their increased implementation and computational complexity tend to prohibit their use.

[4]A popular technique to improve the "first-order" approach is the *iterated EKF*, which effectively iterates the EKF equations at the current time step by redefining the nominal state estimate and re-linearizing the measurement equations. It is capable of providing better performance than the basic EKF, especially in the case of significant nonlinearity in the measurement function [11]. We have not performed a comparison to the UKF at this time, though a similar procedure may also be adapted to iterate the UKF.

where $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter. The constant α determines the spread of the sigma points around $\bar{\mathbf{x}}$, and is usually set to a small positive value (e.g., $1 \le \alpha \le 10^{-4}$). The constant κ is a secondary scaling parameter, which is usually set to $3 - L$ (see [1] for details), and β is used to incorporate prior knowledge of the distribution of $\mathbf{x}$ (for Gaussian distributions, $\beta = 2$ is optimal). $(\sqrt{(L+\lambda)\mathbf{P}_\mathbf{x}})_i$ is the ith column of the matrix square root (e.g., lower-triangular Cholesky factorization). These sigma vectors are propagated through the nonlinear function

$$\mathcal{Y}_i = f(\mathcal{X}_i), \qquad i = 0, \ldots, 2L, \tag{7.31}$$

and the mean and covariance for $\mathbf{y}$ are approximated using a weighted sample mean and covariance of the posterior sigma points,

$$\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_i, \tag{7.32}$$

$$\mathbf{P}_\mathbf{y} \approx \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_i - \bar{\mathbf{y}})(\mathcal{Y}_i - \bar{\mathbf{y}})^T, \tag{7.33}$$

with weights W_i given by

$$\begin{aligned} W_0^{(m)} &= \frac{\lambda}{L+\lambda}, \\ W_0^{(c)} &= \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta \\ W_i^{(m)} &= W_i^{(c)} = \frac{1}{2(L+\lambda)}, \qquad i = 1, \ldots, 2L. \end{aligned} \tag{7.34}$$

A block diagram illustrating the steps in performing the UT is shown in Figure 7.2. Note that this method differs substantially from general Monte Carlo sampling methods, which require orders of magnitude more sample points in an attempt to propagate an accurate (possibly non-Gaussian) distribution of the state. The deceptively simple approach taken with the UT results in approximations that are accurate to the third order for Gaussian inputs for all nonlinearities. For non-Gaussian inputs, approximations are accurate to at least the second order, with the accuracy of third- and higher-order moments being determined by the choice of α and β. The proof of this is provided in Appendix A. Valuable insight into the UT can also be gained by relating it to a numerical technique called

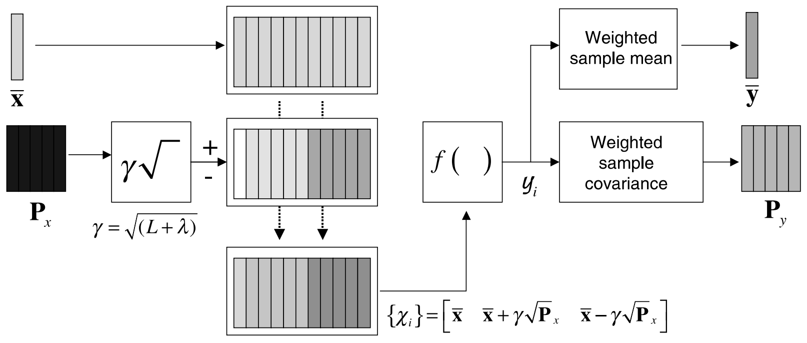

$\bar{\mathbf{x}}$
$\mathbf{P}_x$
$\gamma\sqrt{\ }$
$\gamma = \sqrt{(L+\lambda)}$
+
-
$f(\ )$
$\mathcal{Y}_i$
Weighted sample mean
Weighted sample covariance
$\bar{\mathbf{y}}$
$\mathbf{P}_y$
$\{\chi_i\} = \left[\bar{\mathbf{x}} \quad \bar{\mathbf{x}} + \gamma\sqrt{\mathbf{P}_x} \quad \bar{\mathbf{x}} - \gamma\sqrt{\mathbf{P}_x}\right]$

Figure 7.2 Block diagram of the UT.

Gaussian quadrature numerical evaluation of integrals. Ito and Xiong [12] recently showed the relation between the UT and the *Gauss–Hermite* quadrature rule[5] in the context of state estimation. A close similarity also exists between the UT and the *central difference interpolation* filtering (CDF) techniques developed separately by Ito and Xiong [12] and Nørgaard, Poulsen, and Ravn [13]. In [7] van der Merwe and Wan show how the UKF and CDF can be unified in a general family of *derivative-free* Kalman filters for nonlinear estimation.

A simple example is shown in Figure 7.3 for a two-dimensional system: Figure 7.3*a* shows the true mean and covariance propagation using Monte Carlo sampling; Figure 7.3*b* shows the results using a linearization approach as would be done in the EKF; Figure 7.3*c* shows the performance of the UT (note that only five sigma points are required). The superior performance of the UT is clear.

Unscented Kalman Filter The *unscented Kalman filter* (UKF) is a straightforward extension of the UT to the recursive estimation in Eq. (7.14), where the state RV is redefined as the concatenation of the original state and noise variables: $\mathbf{x}_k^a = [\mathbf{x}_k^T \quad \mathbf{v}_k^T \quad \mathbf{n}_k^T]^T$. The UT sigma point selection scheme, Eq. (7.30), is applied to this new augmented state RV to calculate the corresponding sigma matrix, $\mathcal{X}_k^a$. The UKF equations are

[5]In the scalar case, the Gauss–Hermite rule is given by $\int_{-\infty}^{\infty} f(x)(2\pi)^{-1/2} e^{-x^2}\,dx = \sum_{i=1}^{m} w_i f(x_i)$, where the equality holds for all polynomials, $f(\cdot)$, of degree up to $2m-1$ and the quadrature points x_i and weights w_i are determined according to the rule type (see [12] for details). For higher dimensions, the Gauss–Hermite rule requires on the order of m^L functional evaluations, where L is the dimension of the state. For the scalar case, the UT with $\alpha = 1$, $\beta = 0$, and $\kappa = 2$ coincides with the three-point Gauss–Hermite quadrature rule.

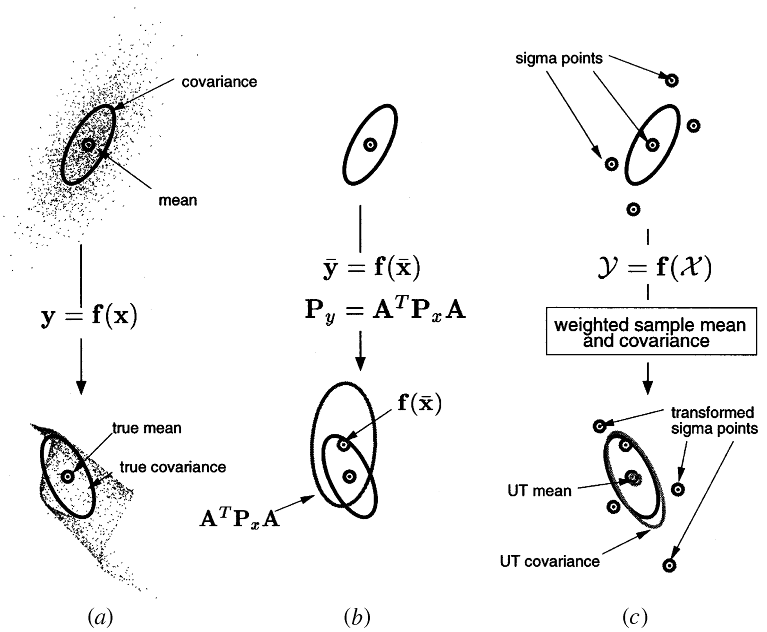

covariance
mean
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
true mean
true covariance
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T\mathbf{P}_x\mathbf{A}$
$\mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{A}^T\mathbf{P}_x\mathbf{A}$
sigma points
$\mathcal{Y} = \mathbf{f}(\mathcal{X})$
weighted sample mean and covariance
transformed sigma points
UT mean
UT covariance
(a)
(b)
(c)

Figure 7.3 Example of the UT for mean and covariance propagation: (*a*) actual; (*b*) first-order linearization (EFK); (*c*) UT.

given in Table 7.2. Note that no explicit calculations of Jacobians or Hessians are necessary to implement this algorithm. Furthermore, the overall number of computations is of the same order as the EKF.

Implementation Variations For the special (but often encountered) case where the process and measurement noise are purely additive, the computational complexity of the UKF can be reduced. In such a case, the system state need not be augmented with the noise RVs. This reduces the dimension of the sigma points as well as the total number of sigma points used. The covariances of the noise source are then incorporated into the state covariance using a simple additive procedure. This implementation is given in Table 7.3. The complexity of the algorithm is of order L^3, where L is the dimension of the state. This is the same complexity as the EKF. The most costly operation is in forming the sample prior covariance matrix $\mathbf{P}_k^-$. Depending on the form of $\mathbf{F}$, this may be simplified; for example, for univariate time series or with parameter estimation (see Section 7.4), the complexity reduces to order L^2.

Table 7.2 Unscented Kalman filter (UKF) equations

Initialize with

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \tag{7.35}$$

$$\mathbf{P}_0 = \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T], \tag{7.36}$$

$$\hat{\mathbf{x}}_0^a = \mathbb{E}[\mathbf{x}^a] = [\hat{\mathbf{x}}_0^T \quad \mathbf{0} \quad \mathbf{0}]^T, \tag{7.37}$$

$$\mathbf{P}_0^a = \mathbb{E}[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T] = \begin{bmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^{\mathrm{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}^{\mathrm{n}} \end{bmatrix}. \tag{7.38}$$

For $k \in \{1, \ldots, \infty\}$,
calculate the sigma points:

$$\mathcal{X}_{k-1}^a = [\hat{\mathbf{x}}_{k-1}^a \quad \hat{\mathbf{x}}_{k-1}^a + \gamma\sqrt{\mathbf{P}_{k-1}^a} \quad \hat{\mathbf{x}}_{k-1}^a - \gamma\sqrt{\mathbf{P}_{k-1}^a}]. \tag{7.39}$$

The time-update equations are

$$\mathcal{X}_{k|k-1}^x = \mathbf{F}(\mathcal{X}_{k-1}^x, \mathbf{u}_{k-1}, (\mathcal{X}_{k-1}^v), \tag{7.40}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}^x, \tag{7.41}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-)(\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-)^T, \tag{7.42}$$

$$\mathcal{Y}_{k|k-1} = \mathbf{H}(\mathcal{X}_{k|k-1}^x, \mathcal{X}_{k-1}^n), \tag{7.43}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \tag{7.44}$$

and the measurement-update equations are

$$\mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T, \tag{7.45}$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T, \tag{7.46}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k}^{-1}, \tag{7.47}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-), \tag{7.48}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} \mathcal{K}_k^T, \tag{7.49}$$

where

$$\mathbf{x}^a = [\mathbf{x}^T \quad \mathbf{v}^T \quad \mathbf{n}^T]^T, \qquad \mathcal{X}^a = [(\mathcal{X}^x)^T \quad (\mathcal{X}^v)^T \quad (\mathcal{X}^n)^T]^T, \qquad \gamma = \sqrt{L + \lambda},$$

λ is the composite scaling parameter, L is the dimension of the augmented state, $\mathbf{R}^{\mathrm{v}}$ is the process-noise covariance, $\mathbf{R}^{\mathrm{n}}$ is the measurement-noise covariance, and W_i are the weights as calculated in Eq. (7.34).

Table 7.3 UKF–additive (zero mean) noise case

Initialize with

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \tag{7.50}$$

$$\mathbf{P}_0 = \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]. \tag{7.51}$$

For $k \in \{1, \ldots, \infty\}$,
calculate the sigma points:

$$\mathcal{X}_{k-1} = [\hat{\mathbf{x}}_{k-1} \quad \hat{\mathbf{x}}_{k-1} + \gamma\sqrt{\mathbf{P}_{k-1}} \quad \hat{\mathbf{x}}_{k-1} - \gamma\sqrt{\mathbf{P}_{k-1}}]. \tag{7.52}$$

The time-update equations are

$$\mathcal{X}^*_{k|k-1} = \mathbf{F}(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}), \tag{7.53}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1} \tag{7.54}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-)(\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-)^T + \mathbf{R}^{\mathbf{v}}, \tag{7.55}$$

(augment sigma points)[6]

$$\mathcal{X}_{k|k-1} = [\mathcal{X}^*_{k|k-1} \quad \mathcal{X}^*_{0,k|k-1} + \gamma\sqrt{\mathbf{R}^{\mathbf{v}}} \quad \mathcal{X}^*_{0,k|k-1} - \gamma\sqrt{\mathbf{R}^{\mathbf{v}}}] \tag{7.56}$$

$$\mathcal{Y}_{k|k-1} = \mathbf{H}(\mathcal{X}_{k|k-1}), \tag{7.57}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \tag{7.58}$$

and the measurement-update equations are

$$\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T + \mathbf{R}^{\mathbf{n}}, \tag{7.59}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T \tag{7.60}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}\mathbf{P}^{-1}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} \tag{7.61}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \tag{7.62}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} \mathcal{K}_k^T, \tag{7.63}$$

where $\gamma = \sqrt{L + \lambda}$, λ is the composite scaling parameter, L is the dimension of the state, $\mathbf{R}^{\mathbf{v}}$ is the process-noise covariance, $\mathbf{R}^{\mathbf{n}}$ is the measurement-noise covariance and W_i are the weights as calculated in Eq. (7.34).

[6]Here we augment the sigma points with additional points derived from the matrix square root of the process noise covariance. This requires setting $L \rightarrow 2L$ and recalculating the various weights W_i accordingly. Alternatively, we may *redraw* a complete new set of sigma points, i.e., $\mathcal{X}_{k|k-1} = [\hat{\mathbf{x}}_k^- \quad \hat{\mathbf{x}}_k^- + \gamma\sqrt{\mathbf{P}_k^-} \quad \hat{\mathbf{x}}_k^- - \gamma\sqrt{\mathbf{P}_k^-}]$. This alternative approach results in fewer sigma points being used, but also discards any odd-moments information captured by the original propagated sigma points.

A number of variations for numerical purposes are also possible. For example, the matrix square root, which can be implemented directly using a Cholesky factorization, is in general of order $\frac{1}{6}L^3$. However, the covariance matrices are expressed recursively, and thus the square root can be computed in only order $M \times L^2$ (where M is the dimension of the output $\mathbf{y}_k$) by performing a recursive update to the Cholesky factorization. Details of an efficient recursive square-root UKF implementation are given in Appendix B.

7.3.1 State-Estimation Examples

The UKF was originally designed for state estimation applied to nonlinear control applications requiring full-state feedback [1–3]. We provide an example for a double inverted pendulum control system. In addition, we provide a new application example corresponding to noisy time-series estimation with neural networks.

Double Inverted Pendulum A double inverted pendulum (see Fig. 7.4) has states corresponding to cart position and velocity, and top and bottom pendulum angle and angular velocity, $\mathbf{x} = [x, \dot{x}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2]$. The system parameters correspond to the length and mass of each pendulum, and the cart mass, $\mathbf{w} = [l_1, l_2, m_1, m_2, M]$. The dynamical equations are

$$\begin{aligned}(M + m_1 + m_2)\ddot{x} - (m_1 + 2m_2)l_1\ddot{\theta}_1 \cos\theta_1 - m_2 l_2 \ddot{\theta}_2 \cos\theta_2 \\ = u + (m_1 + 2m_2)l_1(\dot{\theta}_1)^2 \sin\theta_1 + m_2 l_2 (\dot{\theta}_2)^2 \sin\theta_2, \qquad (7.64)\end{aligned}$$

$$\begin{aligned}-(m_1 + 2m_2)l_1\ddot{x}\cos\theta_1 + 4(\tfrac{1}{3}m_1 + m_2)(l_1)^2\ddot{\theta}_1 + 2m_2 l_1 l_2 \ddot{\theta}_2 \cos(\theta_2 - \theta_1) \\ = (m_1 + 2m_2)g l_1 \sin\theta_1 + 2m_2 l_1 l_2 (\dot{\theta}_2)^2 \sin(\theta_2 - \theta_1), \qquad (7.65)\end{aligned}$$

$$\begin{aligned}-m_2\ddot{x}l_2\cos\theta_2 + 2m_2 l_1 l_2 \ddot{\theta}_1 \cos(\theta_2 - \theta_1) + \tfrac{4}{3}m_2(l_2)^2\ddot{\theta}_2 \\ = m_2 g l_2 \sin\theta_2 - 2m_2 l_1 l_2 (\dot{\theta}_1)^2 \sin(\theta_2 - \theta_1). \qquad (7.66)\end{aligned}$$

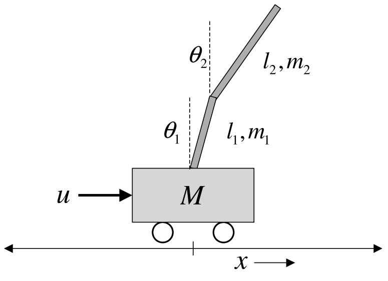


Figure 7.4 Double inverted pendulum.

These continuous-time dynamics are discretized with a sampling period of 0.02 seconds. The pendulum is stabilized by applying a control force u to the cart. In this case, we use a state-dependent Ricatti equation (SDRE) controller to stabilize the system.[7] A state estimator is run outside the control loop in order to compare the EKF with the UKF (i.e., the estimated states are not used in the feedback control for evaluation purposes). The observation corresponds to noisy measurements of the cart position, cart velocity, and angle of the top pendulum. This is a challenging problem, since no measurements are made for the bottom pendulum, nor for the angular velocity of the top pendulum. For this experiment, the pendulum is initialized in a jack-knife position ($+25^\circ/-25^\circ$), with a cart offset of 0.5 meters. The resulting state estimates are shown in Figure 7.5. Clearly, the UKF is better able to track the unobserved states.[8] If the estimated states are used for feedback in the control loop, the UKF system is still able to stabilize the pendulum, while the EKF system crashes. We shall return to the double inverted pendulum problem later in this chapter for both model estimation and dual estimation.

Noisy Time-Series Estimation In this example, the UKF is used to estimate an underlying clean time series corrupted by additive Gaussian white noise. The time-series used is the Mackey–Glass-30 chaotic series [15, 16]. The clean time-series is first modeled as a nonlinear autoregression

$$x_k = f(x_{k-1}, \ldots x_{k-M}, \mathbf{w}) + v_k, \tag{7.67}$$

where the model f (parameterised by $\mathbf{w}$) was approximated by training a feedforward neural network on the clean sequence. The residual error after convergence was taken to be the process-noise variance.

Next, white Gaussian noise was added to the clean Mackey–Glass series to generate a noisy time series $y_k = x_k + n_k$. The corresponding

[7]An SDRE controller [11] is designed by formulating the dynamical equations as $\mathbf{x}_{k+1} = \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)\mathbf{u}_k$. Note, this representation is *not* a linearization, but rather a reformulation of the nonlinear dynamics into a pseudo-linear form. Based on this state-space representation, we design an optimal LQR controller, $\mathbf{u}_k = -\mathbf{R}^{-1}\mathbf{B}^T(\mathbf{x}_k)\mathbf{P}(\mathbf{x}_k)\mathbf{x}_k \equiv \mathbf{K}(\mathbf{x}_k)\mathbf{x}_k$, where $\mathbf{P}(\mathbf{x}_k)$ is a solution of the standard Ricatti equations using state-dependent matrices $\mathbf{A}(\mathbf{x}_k)$ and $\mathbf{B}(\mathbf{x}_k)$. The procedure is repeated at every time step at the current state $\mathbf{x}_k$, and provides local asymptotic stability of the plant [14]. The approach has been found to be far more robust than LQR controllers based on standard linearization techniques.

[8]Note that if all six states are observed with noise, then the performances of the EKF and UKF are comparable.

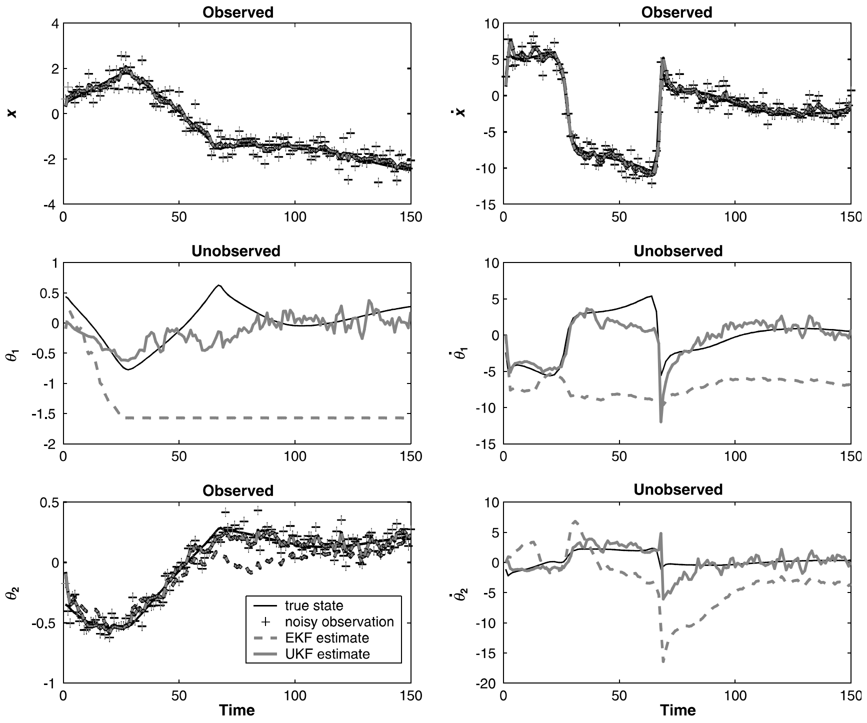


Figure 7.5 State estimation for the double inverted pendulum problem. Only three noisy states are observed: cart position, cart velocity, and the angle of the top pendulum. (10 dB SNR; $\alpha = 1$, $\beta = 0$, $\kappa = 0$.)

state-space representation is given by

$$\begin{aligned}
\mathbf{x}_{k+1} &= \qquad \mathbf{F}(\mathbf{x}_k, \mathbf{w}) \qquad + \mathbf{B}v_k, \\
\begin{bmatrix} x_{k+1} \\ x_k \\ \vdots \\ x_{k-M} \end{bmatrix} &= \begin{bmatrix} f(x_k, \ldots, x_{k-M+1}, \mathbf{w}) \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \vdots \\ x_{k-M+1} \end{bmatrix} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} v_k, \\
y_k &= [1 \quad 0 \quad \ldots \quad 0]\mathbf{x}_k + n_k.
\end{aligned} \tag{7.68}$$

In the estimation problem, the noisy time-series y_k is the only observed input to either the EKF or UKF algorithms (both utilize the known neural network model). Figure 7.6 shows a subsegment of the estimates generated by both the EKF and the UKF (the original noisy time series has a 3 dB SNR). The superior performance of the UKF is clearly visible.

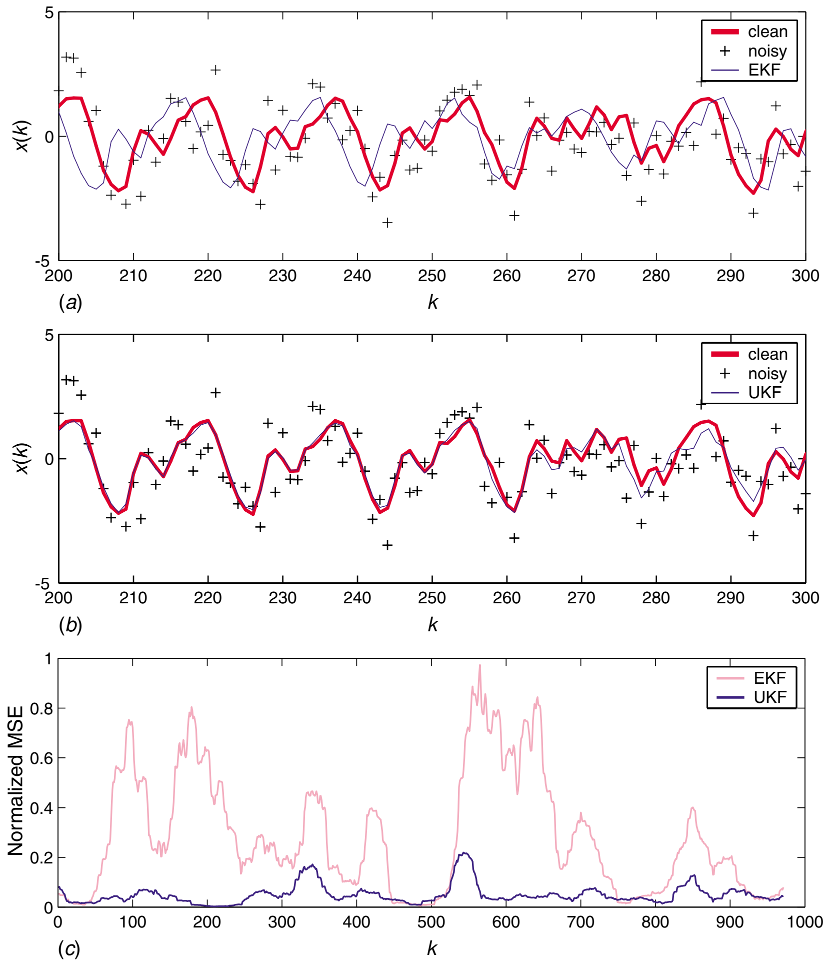


Figure 7.6 Estimation of Mackey–Glass time series using a known model: (*a*) with the EKF; (*b*) with the UKF. (*c*) shows a comparison of estimation errors for the complete sequence.

7.3.2 The Unscented Kalman Smoother

As has been discussed, the Kalman filter is a recursive algorithm providing the conditional expectation of the state $\mathbf{x}_k$ given all observations $\mathbf{Y}_0^k$ up to the current time k. In contrast, the Kalman smoother estimates the state given all observations past and future, $\mathbf{Y}_0^N$, where N is the final time. Kalman smoothers are commonly used for applications such as trajectory planning, noncausal noise reduction, and the *E-step* in the EM algorithm

[17, 18]. A thorough treatment of the Kalman smoother in the linear case is given in [19]. The basic idea is to run a Kalman filter forward in time to estimate the mean and covariance ($\hat{\mathbf{x}}_k^f$, $\mathbf{P}_k^f$) of the state, given *past* data. A second Kalman filter is then run backward in time to produce a backward-time predicted mean and covariance ($\hat{\mathbf{x}}_k^{-b}$, $\mathbf{P}_k^{-b}$), given the *future* data. These two estimates are then combined, producing the following smoothed statistics, given *all* the data:

$$(\mathbf{P}_k^s)^{-1} = (\mathbf{P}_k^f)^{-1} + (\mathbf{P}_k^{-b})^{-1}, \tag{7.69}$$

$$\hat{\mathbf{x}}_k^s = \mathbf{P}_k^s[(\mathbf{P}_k^{-b})^{-1}\hat{\mathbf{x}}_k^{-b} + (\mathbf{P}_k^f)^{-1}\hat{\mathbf{x}}_k^f]. \tag{7.70}$$

For the nonlinear case, the EKF replaces the Kalman filter. The use of the EKF for the forward filter is straightforward. However, implementation of the backward filter is achieved by using the following linearized backward-time system:

$$\mathbf{x}_{k-1} = \mathbf{A}^{-1}\mathbf{x}_k + \mathbf{A}^{-1}\mathbf{B}\mathbf{v}_k \tag{7.71}$$

that is, the forward nonlinear dynamics are linearized, and then inverted for the backward model. A linear Kalman filter is then applied.

Our proposed unscented Kalman smoother (UKS) replaces the EKF with the UKF. In addition, we consider using a nonlinear backward model as well, either derived from first principles or by training a backward predictor using a neural network model, as illustrated for the time-series case in Figure 7.7. The nonlinear backward model allows us to take full advantage of the UKF, which requires no linearization step.

To illustrate performance, we reconsider the noisy Mackey–Glass time-series problem of the previous section, as well as a second time series generated using a chaotic autoregressive neural network. Table 7.4 compares *smoother* performance. In this case, the network models are trained on the clean time series, and then tested on the noisy data using the standard extended Kalman smoother with linearized backward model

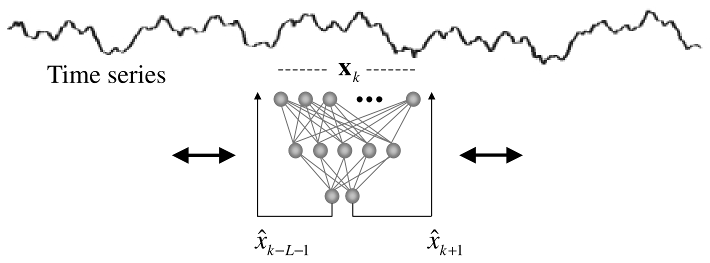


Figure 7.7 Forward/backward neural network prediction training.

Table 7.4 Comparison of smoother performance

Mackey–Glass

	Normalized MSE		
Algorithm	F	B	S
EKS1	0.20	0.70	0.27
EKS2	0.20	0.31	0.19
UKS	0.10	0.24	0.08

Chaotic AR–NN

	Normalized MSE		
Algorithm	F	B	S
EKS1	0.35	0.32	0.28
EKS2	0.35	0.22	0.23
UKS	0.23	0.21	0.16

(EKS1), an extended Kalman smoother with a second nonlinear backward model (EKS2), and the unscented Kalman smoother (UKS). The forward (F), backward (B), and smoothed (S) estimation errors are reported. Again, the performance benefits of the unscented approach are clear.

7.4 UKF PARAMETER ESTIMATION

Recall that parameter estimation involves learning a nonlinear mapping $\mathbf{y}_k = \mathbf{G}(\mathbf{x}_k, \mathbf{w})$, where $\mathbf{w}$ corresponds to the set of unknown parameters. $\mathbf{G}(\cdot)$ may be a neural network or another parameterized function. The EKF may be used to estimate the parameters by writing a new state-space representation

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{r}_k, \tag{7.73}$$

$$\mathbf{d}_k = \mathbf{G}(\mathbf{x}_k, \mathbf{w}_k) + \mathbf{e}_k, \tag{7.74}$$

where $\mathbf{w}_k$ corresponds to a stationary process with identity state transition matrix, driven by process noise $\mathbf{r}_k$. The desired output $\mathbf{d}_k$ corresponds to a nonlinear observation on $\mathbf{w}_k$. In the linear case, the relationship between the Kalman Filter (KF) and the popular recursive least-squares (RLS) is given in [20] and [25]. In the nonlinear case, the EKF training corresponds to a modified-Newton method [22] (see also Chapter 2).

From an optimization perspective, the following *prediction error* cost is minimized:

$$J(\mathbf{w}) = \sum_{t=1}^{k} [\mathbf{d}_t - \mathbf{G}(\mathbf{x}_t, \mathbf{w})]^{\mathrm{T}} (\mathbf{R}^{\mathbf{e}})^{-1} [\mathbf{d}_t - \mathbf{G}(\mathbf{x}_t, \mathbf{w})]. \tag{7.75}$$

Thus, if the "noise" covariance $\mathbf{R}^{\mathbf{e}}$ is a constant diagonal matrix, then, in fact, it cancels out of the algorithm (this can be shown explicitly), and hence can be set arbitrarily (e.g., $\mathbf{R}^{\mathbf{e}} = 0.5\mathbf{I}$). Alternatively, $\mathbf{R}^{\mathbf{e}}$ can be set to specify a weighted MSE cost. The innovations covariance $\mathbb{E}[\mathbf{r}_k \mathbf{r}_k^T] = \mathbf{R}_k^{\mathbf{r}}$, on the other hand, affects the convergence rate and tracking performance. Roughly speaking, the larger the covariance, the more quickly older data is discarded. There are several options on how to choose $\mathbf{R}_k^{\mathbf{r}}$.

- Set $\mathbf{R}_k^{\mathbf{r}}$ to an arbitrary "fixed" diagonal value, which may then be "annealed" towards zero as training continues.
- Set $\mathbf{R}_k^{\mathbf{r}} = (\lambda_{RLS}^{-1} - 1)\mathbf{P}_{\mathbf{w}_k}$, where $\lambda_{RLS} \in (0, 1]$ is often referred to as the "forgetting factor," as defined in the recursive least-squares (RLS) algorithm [21]. This provides for an approximate exponentially decaying weighting on past data, and is described more fully in [22]. Note that λ_{RLS} should not be confused with λ used for sigma-point calculation.
- Set

$$\begin{aligned}\mathbf{R}_k^{\mathbf{r}} = (1 - \alpha_{\mathrm{RM}})\mathbf{R}_{k-1}^{\mathbf{r}} + \alpha_{\mathrm{RM}} \mathbf{K}_k^{\mathbf{w}} [\mathbf{d}_k - \mathbf{G}(\mathbf{x}_k, \hat{\mathbf{w}})] \\ \times [\mathbf{d}_k - \mathbf{G}(\mathbf{x}_k, \hat{\mathbf{w}})]^T (\mathbf{K}_k^{\mathbf{w}})^T,\end{aligned}$$

  which is a Robbins–Monro stochastic approximation scheme for estimating the innovations [23]. The method assumes that the covariance of the Kalman update model is consistent with the actual update model. Typically, $\mathbf{R}_k^{\mathbf{r}}$ is also constrained to be a diagonal matrix, which implies an independence assumption on the parameters. Note that a similar update may also be used for $\mathbf{R}_k^{\mathbf{e}}$.

Our experience indicates that the "Robbins–Monro" method provides the fastest rate of absolute convergence and lowest final MMSE values (see the experiments in the next section). The "fixed" $\mathbf{R}_k^{\mathbf{r}}$ in combination with annealing can also achieve good final MMSE performance, but requires more monitoring and a greater prior knowledge of the noise levels. For problems where the MMSE is zero, the covariance should be lower-bounded to prevent the algorithm from stalling and potential numerical problems. The "forgetting-factor" and "fixed" $\mathbf{R}_k^{\mathbf{r}}$ methods are most appropriate for on-line learning problems in which tracking of time-varying parameters is necessary. In this case, the parameter covariance stays lower-bounded, allowing the most recent data to be emphasized. This leads to some misadjustment, but also keeps the Kalman gain sufficiently large to maintain good tracking. In general, study of the various trade-offs between these different approaches is still an area of open research.

The UKF represents an alternative to the EKF for parameter estimation. However, as the state transition function is linear, the advantage of the UKF may not be as obvious. Note that the observation function is still nonlinear. Furthermore, the EKF essentially builds up an approximation to the expected Hessian by taking outer products of the gradient. The UKF, however, may provide a more accurate estimate through direct approximation of the expectation of the Hessian. While both the EKF and UKF can be expected to achieve similar final MMSE performance, their covergence properties may differ. In addition, a distinct advantage of the UKF occurs when either the architecture or error metric is such that differentiation with respect to the parameters is not easily derived, as is necessary in the EKF. The UKF effectively evaluates both the Jacobian and Hessian precisely through its sigma-point propagation, without the need to perform any analytical differentiation.

Specific equations for UKF parameter estimation are given in Table 7.5. Simplifications have been made relative to the state UKF, accounting for the specific form of the state transition function. In Table 7.5, we have provided two options on how the function output $\hat{\mathbf{d}}_k$ is achieved. In the first option, the output is given as

$$\hat{\mathbf{d}}_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{D}_{i,k|k-1} \approx \mathbb{E}[\mathbf{G}(\mathbf{x}_k, \mathbf{w}_k)], \tag{7.89}$$

corresponding to the direct interpretation of the UKF equations. The output is the expected value (mean) of a function of the random variable $\mathbf{w}_k$. In the second option, we have

$$\hat{\mathbf{d}}_k = \mathbf{G}(\mathbf{x}_k, \hat{\mathbf{w}}_k^-), \tag{7.90}$$

corresponding to the typical interpretation, in which the output is the function with the current "best" set of parameters. This option yields convergence performance that is indistinguishable from the EKF. The first option, however, has different convergence characteristics, and requires

Table 7.5 UKF parameter estimation

Initialize with

$$\hat{\mathbf{w}}_0 = E[\mathbf{w}], \tag{7.76}$$

$$\mathbf{P}_{\mathbf{w}_0} = E[(\mathbf{w} - \hat{\mathbf{w}}_0)(\mathbf{w} - \hat{\mathbf{w}}_0)^T]. \tag{7.77}$$

For $k \in \{1, \ldots, \infty\}$,
The time update and sigma-point calculation are given by

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1}, \tag{7.78}$$

$$\mathbf{P}_{\mathbf{w}_k}^- = \mathbf{P}_{\mathbf{w}_{k-1}} + \mathbf{R}_{k-1}^{\mathbf{r}}, \tag{7.79}$$

$$\mathcal{W}_{k|k-1} = [\hat{\mathbf{w}}_k^- \quad \hat{\mathbf{w}}_k^- + \gamma\sqrt{\mathbf{P}_{\mathbf{w}_k}^-} \quad \hat{\mathbf{w}}_k^- - \gamma\sqrt{\mathbf{P}_{\mathbf{w}_k}^-}], \tag{7.80}$$

$$\mathcal{D}_{k|k-1} = \mathbf{G}(\mathbf{x}_k, \mathcal{W}_{k|k-1}), \tag{7.81}$$

$$\text{option 1:} \quad \hat{\mathbf{d}}_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{D}_{i,k|k-1}, \tag{7.82}$$

$$\text{option 2:} \quad \hat{\mathbf{d}}_k = \mathbf{G}(\mathbf{x}_k, \hat{\mathbf{w}}_k^-). \tag{7.83}$$

and the measurement-update equations are

$$\mathbf{P}_{\tilde{\mathbf{d}}_k \tilde{\mathbf{d}}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k)(\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k)^T + \mathbf{R}_k^{\mathbf{e}}, \tag{7.84}$$

$$\mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{W}_{i,k|k-1} - \hat{\mathbf{w}}_k^-)(\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k)^T, \tag{7.85}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} \mathbf{P}_{\tilde{\mathbf{d}}_k \tilde{\mathbf{d}}_k}^{-1}, \tag{7.86}$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathcal{K}_k(\mathbf{d}_k - \hat{\mathbf{d}}_k), \tag{7.87}$$

$$\mathbf{P}_{\mathbf{w}_k} = \mathbf{P}_{\mathbf{w}_k}^- - \mathcal{K}_k \mathbf{P}_{\tilde{\mathbf{d}}_k \tilde{\mathbf{d}}_k} \mathcal{K}_k^T, \tag{7.88}$$

where $\gamma = \sqrt{L + \lambda}$, λ is the composite scaling parameter, L is the dimension of the state, $\mathbf{R}^{\mathbf{r}}$ is the process-noise covariance, $\mathbf{R}^{\mathbf{e}}$ is the measurement-noise covariance, and W_i are the weights as calculated in Eq. (7.34).

further explanation. In the state-space approach to parameter estimation, absolute convergence is achieved when the parameter covariance $\mathbf{P}_{\mathbf{w}_k}$ goes to zero (this also forces the Kalman gain to zero). At this point, the output for either option is identical. However, prior to this, the finite covariance provides a form of averaging on the output of the function, which in turn prevents the parameters from going to the minimum of the error surface. Thus, the method may help avoid falling into a local minimum. Furthermore, it provides a form of built-in regularization for short or noisy data sets that are prone to overfitting (exact specification of the level of regularization requires further study).

Note that the complexity of the UKF algorithm is still of order L^3 (L is the number of parameters), owing to the need to compute a matrix square root at each time step. An order L^2 complexity (same as the EKF) can be achieved by using a recursive square-root formulation as given in Appendix B.

7.4.1 Parameter Estimation Examples

We have performed a number of experiments to illustrate the performance of the UKF parameter-estimation approach. The first set of experiments corresponds to benchmark problems for neural network training, and serve to illustrate some of the differences between the EKF and UKF, as well as the different options discussed above. Two parametric optimization problems are also included, corresponding to model estimation of the double pendulum, and the benchmark "Rosenbrock's Banana" optimization problem.

Benchmark NN Regression and Time-Series Problems The *Mackay robot-arm* dataset [24, 25] and the *Ikeda* chaotic time series [26] are used as benchmark problems to compare neural network training. Figure 7.8 illustrates the differences in learning curves for the EKF versus UKF (option 1). Note the slightly lower final MSE performance of the UKF weight training. If option 2 for the UKF output is used (see Eq. (7.82), then the learning curves for the EKF and UKF are indistinguishable; this has been found to be consistent with all experiments; therefore, we shall not show explicit learning curves for the UKF with option 2.

Figure 7.9 illustrates performance differences based on the choice of processing noise covariance $\mathbf{R}_k^{\mathbf{r}}$. The Mackey–Glass and Ikeda time series are used. The plots show only comparisons for the UKF (differences are similar for the EKF). In general, the Robbins–Monro method is the most robust approach, with the fastest rate of convergence. In some examples, we have seen faster convergence with the "annealed" approach; however, this also requires additional insight and heuristic methods to monitor the learning. We should reiterate that the "fixed" and "lambda" approaches are more appropriate for on-line tracking problems.

Four-Regions Classification In the next example, we consider a benchmark pattern classification problem having four interlocking regions

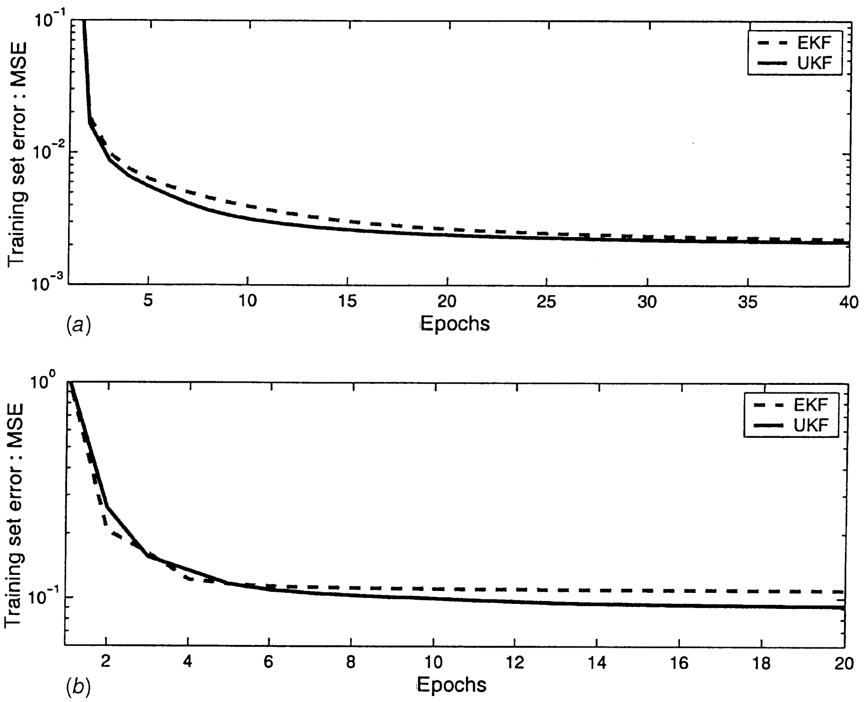


Figure 7.8 (*a*) MacKay robot-arm problem: comparison of learning curves for the EKF and UKF training, 2-12-2 MLP, annealing noise estimation. (*b*) Ikeda chaotic time series: comparison of learning curves for the EKF and UKF training, 10-7-1 MLP, Robbins–Monro noise estimation.

[8]. A three-layer feedforward network (MLP) with 2-10-10-4 nodes is trained using inputs randomly drawn within the pattern space, $S = [-1, -1] \times [1, 1]$, with the desired output value of $+0.8$ if the pattern fell within the assigned region and -0.8 otherwise. Figure 7.10 illustrates the classification task, learning curves for the UKF and EKF, and the final classification regions. For the learning curve, each epoch represents 100 randomly drawn input samples. The test set evaluated on each epoch corresponds to a uniform grid of 10,000 points. Again, we see the superior performance of the UKF.

Double Inverted Pendulum Returning to the double inverted pendulum (Section 7.3.1), we consider learning the system parameters, $\mathbf{w} = [l_1, l_2, m_1, m_2, M]$. These parameter values are treated as unknown (all initialized to 1.0). The full state, $\mathbf{x} = [x, \dot{x}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2]$, is observed.

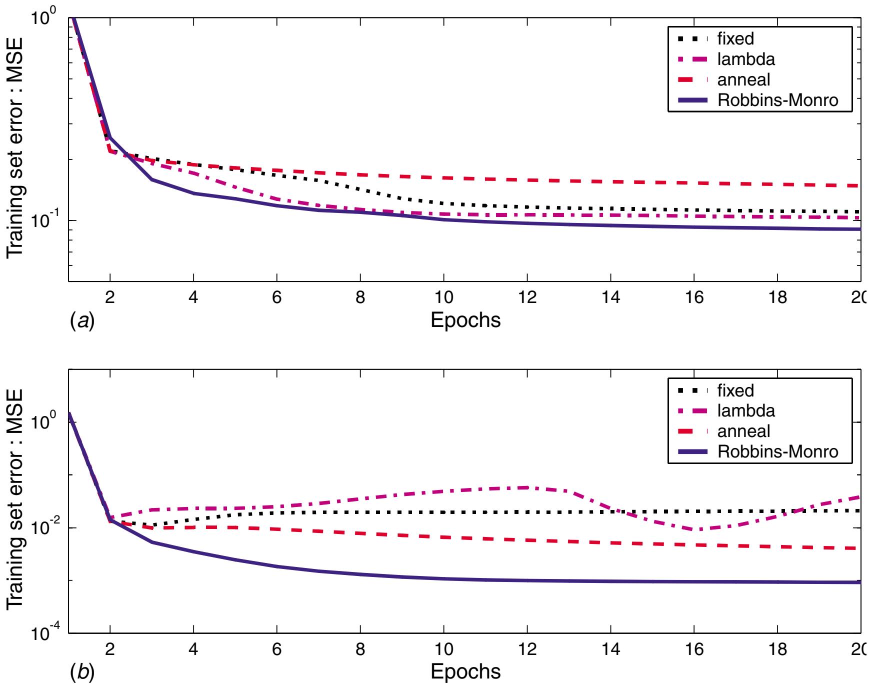


Figure 7.9 Neural network parameter estimation using different methods for noise estimation. (*a*) Ikeda chaotic time series. (*b*) Mackey–Glass chaotic time series. (UKF settings: $\alpha = 10^{-4}$, $\beta = 2$, $\kappa = 3 - L$, where L is the state dimension.)

Figure 7.11 shows the total model MSE versus iteration comparing EKF with UKF. Each iteration represents a pendulum crash with different initial conditions for the state (no control is applied). The final converged parameter estimates are as follows:

	l_1	l_2	m_1	m_2	M
True model	0.50	0.75	0.75	0.50	1.50
UKF estimate	0.50	0.75	0.75	0.50	1.49
EKF estimate	0.50	0.75	0.68	0.45	1.35

In this case, the EKF has converged to a biased solution, possibly corresponding to a local minimum in the error surface.

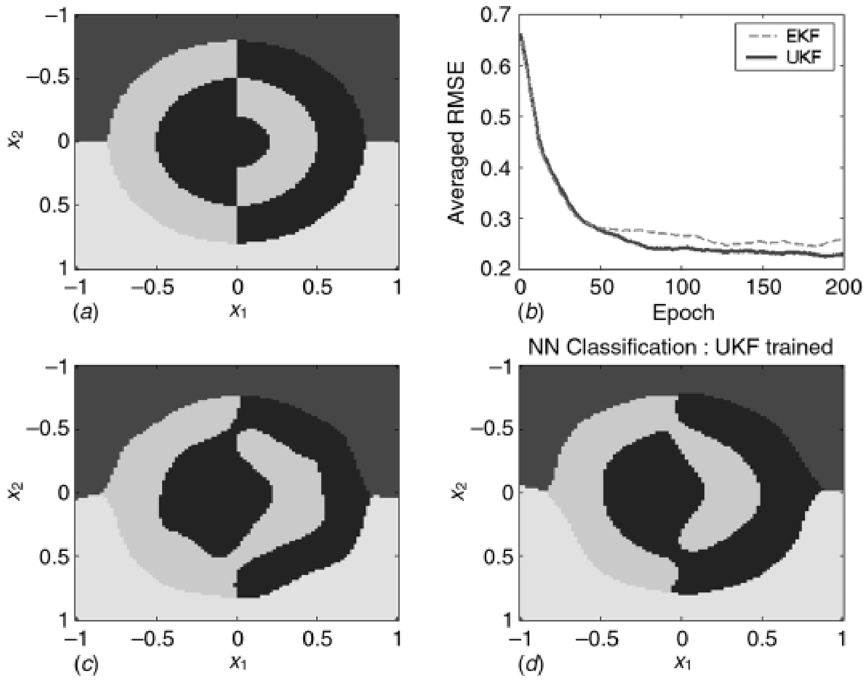


Figure 7.10 Singhal and Wu's four-region classification problem. (*a*) True mapping. (*b*) Learning curves on the test set. (*c*) NN classification: EKF-trained. (*d*) NN classification: UKF-trained. (UKF settings: $\alpha = 10^{-4}$, $\beta = 2$, $\kappa = 3 - L$, where L is the state dimension; 2-10-10-4 MLP; Robbins–Monro; 1 epoch = 100 random examples.)

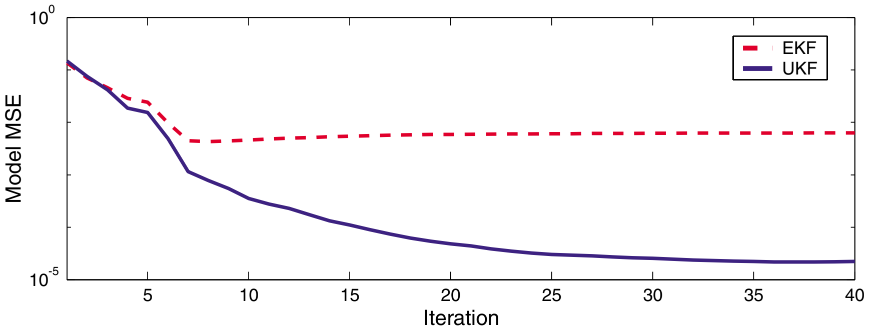


Figure 7.11 Inverted double pendulum parameter estimation. (UKF settings: $\alpha = 10^{-4}$, $\beta = 2$, $\kappa = 3 - L$, where L is the state dimension; Robbins–Monro.)

Rosenbrock's Banana Function For the last parameter estimation example, we turn to a pure optimization problem. The Banana function [27] can be thought of as a two-dimensional surface with a saddle-like curvature that bends around the origin. Specifically, we wish to find the values of x_1 and x_2 that minimize the function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{7.91}$$

The true minimum is at $x_1 = 1$ and $x_2 = 1$. The Banana function is a well-known test problem used to compare the convergence rates of competing minimization techniques.

In order to use the UKF or EKF, the basic parameter estimation equations need to be reformulated to minimize a non-MSE cost function. To do this we write the state-space equations in observed error form [28]:

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mathbf{r}_k, \tag{7.92}$$

$$0 = -\mathfrak{e}_k + \mathbf{e}_k, \tag{7.93}$$

where the target "observation" is fixed at zero, and $\mathfrak{e}_k$ is an error term resulting in the optimization of the sum of instantaneous costs $J_k = \mathfrak{e}_k^T \mathfrak{e}_k$. The MSE cost is optimized by setting $\mathfrak{e}_k = \mathbf{d}_k - \mathbf{G}(\mathbf{x}_k, \mathbf{w}_k)$. However, arbitrary costs (e.g., cross-entropy) can also be minimized simply by specifying $\mathfrak{e}_k$ appropriately. Further discussion of this approach has been given in Chapter 5. Reformulation of the UKF equations requires changing only the effective output to $\mathfrak{e}_k$, and setting the desired response to zero.

For the example at hand, we set $\mathfrak{e}_k = [10(x_2 - x_1) \quad 1 - x_1]^T$. Furthermore, since this optimization problem is a special case of "noiseless" parameter estimation where the actual error can be minimized to zero, we make use of Eq. (7.89) (option 2) to calculate the output of the UKF algorithm. This will allow the UKF to reach the true minimum of the error surface more rapidly.[9] We also set the scaling parameter α to a small value, which we have found to be appropriate again for zero MSE problems. Under these circumstances, the performances of the UKF and EKF are indistinguishable, as illustrated in Figure 7.12. Overall, the performances

[9]Note that the use of option 1, where the expected value of the function is used as the output, essentially involves averaging of the output based on the current parameter covariance. This shows convergence in the case where zero MSE is possible, since convergence of the state covariance to zero would also be necessary through proper annealing of the state noise innovations $\mathbf{R}^{\mathbf{r}}$.

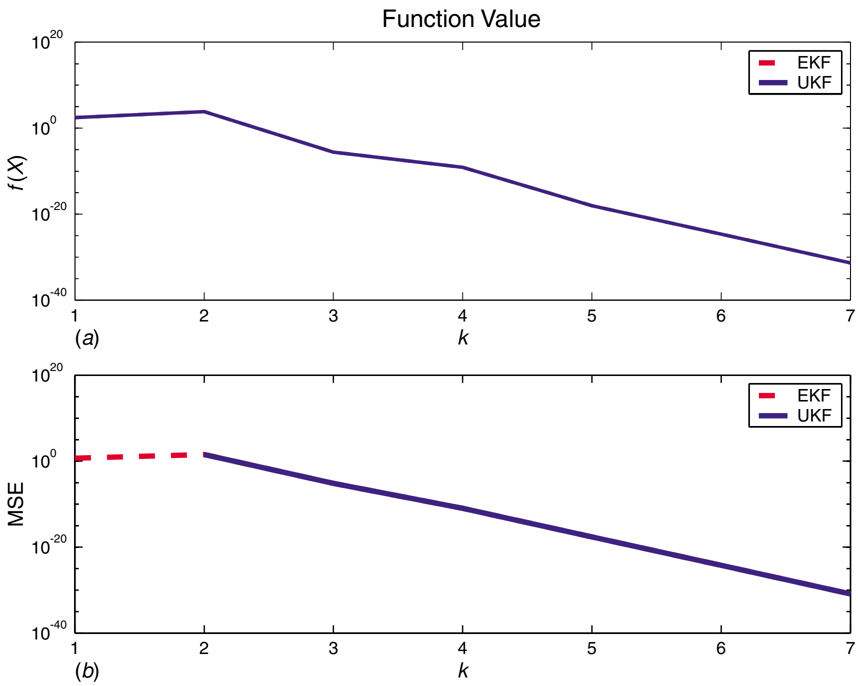


Figure 7.12 Rosenbrock's "Banana" optimization problem. (*a*) Function value. (*b*) Model error. (UKF settings: $\alpha = 10^{-4}$, $\beta = 2$, $\kappa = 3 - L$, where L is the state dimension; Fixed.)

of the two filters are comparable or superior to those of a number of alternative optimization approaches (e.g., Davidson–Fletcher–Powell, Levenburg–Marquardt, etc. See "optdemo" in *Matlab*). The main purpose of this example was to illustrate the versatility of the UKF to general optimization problems.

7.5 UKF DUAL ESTIMATION

Recall that the dual estimation problem consists of simultaneously estimating the clean state $\mathbf{x}_k$ and the model parameters $\mathbf{w}$ from the noisy data y_k (see Eq. (7.7)). A number of algorithmic approaches exist for this problem, including joint and dual EKF methods (recursive prediction error and maximum-likelihood versions), and expectation–maximization (EM) approaches. A thorough coverage of these algorithms is given in Chapters 5 and 6. In this section, we present results for the dual UKF (prediction error) and joint UKF methods.

In the *dual extended Kalman filter* [29], a separate state-space representation is used for the signal and the weights. Two EKFs are run simultaneously for signal and weight estimation. At every time step, the current estimate of the weights is used in the signal filter, and the current estimate of the signal state is used in the weight filter. In the *dual UKF* algorithm, both state and weight estimation are done with the UKF.

In the *joint extended Kalman filter* [30], the signal-state and weight vectors are concatenated into a single, *joint* state vector: $[\mathbf{x}_k^T \ \mathbf{w}_k^T]^T$. Estimation is done recursively by writing the state-space equations for the joint state as

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{w}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{I}\mathbf{w}_k \end{bmatrix} + \begin{bmatrix} \mathbf{B}v_k \\ \mathbf{r}_k \end{bmatrix}, \tag{7.94}$$

$$y_k = [1 \quad 0 \quad \ldots \quad 0] \begin{bmatrix} \mathbf{x}_k \\ \mathbf{w}_k \end{bmatrix} + n_k, \tag{7.95}$$

and running an EKF on the joint state space to produce simultaneous estimates of the states $\mathbf{x}_k$ and $\mathbf{w}$. Again, our approach is to use the UKF instead of the EKF.

7.5.1 Dual Estimation Experiments

Noisy Time-Series We present results on two time-series to provide a clear illustration of the use of the UKF over the EKF. The first series is again the Mackey–Glass-30 chaotic series with additive noise (SNR $\approx$ 3 dB). The second time series (also chaotic) comes from an autoregressive neural network with random weights driven by Gaussian process noise and also corrupted by additive white Gaussian noise (SNR $\approx$ 3 dB). A standard 6-10-1 MLP with tanh hidden activation functions and a linear output layer was used for all the filters in the Mackey–Glass problem. A 5-3-1 MLP was used for the second problem. The process- and measurement-noise variances associated with the state were assumed to be known. Note that, in contrast to the state estimation example in the previous section, only the noisy time series is observed. A clean reference is never provided for training.

Example training curves for the different dual and joint Kalman-based estimation methods are shown in Figure 7.13. A final estimate for the Mackey–Glass series is also shown for the dual UKF. The superior performance of the UKF-based algorithms is clear.

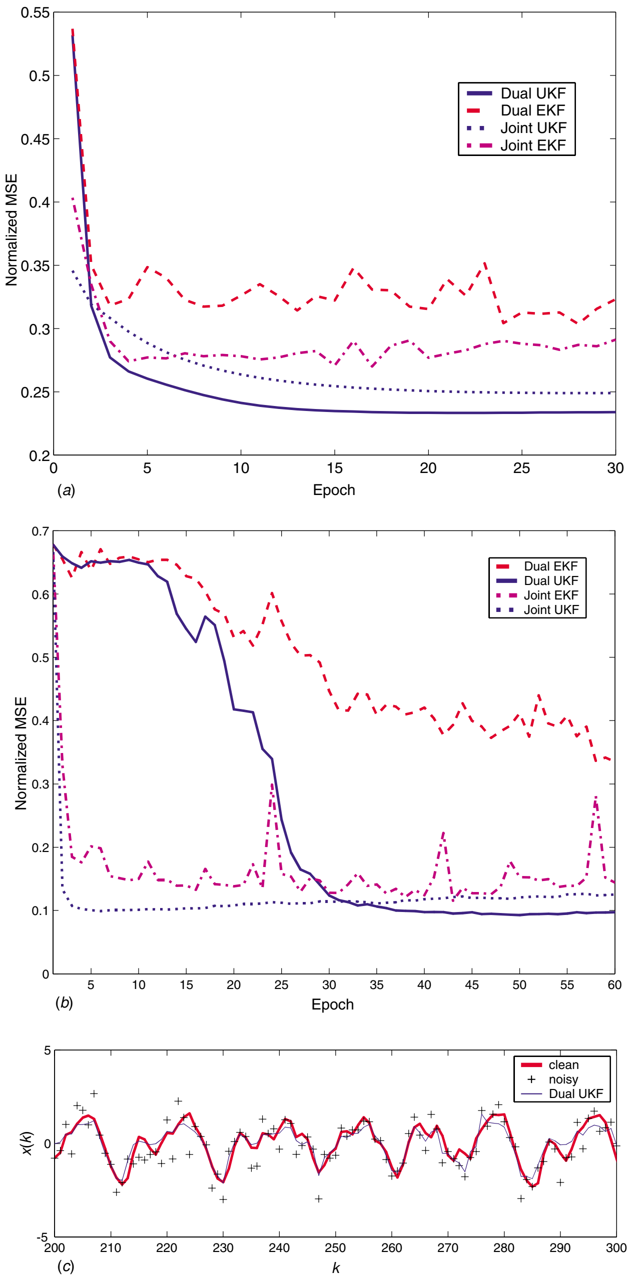
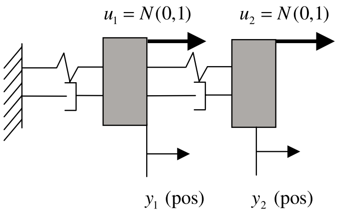


Figure 7.13 Comparative learning curves and results for the dual estimation experiments. Curves are averaged over 10 and 3 runs, respectively, using different initial weights. "Fixed" innovation covariances are used in the joint algorithms. "Annealed" covariances are used for the weight filter in the dual algorithms. (*a*) Chaotic AR neural network. (*b*) Mackey–Glass chaotic time series. (*c*) Estimation of Mackey–Glass time series: dual UKF.



$$\begin{bmatrix} x_1 \\ x_1 \\ x_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_1^2 & -2\zeta_1\omega_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\omega_2^2 & -2\zeta_2\omega_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1 \\ x_2 \\ x_2 \end{bmatrix}$$

Figure 7.14 Mass-and-spring system.

Mode Estimation This example illustrates the use of the joint UKF for estimating the modes of a mass-and-spring system (see Fig. 7.14). This work was performed at the University of Washington by Mark Campbell and Shelby Brunke. While the system is linear, direct estimation of the natural frequencies ω_1 and ω_2 jointly with the states is a nonlinear estimation problem. Figure 7.15 compares the performance of the EKF and UKF. Note that the EKF does not converge to the true value for ω_2. For this experiment, the input process noise SNR is approximately 100 dB, and the measured positions y_1 and y_2 have additive noise at a 60 dB SNR (these settings effectively turn the task into a pure parameter-estimation

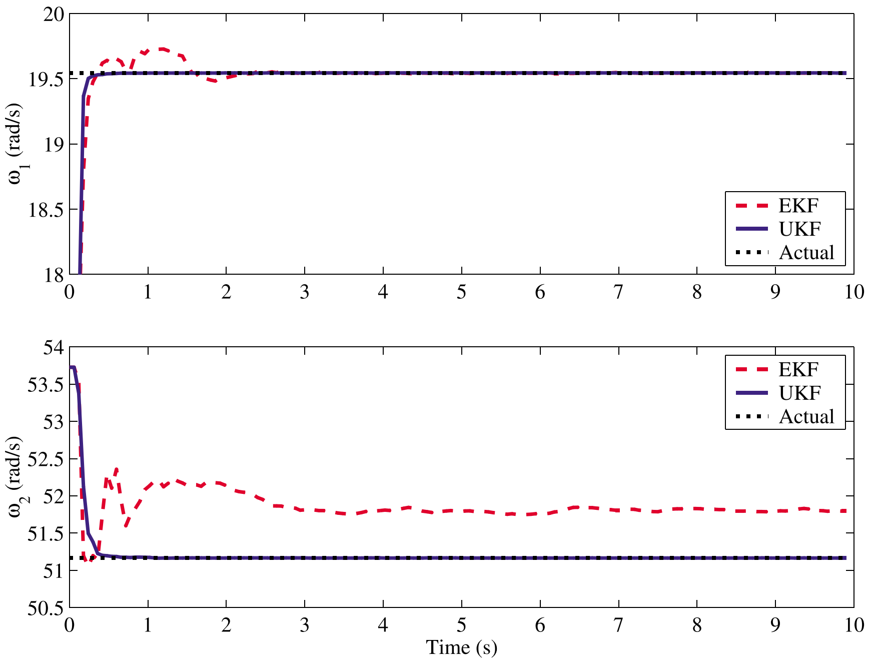


Figure 7.15 Linear mode prediction.

problem). A fixed innovations $\mathbf{R}^{\mathbf{r}}$ was used for the parameter estimation in the joint algorithms. Sampling was done at the Nyquist rate (based on ω_2), which emphasizes the effect of linearization in the EKF. For faster sampling rates, the performance of the EKF and UKF become more similar.

F15 Flight Simulation In this example (also performed at the University of Washington), joint estimation is done on an F15 aircraft model [31]. The simulation includes vehicle nonlinear dynamics, and engine and sensor noise modeling, as well as atmospheric modeling (densities, pressure, etc.) based on look-up tables. Also incorporated are aerodynamic forces based on data from Wright Patterson AFB. A closed-loop system using a gain-scheduled TECS controller is used to control the model [32]. A simulated mission was used to test the UKF estimator, and involved a quick descent, short tactical run, 180° turn, and ascent, with a possible failure in the stabilitator (horizontal control surface on the tail of the aircraft). Measurements consisted of the states with additive noise (20 dB SNR). Turbulence was approximately 1 m/s RMS. During the mission, the joint UKF estimated the 12 states (positions, orientations, and their derivatives) as well as parameters corresponding to aerodynamic forces and moments. This was done "off-line"; that is, the estimated states were not used within the control loop. Illustrative results are shown in Figure 7.16 for estimation of the altitude, velocity, and lift parameter (overall lift force on the aircraft). The left column shows the mission without a failure. The right column includes a 50% stabilitator failure at 65 seconds. Note that even with this failure, the UKF is still capable of tracking the state and parameters. It should be pointed out that the "black-box" nature of the simulator was not conducive to taking Jacobians necessary for running the EKF. Hence, implementation of the EKF for comparison was not performed.

Double Inverted Pendulum For the final dual estimation example, we again consider the double inverted pendulum, but this time we estimate both the states and system parameters using the joint UKF. Observations correspond to noisy measurements of the six states. Estimated states are then fed back for closed-loop control. In addition, parameter estimates are used at every time step to design the controller using the SDRE approach. Figure 7.17 illustrates performance of this adaptive control system by showing the evolution of the estimated and actual states. At the start of the simulation, both the states and parameters are unknown (the control system is unstable at this point). However, within one trial, the UKF

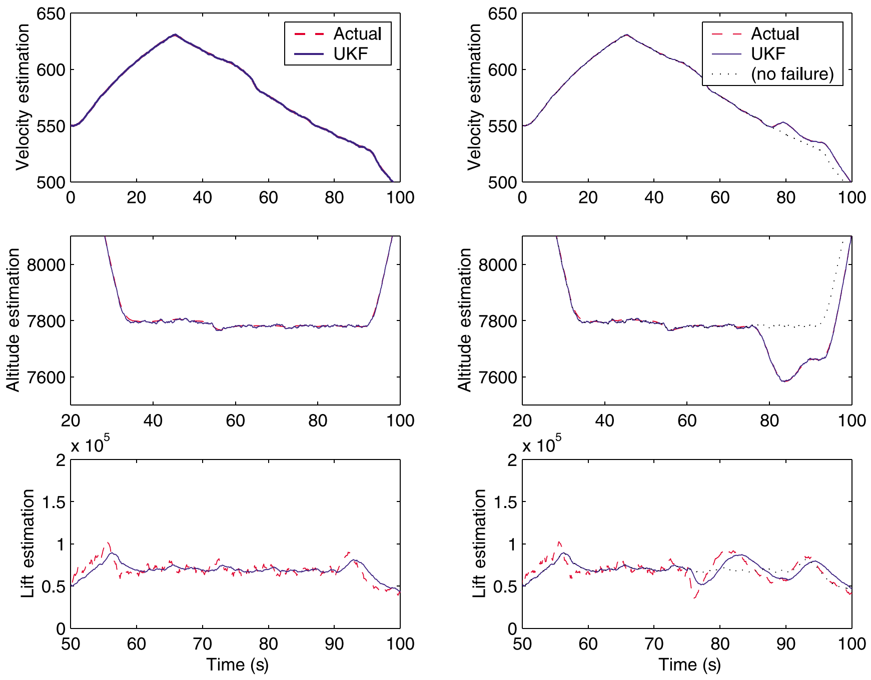


Figure 7.16 F15 model joint estimation (note that the estimated and true values of the state are indistinguishable at this resolution).

enables convergence and stabilization of the pendulum without a single crash.

7.6 THE UNSCENTED PARTICLE FILTER

The *particle filter* is a sequential Monte Carlo method that allows for a complete representation of the state distribution using sequential importance sampling and resampling [33–35]. Whereas the standard EKF and UKF make a Gaussian assumption to simplify the optimal recursive Bayesian estimation (see Section 7.2), particle filters make no assumptions on the form of the probability densities in question; that is, they employ full nonlinear, non-Gaussian estimation. In this section, we present a method that utilizes the UKF to augment and improve the standard particle filter, specifically through generation of the *importance proposal distribution*. This chapter will review the background fundamentals necessary to introduce particle filtering, and the extension based on the UKF. The

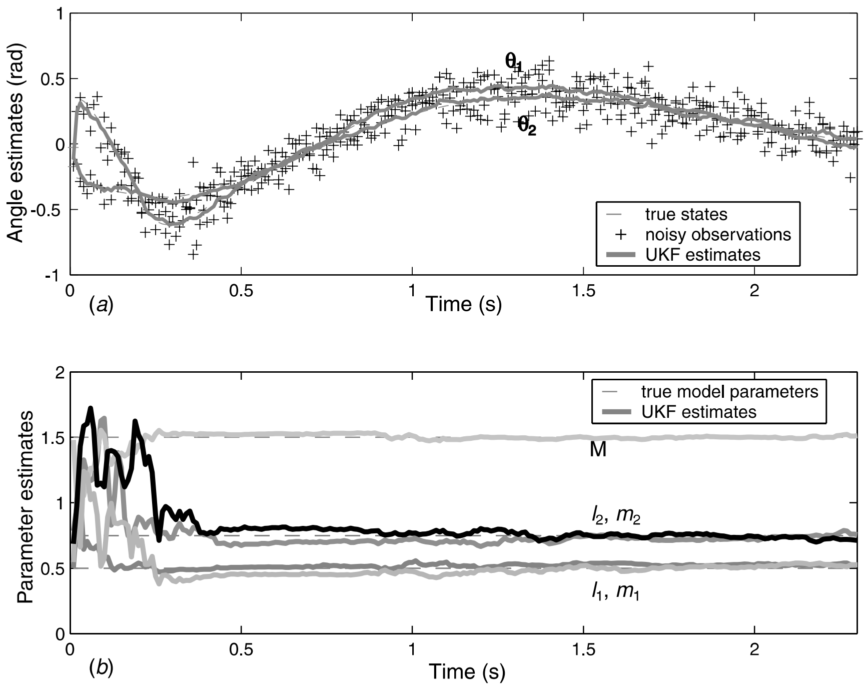


Figure 7.17 Double Inverted Pendulum joint estimation. Estimated states (*a*) and parameters (*b*). Only θ_1 and θ_2 are plotted (in radians).

material is based on work done by van der Merwe, de Freitas, Doucet, and Wan in [6], which also provides a more thorough review and treatment of particle filters in general.

Monte Carlo Simulation and Sequential Importance Sampling

Particle filtering is based on Monte Carlo simulation with sequential importance sampling (SIS). The overall goal is to directly implement optimal Bayesian estimation (see Eqs. (7.9)–(7.11)) by recursively approximating the complete posterior state density. In Monte Carlo simulation, a set of weighted particles (samples), drawn from the posterior distribution, is used to map integrals to discrete sums. More precisely, the posterior filtering density can be approximated by the following empirical estimate:

$$\hat{p}(\mathbf{x}_k|\mathbf{Y}_0^k) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}_k - \boldsymbol{x}_k^{(i)}),$$

where the random samples $\{\boldsymbol{x}_k^{(i)};\ i = 1, \ldots, N\}$, are drawn from $p(\mathbf{x}_k|\mathbf{Y}_0^k)$ and $\delta(\cdot)$ denotes the Dirac delta function. The posterior *filtering* density $p(\mathbf{x}_k|\mathbf{Y}_0^k)$ is a marginal of the *full* posterior density given by $p(\mathbf{X}_0^k|\mathbf{Y}_0^k)$. Consequently, any expectations of the form

$$\mathbb{E}(\mathbf{g}(\mathbf{x}_k)) = \int \mathbf{g}(\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{Y}_0^k)\, d\mathbf{x}_k \tag{7.96}$$

may be approximated by the following estimate:

$$\mathbb{E}(\mathbf{g}(\mathbf{x}_k)) \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}(\boldsymbol{x}_k^{(i)}). \tag{7.97}$$

For example, letting $g(\mathbf{x}) = \mathbf{x}$ yields the optimal MMSE estimate $\hat{\mathbf{x}}_k = \mathbb{E}[\mathbf{x}_k|\mathbf{Y}_0^k]$. The particles $\boldsymbol{x}_k^{(i)}$ are assumed to be independent and identically distributed (i.i.d) for the approximation to hold. As N goes to infinity, the estimate converges to the true expectation almost surely. Sampling from the filtering posterior is only a special case of Monte Carlo simulation, which in general deals with the complete posterior density $p(\mathbf{X}_0^k|\mathbf{Y}_0^k)$. We shall use this more general form to derive the particle filter algorithm.

It is often impossible to sample directly from the posterior density function. However, we can circumvent this difficulty by making use of importance sampling and alternatively sampling from a *known* proposal distribution $q(\mathbf{X}_0^k|\mathbf{Y}_0^k)$. The exact form of this distribution is a critical design issue, and is usually chosen in order to facilitate easy sampling. The details of this are discussed later. Given this proposal distribution, we can make use of the following substitution:

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_k(\mathbf{X}_0^k)) &= \int \mathbf{g}_k(\mathbf{X}_0^k) \frac{p(\mathbf{X}_0^k|\mathbf{Y}_0^k)}{q(\mathbf{X}_0^k|\mathbf{Y}_0^k)} q(\mathbf{X}_0^k|\mathbf{Y}_0^k)\, d\mathbf{X}_0^k \\
&= \int \mathbf{g}_k(\mathbf{X}_0^k) \frac{p(\mathbf{Y}_0^k|\mathbf{X}_0^k) p(\mathbf{X}_0^k)}{p(\mathbf{Y}_0^k) q(\mathbf{X}_0^k|\mathbf{Y}_0^k)} q(\mathbf{X}_0^k|\mathbf{Y}_0^k)\, d\mathbf{X}_0^k \\
&= \int \mathbf{g}_k(\mathbf{X}_0^k) \frac{w_k(\mathbf{X}_0^k)}{p(\mathbf{Y}_0^k)} q(\mathbf{X}_0^k|\mathbf{Y}_0^k)\, d\mathbf{X}_0^k,
\end{aligned}$$

where the variables $w_k(\mathbf{X}_0^k)$ are known as the unnormalized importance weights,

$$w_k = \frac{p(\mathbf{Y}_0^k|\mathbf{X}_0^k) p(\mathbf{X}_0^k)}{q(\mathbf{X}_0^k|\mathbf{Y}_0^k)}. \tag{7.98}$$

We can get rid of the unknown normalizing density $p(\mathbf{Y}_0^k)$ as follows:

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_k(\mathbf{X}_0^k)) &= \frac{1}{p(\mathbf{Y}_0^k)}\int \mathbf{g}_k(\mathbf{X}_0^k)w_k(\mathbf{X}_0^k)q(\mathbf{X}_0^k|\mathbf{Y}_0^k)\,d\mathbf{X}_0^k \\
&= \frac{\int \mathbf{g}_k(\mathbf{X}_0^k)w_k(\mathbf{X}_0^k)q(\mathbf{X}_0^k|\mathbf{Y}_0^k)\,d\mathbf{X}_0^k}{\int p(\mathbf{Y}_0^k|\mathbf{X}_0^k)p(\mathbf{X}_0^k)\dfrac{q(\mathbf{X}_0^k|\mathbf{Y}_0^k)}{q(\mathbf{X}_0^k|\mathbf{Y}_0^k)}\,d\mathbf{X}_0^k} \\
&= \frac{\int \mathbf{g}_k(\mathbf{X}_0^k)w_k(\mathbf{X}_0^k)q(\mathbf{X}_0^k|\mathbf{Y}_0^k)\,d\mathbf{X}_0^k}{\int w_k(\mathbf{X}_0^k)q(\mathbf{X}_0^k|\mathbf{Y}_0^k)\,d\mathbf{X}_0^k} \\
&= \frac{\mathbb{E}_{q(\cdot|\mathbf{Y}_0^k)}(w_k(\mathbf{X}_0^k)\mathbf{g}_k(\mathbf{X}_0^k))}{\mathbb{E}_{q(\cdot|\mathbf{Y}_0^k)}(w_k(\mathbf{X}_0^k))},
\end{aligned}$$

where the notation $\mathbb{E}_{q(\cdot|\mathbf{Y}_0^k)}$ has been used to emphasize that the expectations are taken over the proposal distribution $q(\cdot|\mathbf{Y}_0^k)$.

A *sequential* update to the importance weights is achieved by expanding the proposal distribution as $q(\mathbf{X}_0^k|\mathbf{Y}_0^k) = q(\mathbf{X}_0^{k-1}|\mathbf{Y}_0^{k-1})q(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k)$, where we are making the assumption that the current state is not dependent on future observations. Furthermore, under the assumption that the states correspond to a Markov process and that the observations are conditionally independent given the states, we can arrive at the recursive update:

$$w_k = w_{k-1}\frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})}{q(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k)}. \tag{7.99}$$

Equation (7.99) provides a mechanism to sequentially update the importance weights given an appropriate choice of proposal distribution, $q(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k)$. Since we can sample from the proposal distribution and evalute the likelihood $p(\mathbf{y}_k|\mathbf{x}_k)$ and transition probabilities $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, all we need to do is generate a prior set of samples and iteratively compute the importance weights. This procedure then allows us to evaluate the expectations of interest by the following estimate:

$$\mathbb{E}(\mathbf{g}(\mathbf{X}_0^k)) \approx \frac{N^{-1}\sum_{i=1}^{N}\mathbf{g}(\boldsymbol{x}_{0:k}^{(i)})w_k(\boldsymbol{x}_{0:k}^{(i)})}{N^{-1}\sum_{i=1}^{N}w_k(\boldsymbol{x}_{0:k}^{(i)})} = \sum_{i=1}^{N}\mathbf{g}(\boldsymbol{x}_{0:k}^{(i)})\tilde{w}_k(\boldsymbol{x}_{0:k}^{(i)}), \tag{7.100}$$

where the normalized importance weights $\tilde{w}_k^{(i)} = w_k^{(i)} / \sum_{j=1}^N w_k^{(j)}$ and $\boldsymbol{x}_{0:k}^{(i)}$ denotes the ith sample trajectory drawn from the proposal distribution $q(\mathbf{x}_k | \mathbf{X}_0^{k-1}, \mathbf{Y}_0^k)$. This estimate asymptotically converges if the expectation and variance of $\mathbf{g}(\mathbf{X}_0^k)$ and w_k exist and are bounded, and if the support of the proposal distribution includes the support of the posterior distribution. Thus, as N tends to infinity, the posterior density function can be approximated arbitrarily well by the point-mass estimate

$$\hat{p}(\mathbf{X}_0^k | \mathbf{Y}_0^k) = \sum_{i=1}^N \tilde{w}_k^{(i)} \delta(\mathbf{X}_0^k - \boldsymbol{x}_{0:k}^{(i)}) \tag{7.101}$$

and the posterior *filtering* density by

$$\hat{p}(\mathbf{x}_k | \mathbf{Y}_0^k) = \sum_{i=1}^N \tilde{w}_k^{(i)} \delta(\mathbf{x}_k - \boldsymbol{x}_k^{(i)}). \tag{7.102}$$

In the case of filtering, we do not need to keep the whole history of the sample trajectories, in that only the current set of samples at time k is needed to calculate expectations of the form given in Eq. (7.96) and (7.97). To do this, we simply set, $\mathbf{g}(\mathbf{X}_0^k) = \mathbf{g}(\mathbf{x}_k)$. These point-mass estimates can approximate any general distribution arbitrarily well, limited only by the number of particles used and how well the above-mentioned importance sampling conditions are met. In contrast, the posterior distribution calculated by the EKF is a minimum-variance Gaussian approximation to the true distribution, which inherently cannot capture complex structure such as multimodalities, skewness, or other higher-order moments.

Resampling and MCMC Step The sequential importance sampling (SIS) algorithm discussed so far has a serious limitation: the variance of the importance weights increases stochastically over time. Typically, after a few iterations, one of the normalized importance weights tends to unity, while the remaining weights tend to zero. A large number of samples are thus effectively removed from the sample set because their importance weights become numerically insignificant. To avoid this degeneracy, a resampling or selection stage may be used to eliminate samples with low importance weights and multiply samples with high importance weights. This is often followed by a Markov-chain Monte Carlo (MCMC) move step, which introduces sample variety without affecting the posterior distribution they represent.

A selection scheme associates to each particle $\boldsymbol{x}_k^{(i)}$ a number of "children," N_i, such that $\sum_{i=1}^N N_i = N$. Several selection schemes have been proposed in the literature, including *sampling-importance resampling* (*SIR*) [36–38], *residual resampling* [25, 39], and *minimum-variance sampling* [34].

Sampling-importance resampling (SIR) involves mapping the Dirac random measure $\{\boldsymbol{x}_k^{(i)}, \tilde{w}_k^{(i)}\}$ into an equally weighted random measure $\{\boldsymbol{x}_k^{(j)}, N^{-1}\}$. In other words, we produce N new samples all with weighting $1/N$. This can be accomplished by sampling uniformly from the discrete set $\{\boldsymbol{x}_k^{(i)}; i = 1, \ldots, N\}$ with probabilities $\{\tilde{w}_k^{(i)}; i = 1, \ldots, N\}$. Figure 7.18 gives a graphical representation of this process. This procedure effectively replicates the original $\boldsymbol{x}_k^{(i)}$ particle N_i times (N_i may be zero).

In *residual resampling* [25, 39] a two-step process is used, which makes use of SIR. In the first step, the number of children are deterministicly set using the floor function, $N_i^A = \lfloor N\tilde{w}_t^{(i)} \rfloor$. Each $\boldsymbol{x}_k^{(i)}$ particle is replicated N_i^A times. In the second step, SIR is used to select the remaining $\bar{N}_t = N - \sum_{i=1}^N N_i^A$ samples, with new weights $w_t^{'(i)} = \bar{N}_t^{-1}(\tilde{w}_t^{(i)}N - N_i^A)$. These samples form a second set N_i^B, such that $\bar{N}_t = \sum_{i=1}^N N_i^B$, and are drawn as described previously. The total number of children of each particle is then set to $N_i = N_i^A + N_i^B$. This procedure is computationally cheaper than pure SIR, and also has lower sample variance. Thus, residual resampling is used for all experiments in Section 7.6.2 (in general, we have found that the specific choice of resampling scheme does not significantly affect the performance of the particle filter).

After the selection/resampling step at time k, we obtain N particles distributed approximately according to the posterior distribution. Since the selection step favors the creation of multiple copies of the "fittest"

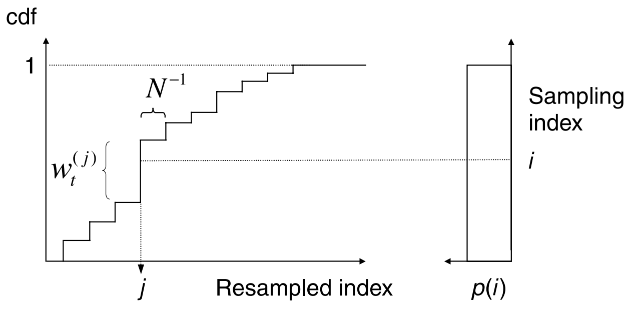


Figure 7.18 Resampling process, whereby a random measure $\{\mathbf{x}_k^{(i)}, \tilde{\mathbf{w}}_k^{(i)}\}$ is mapped into an equally weighted random measure $\{\mathbf{x}_k^{(j)}, N^{-1}\}$. The index i is drawn from a uniform distribution.

particle, many particles may end up having no children ($N_i = 0$), whereas others might end up having a large number of children, the extreme case being $N_i = N$ for a particular value i. In this case, there is a severe depletion of samples. Therefore, an additional procedure is often required to introduce sample variety after the selection step without affecting the validity of the approximation inferred. This is achieved by performing a single MCMC step on each particle. The basic idea is that if the particles are already distributed according to the posterior $p(\mathbf{x}_k|\mathbf{Y}_0^k)$ (which is the case), then applying a Markov-chain transition kernel with the same invariant distribution to each particle results in a set of new particles distributed according to the posterior of interest. However, the new particles may move to more interesting areas of the state space. Details of the MCMC step are given in [6]. For our experiments in Section 7.6.2, we found an MCMC step to be unnecessary. However, this cannot be assumed in general.

7.6.1 The Particle Filter Algorithm

The pseudo-code of a generic particle filter is presented in Table 7.6. In implementing this algorithm, the choice of the proposal distribution $q(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k)$ is the most critical design issue. The optimal proposal distribution (which minimizes the variance on the importance weights) is given by [40–43]

$$q(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k) = p(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k), \tag{7.103}$$

that is, the true conditional state density given the previous state history and all observations. Sampling from this is, of course, impractical for arbitrary densities (recall the motivation for using importance sampling in the first place). Consequently, the transition prior is the most popular choice of proposal distribution [35, 44–47]:[10]

$$q(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k) \overset{\circ}{=} p(\mathbf{x}_k|\mathbf{x}_{k-1}). \tag{7.104}$$

For example, if an additive Gaussian process noise model is used, the transition prior is simply

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{F}(\bar{\mathbf{x}}_{k-1}, 0), \mathbf{R}_{k-1}^{\mathbf{v}}). \tag{7.105}$$

[10]The notation $\overset{\circ}{=}$ denotes "chosen as," to indicate a subtle difference versus "approximation".

Table 7.6 Algorithm for the generic particle filter

1. *Initialization: $k = 0$*
   - For $i = 1, \ldots, N$, draw the states $\boldsymbol{x}_0^{(i)}$ from the prior $p(\mathbf{x}_0)$.
2. For $k = 1, 2, \ldots$

   (*a*) *Importance sampling step*
   - For $i = 1, \ldots, N$, sample $\boldsymbol{x}_k^{(i)} \sim q(\mathbf{x}_k | \mathbf{x}_{0:k-1}^{(i)}, \mathbf{Y}_0^k)$.
   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalizing constant:

   $$w_k^{(i)} = w_{k-1}^{(i)} \frac{p(\mathbf{y}_k | \boldsymbol{x}_k^{(i)}) p(\boldsymbol{x}_k^{(i)} | \boldsymbol{x}_{k-1}^{(i)})}{q(\boldsymbol{x}_k^{(i)} | \boldsymbol{x}_{0:k-1}^{(i)}, \mathbf{Y}_0^k)}. \tag{7.106}$$

   - For $i = 1, \ldots, N$, normalize the importance weights:

   $$\tilde{w}_k^{(i)} = w_k^{(i)} \left(\sum_{j=1}^{N} w_k^{(j)} \right)^{-1}.$$

   (*b*) *Selection step* (*resampling*)
   - Multiply/suppress samples $\boldsymbol{x}_k^{(i)}$ with high/low importance weights $\tilde{w}_k^{(i)}$, respectively, to obtain N random samples $\boldsymbol{x}_k^{(i)}$ approximately distributed according to $p(\mathbf{x}_k^{(i)} | \mathbf{Y}_0^k)$.
   - For $i = 1, \ldots, N$, set $w_k^{(i)} = \tilde{w}_k^{(i)} = N^{-1}$.

   (*c*) *MCMC move step* (*optional*)

   (*d*) *Output:* The output of the algorithm is a set of samples that can be used to approximate the posterior distribution as follows:

   $$\hat{p}(\mathbf{x}_k | \mathbf{Y}_0^k) = \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x}_k - \boldsymbol{x}_k^{(i)}).$$

   The optimal MMSE estimator is given as

   $$\hat{\mathbf{x}}_k = \mathbb{E}(\mathbf{x}_k | \mathbf{Y}_0^k) \approx \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_k^{(i)}.$$

   Similar expectations of the function $\mathbf{g}(\mathbf{x}_k)$ can also be calculated as a sample average.

The effectiveness of this approximation depends on how close the proposal distribution is to the true posterior distribution. If there is not sufficient overlap, only a few particles will have significant importance weights when their likelihood are evaluated.

The EKF and UKF Particle Filter An improvement in the choice of proposal distribution over the simple transition prior, which also address the problem of sample depletion, can be accomplished by moving the

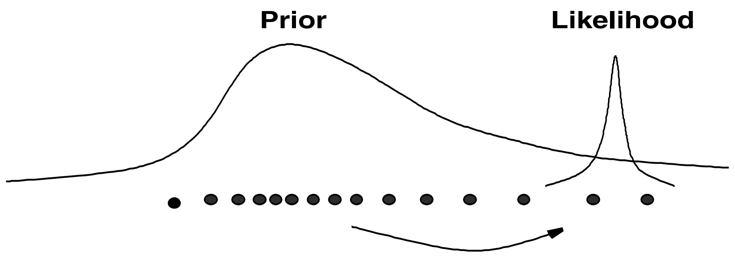


Figure 7.19 Including the most current observation into the proposal distribution, allows us to move the samples in the prior to regions of high likelihood. This is of paramount importance if the likelihood happens to lie in one of the tails of the prior distribution, or if it is too narrow (low measurement error).

particles towards the regions of high likelihood, based on the most recent observations $\mathbf{y}_k$ (see Fig. 7.19). An effective approach to accomplish this, is to use an EKF generated Gaussian approximation to the optimal proposal, that is,

$$q(\mathbf{x}_k|\mathbf{X}_0^{k-1}, \mathbf{Y}_0^k) \overset{\circ}{=} q_{\mathcal{N}}(\mathbf{x}_k|\mathbf{Y}_0^k), \tag{7.107}$$

which is accomplished by using a separate EKF to generate and propagate a Gaussian proposal distribution for each particle,

$$q_{\mathcal{N}}(\mathbf{x}_k^{(i)}|\mathbf{Y}_0^k) = \mathcal{N}(\bar{\mathbf{x}}_k^{(i)}, \mathbf{P}_k^{(i)}), \qquad i = 1, \ldots, N. \tag{7.108}$$

That is, at time k one uses the EKF equations, with the new data, to compute the mean and covariance of the importance distribution for each particle from the previous time step $k-1$. Next, we redraw the ith particle (at time k) from this new updated distribution. While still making a Gaussian assumption, the approach provides a better approximation to the optimal conditional proposal distribution and has been shown to improve performance on a number of applications [33, 48].

By replacing the EKF with the UKF, we can more accurately propagate the mean and covariance of the Gaussian approximation to the state distribution. Distributions generated by the UKF will have a greater support overlap with the true posterior distribution than the overlap achieved by the EKF estimates. In addition, scaling parameters used for sigma-point selection can be optimised to capture certain characteristic of the prior distribution if known; e.g. the algorithm can be modified to work with distributions that have heavier tails than Gaussian distributions such as Cauchy or Student-t distributions. The new filter that results from using a UKF for proposal distribution generation within a particle filter framework is called the *unscented particle filter* (UPF). Referring to the algorithm in Table 7.6 for the generic particle filter, the first item in the importance sampling step,

- For $i = 1, \ldots, N$, sample $\boldsymbol{x}_k^{(i)} \sim q(\mathbf{x}_k | \mathbf{x}_{0:k-1}^{(i)}, \mathbf{Y}_0^k)$,

is replaced with the following UKF update:

- For $i = 1, \ldots, N$:
  - Update the prior $(k-1)$ distribution for each particle with the UKF:
    - Calculate sigma points:

$$\boldsymbol{\mathcal{X}}_{k-1}^{(i)a} = [\bar{\boldsymbol{x}}_{k-1}^{(i)a} \quad \bar{\boldsymbol{x}}_{k-1}^{(i)a} + \gamma\sqrt{\mathbf{P}_{k-1}^{(i)a}} \quad \bar{\boldsymbol{x}}_{k-1}^{(i)a} - \gamma\sqrt{\mathbf{P}_{k-1}^{(i)a}}]. \tag{7.109}$$

    - Propagate particle into future (time update):

$$\boldsymbol{\mathcal{X}}_{k|k-1}^{(i)x} = \mathbf{F}(\boldsymbol{\mathcal{X}}_{k-1}^{(i)x}, \mathbf{u}_k, \boldsymbol{\mathcal{X}}_{k-1}^{(i)v}), \qquad \bar{\boldsymbol{x}}_{k|k-1}^{(i)} = \sum_{j=0}^{2L} W_j^{(m)} \mathcal{X}_{j,k|k-1}^{(i)x}, \tag{7.110}$$

$$\mathbf{P}_{k|k-1}^{(i)} = \sum_{j=0}^{2L} W_j^{(c)} (\mathcal{X}_{j,k|k-1}^{(i)x} - \bar{\boldsymbol{x}}_{k|k-1}^{(i)})(\mathcal{X}_{j,k|k-1}^{(i)x} - \bar{\boldsymbol{x}}_{k|k-1}^{(i)})^T \tag{7.111}$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1}^{(i)} = \mathbf{H}(\boldsymbol{\mathcal{X}}_{k|k-1}^{(i)x}, \boldsymbol{\mathcal{X}}_{k-1}^{(i)n}),$$

$$\bar{\mathbf{y}}_{k|k-1}^{(i)} = \sum_{j=0}^{2L} W_j^{(m)} \mathcal{Y}_{j,k|k-1}^{(i)}. \tag{7.112}$$

    - Incorporate new observation (measurement update):

$$\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} = \sum_{j=0}^{2L} W_j^{(c)} (\mathcal{Y}_{j,k|k-1}^{(i)} - \bar{\mathbf{y}}_{k|k-1}^{(i)})(\mathcal{Y}_{j,k|k-1}^{(i)} - \bar{\mathbf{y}}_{k|k-1}^{(i)})^T, \tag{7.113}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{J=0}^{2L} W_j^{(c)} (\mathcal{X}_{j,k|k-1}^{(i)} - \bar{\boldsymbol{x}}_{k|k-1}^{(i)})(\mathcal{Y}_{j,k|k-1}^{(i)} - \bar{\mathbf{y}}_{k|k-1}^{(i)})^T, \tag{7.114}$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}^{-1},$$

$$\bar{\boldsymbol{x}}_k^{(i)} = \bar{\boldsymbol{x}}_{k|k-1}^{(i)} + \mathbf{K}_k(\mathbf{y}_k - \bar{\mathbf{y}}_{k|k-1}^{(i)}), \tag{7.115}$$

$$\mathbf{P}_k^{(i)} = \mathbf{P}_{k|k-1}^{(i)} - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} \mathbf{K}_k^T. \tag{7.116}$$

  - Sample $\boldsymbol{x}_k^{(i)} \sim q(\mathbf{x}_k^{(i)} | \mathbf{x}_{0:k-1}^{(i)}, \mathbf{Y}_0^k) \approx \mathcal{N}(\bar{\boldsymbol{x}}_k^{(i)}, \mathbf{P}_k^{(i)})$.

All other steps in the particle filter formulation remain unchanged.

7.6.2 UPF Experiments

The performance of the UPF is evaluated on two estimation problems. The first problem is a synthetic scalar estimation problem and the second is a real-world problem concerning the pricing of financial instruments.

Synthetic Experiment For this experiment, a time series was generated by the following process model:

$$x_{k+1} = 1 + \sin(\omega \pi t) + \phi_1 x_k + v_k, \tag{7.117}$$

where v_k is a Gamma $\mathcal{G}a(3, 2)$ random variable modeling the process noise, and $\omega = 0.04$ and $\phi_1 = 0.5$ are scalar parameters. A nonstationary observation model,

$$y_k = \begin{cases} \phi_2 x_k^2 + n_k, & t \leq 30, \\ \phi_3 x_k - 2 + n_k & t > 30, \end{cases} \tag{7.118}$$

is used, with $\phi_2 = 0.2$ and $\phi_3 = 0.5$. The observation noise, n_k, is drawn from a Gaussian distribution $\mathcal{N}(0, 0.00001)$. Given only the noisy observations y_k, the different filters were used to estimate the underlying clean state sequence x_k for $k = 1 \ldots 60$. The experiment was repeated 100 times with random re-initialization for each run. All of the particle filters used 200 particles and residual resampling. The UKF parameters were set to $\alpha = 1$, $\beta = 0$ and $\kappa = 2$. These parameters are optimal for the scalar case. Table 7.7 summarizes the performance of the different filters. The table shows the means and variances of the mean-square error (MSE) of the state estimates. Figure 7.20 compares the estimates generated from a

Table 7.7 State-estimation experiment results: the mean and variance of the MSE were calculated over 100 independent runs

	MSE	
Algorithm	Mean	Variance
Extended Kalman filter (EKF)	0.374	0.015
Unscented Kalman filter (UKF)	0.280	0.012
Particle filter: generic	0.424	0.053
Particle filter: MCMC move step	0.417	0.055
Particle filter: EKF proposal	0.310	0.016
Particle filter: EKF proposal and MCMC move step	0.307	0.015
Particle filter: UKF proposal ("*unscented particle filter*")	0.070	0.006
Particle filter: UKF proposal and MCMC move step	0.074	0.008

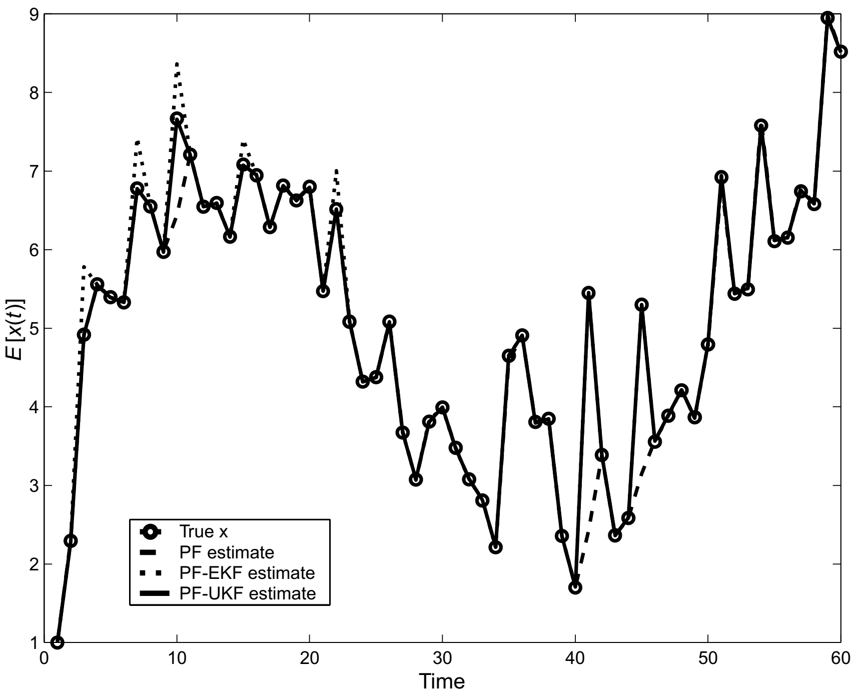


Figure 7.20 Plot of estimates generated by the different filters on the synthetic state-estimation experiment.

single run of the different particle filters. The superior performance of the *unscented particle filter* (UPF) is clear.

Pricing Financial Options Derivatives are financial instruments whose value depends on some basic underlying cash product, such as interest rates, equity indices, commodities, foreign exchange, or bonds [49]. A call option allows the holder to buy a cash product, at a specified date in the future, for a price determined in advance. The price at which the option is exercised is known as the strike price, while the date in which the option lapses is often referred to as the maturity time. Put options, on the other hand, allow the holder to sell the underlying cash product. In their seminal work [50], Black and Scholes derived the following industry standard equations for pricing European call and put options:

$$C = S\mathcal{N}_c(d_1) - Xe^{-rt_m}\mathcal{N}_c(d_2), \tag{7.119}$$

$$P = -S\mathcal{N}_c(-d_1) + Xe^{-rt_m}\mathcal{N}_c(-d_2), \tag{7.120}$$

where C denotes the price of a call option, P the price of a put option, S the current value of the underlying cash product, X the desired strike price, t_m the time to maturity, and $\mathcal{N}_c(.)$ the cumulative normal distribution, and d_1 and d_2 are given by

$$d_1 = \frac{\ln(S/X) + (r + \sigma^2/2)t_m}{\sigma\sqrt{t_m}},$$
$$d_2 = d_1 - \sigma\sqrt{t_m},$$

where σ is the (unknown) volatility of the cash product and r is the risk-free interest rate.

The volatility, σ, is usually estimated from a small moving window of data over the most recent 50–180 days [49]. The risk-free interest rate r is often estimated by monitoring interest rates in the bond markets. Our approach is to treat r and σ as the hidden states, and C and P as the output

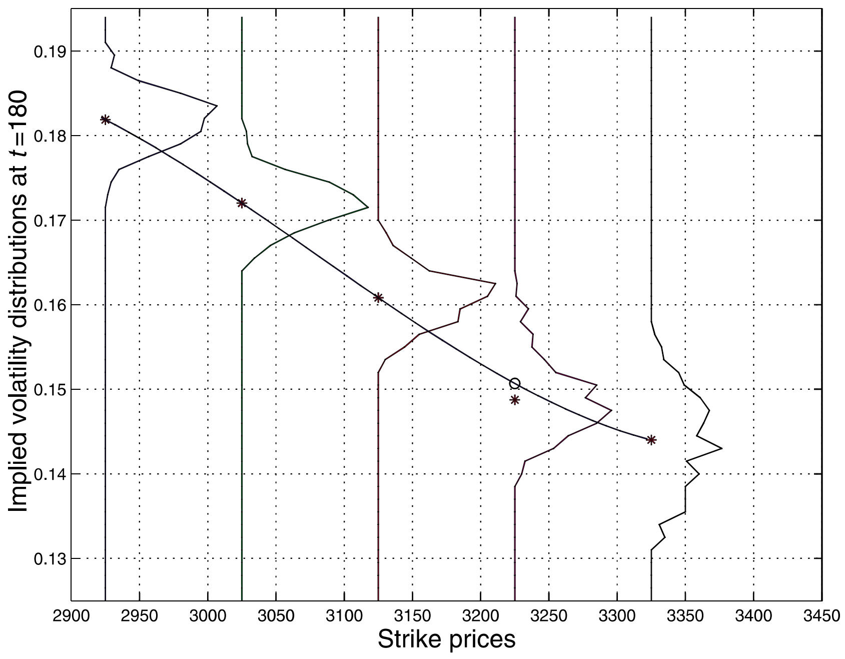


Figure 7.21 Probability smile for options on the FTSE-100 index (1994). Although the volatility smile indicates that the option with strike price equal to 3225 is underpriced, the shape of the probability gives us a warning against the hypothesis that the option is under-priced. Posterior mean estimates were obtained with the Black–Scholes model and particle filter ($*$), a fourth-order polynomial fit (–), and hypothesized volatility ($\circ$).

observations. S and t_m are treated as known control signals (input observations). This represents a parameter estimation problem, with the nonlinear observation given by Eqs. (7.119) or (7.120). This allows us to compute daily complete probability distributions for r and σ and to decide whether the current value of an option in the market is being either over-priced or under-priced. See [51] and [52] for details.

As an example, Figure 7.21 shows the implied probability density function of each volatility against several strike prices using five pairs of call and put option contracts on the British FTSE-100 index (from February 1994 to December 1994). Figure 7.22 shows the estimated volatility and interest rate for a contract with a strike price of 3225. In Table 7.8, we compare the one-step-ahead normalized square errors on a pair of options with strike price 2925. The square errors were only measured over the last 100 days of trading, so as to allow the algorithms to converge. The experiment was repeated 100 times with 100 particles in each particle filter (the mean value is reported; all variance were essentially zero). In this example, both the EKF and UKF approaches to improving the proposal distribution lead to a significant improvement over the standard particle filters. The main advantage of the UKF over the

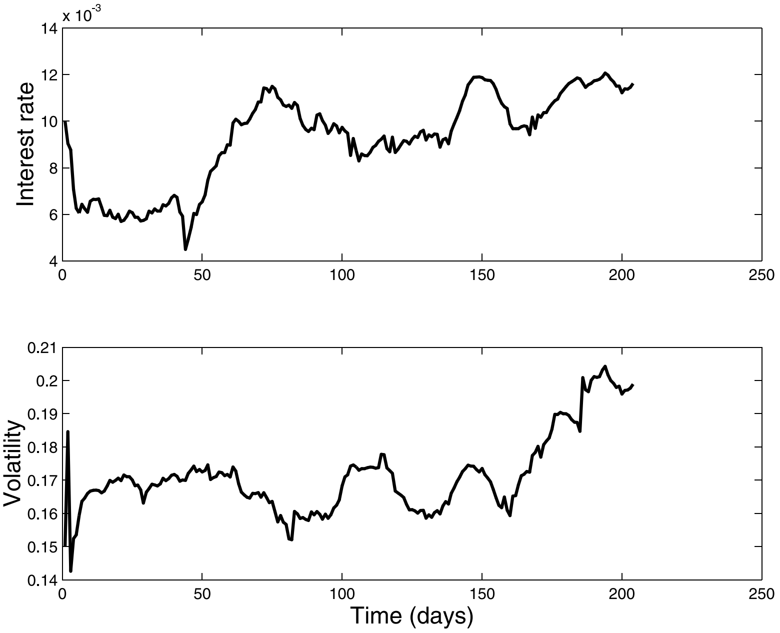


Figure 7.22 Estimated interest rate and volatility.

Table 7.8 One-step-ahead normalized square errors over 100 runs. The trivial prediction is obtained by assuming that the price on the following day corresponds to the current price

Option type	Algorithm	Mean NSE
Call	Trivial	0.078
	Extended Kalman filter (EKF)	0.037
	Unscented Kalman filter (UKF)	0.037
	Particle filter: generic	0.037
	Particle filter: EKF proposal	0.009
	Unscented particle filter	0.009
Put	Trivial	0.035
	Extended Kalman filter (EKF)	0.023
	Unscented Kalman filter (UKF)	0.023
	Particle filter: generic	0.023
	Particle filter: EKF proposal	0.007
	Unscented particle filter	0.008

EKF is the ease of implementation, which avoids the need to analytically differentiate the Black–Scholes equations.

7.7 CONCLUSIONS

The EKF has been widely accepted as a standard tool in the control and machine-learning communities. In this chapter, we have presented an alternative to the EKF using the unscented Kalman filter. The UKF addresses many of the approximation issues of the EKF, and consistently achieves an equal or better level of performance at a comparable level of complexity. The performance benefits of the UKF-based algorithms have been demonstrated in a number of application domains, including state estimation, dual estimation, and parameter estimation.

There are a number of clear advantages to the UKF. First, the mean and covariance of the state estimate is calculated to second order or better, as opposed to first order in the EKF. This provides for a more accurate implementation of the optimal recursive estimation equations, which is the basis for both the EKF and UKF. While equations specifying the UKF may appear more complicated than the EKF, the actual computational complexity is equivalent. For state estimation, both algorithms are in general of order L^3 (where L is the dimension of the state). For parameter estimation, both algorithms are of order L^2 (where L is the number of parameters). An efficient recursive square-root implementation (see Appendix B) was necessary to achieve the level of complexity in the parameter-estimation case. Furthermore, a distinct advantage of the UKF is its ease of implementation. In contrast to the EKF, no analytical derivatives (Jacobians or Hessians) need to be calculated. The utility of this is especially valuable in situations where the system is a "black box" model in which the internal dynamic equations are unavailable. In order to apply an EKF to such systems, derivatives must be found either from a principled analytical re-derivation of the system, or through costly and often inaccurate numerical methods (e.g., by perturbation). In contrast, the UKF relies on only functional evaluations (inputs and outputs) through the use of deterministically drawn samples from the prior distribution of the state random variable. From a coding perspective, this also allows for a much more general and modular implementation.

Even though the UKF has clear advantages over the EKF, there are still a number of limitations. As in the EKF, it makes a Gaussian assumption on the probability density of the state random variable. Often this assumption is valid, and numerous real-world applications have been successfully implemented based on this assumption. However, for certain problems (e.g., multimodal object tracking), a Gaussian assumption will not suffice, and the UKF (or EKF) cannot be applied with confidence. In such examples, one has to resort to more powerful, but also more computationally expensive, filtering paradigms such as particle filters (see Section 7.6). Finally, another implementation limitation leading to some uncertainty, is the necessity to choose the three unscented transformation parameters (i.e., α, β, and κ). While we have attempted to provide some guidelines on how to choose these parameters, the optimal selection clearly depends on the specifics of the problem at hand, and is not fully understood. In general, the choice of settings does not appear critical for state estimation, but has a greater affect on performance and convergence properties for parameter estimation. Our current work focuses on addressing this issue through developing a unified and adaptive way of calculating the optimal value of these parameters. Other areas of open research include utilizing the UKF for estimation of noise covariances, extension of the UKF to recurrent architectures that may require dynamic derivatives (see Chapter 2 and 5), and the use of the UKF and smoother in the expectation–maximization algorithm (see Chapter 6). Clearly, we have only begun to scratch the surface of the numerous applications that can benefit with use of the UKF.

APPENDIX A: ACCURACY OF THE UNSCENTED TRANSFORMATION

In this appendix, we show how the unscented transformation achieves second-order accuracy in the prediction of the posterior mean and covariance of a random variable that undergoes a nonlinear transformation. For the purpose of this analysis, we assume that all nonlinear transformations are analytic across the domain of all possible values of $\mathbf{x}$. This condition implies that the nonlinear function can be expressed as a multidimensional Taylor series consisting of an arbitrary number of terms. As the number of terms in the sum tend to infinity, the residual of the series tends to zero. This implies that the series always converges to the true value of the function.

If we consider the prior variable $\mathbf{x}$ as being perturbed about a mean $\bar{\mathbf{x}}$ by a zero-mean disturbance $\delta\mathbf{x}$ with covariance $\mathbf{P}_\mathbf{x}$, then the Taylor series expansion of the nonlinear transformation $f(\mathbf{x})$ about $\bar{\mathbf{x}}$ is

$$f(\mathbf{x}) = f(\bar{\mathbf{x}} + \delta\mathbf{x}) = \sum_{n=0}^{\infty}\left[\frac{(\delta\mathbf{x}\cdot\nabla_\mathbf{x})^n f(\mathbf{x})}{n!}\right]_{\mathbf{x}=\bar{\mathbf{x}}}. \tag{7.121}$$

If we define the operator $\mathbf{D}^n_{\delta\mathbf{x}} f$ as

$$\mathbf{D}^n_{\delta\mathbf{x}} f \triangleq [(\delta\mathbf{x}\cdot\nabla_\mathbf{x})^n f(\mathbf{x})]_{\mathbf{x}=\bar{\mathbf{x}}}, \tag{7.122}$$

then the Taylor series expansion of the nonlinear transformation $\mathbf{y} = f(\mathbf{x})$ can be written as

$$\mathbf{y} = f(\mathbf{x}) = f(\bar{\mathbf{x}}) + \mathbf{D}_{\delta\mathbf{x}} f + \frac{1}{2}\mathbf{D}^2_{\delta\mathbf{x}} f + \frac{1}{3!}\mathbf{D}^3_{\delta\mathbf{x}} f + \frac{1}{4!}\mathbf{D}^4_{\delta x} f + \cdots. \tag{7.123}$$

Accuracy of the Mean

The true mean of $\mathbf{y}$ is given by

$$\bar{\mathbf{y}} = \mathbb{E}[\mathbf{y}] = \mathbb{E}[f(\mathbf{x})] \tag{7.124}$$

$$= \mathbb{E}\left[f(\bar{\mathbf{x}}) + \mathbf{D}_{\delta\mathbf{x}} f + \frac{1}{2}\mathbf{D}^2_{\delta\mathbf{x}} f + \frac{1}{3!}\mathbf{D}^3_{\delta\mathbf{x}} f + \frac{1}{4!}\mathbf{D}^3_{\delta\mathbf{x}} f + \cdots\right]. \tag{7.125}$$

If we assume that $\mathbf{x}$ is a symmetrically distributed[11] random variable, then all odd moments will be zero. Also note that $\mathbb{E}[\delta\mathbf{x}\ \delta\mathbf{x}^T] = \mathbf{P}_\mathbf{x}$. Given this, the mean can be reduced further to

$$\bar{\mathbf{y}} = f(\bar{\mathbf{x}}) + \frac{1}{2}[(\nabla^T \mathbf{P}_\mathbf{x} \nabla) f(\mathbf{x})]_{\mathbf{x}=\bar{\mathbf{x}}} + \mathbb{E}\left[\frac{1}{4!}\mathbf{D}^4_{\delta\mathbf{x}} f + \frac{1}{6!}\mathbf{D}^6_{\delta\mathbf{x}} f + \cdots\right]. \quad (7.126)$$

The UT calculates the posterior mean from the propagated sigma points using Eq. (7.32). The sigma points are given by

$$\begin{aligned} \mathcal{X}_i &= \bar{\mathbf{x}} \pm (\sqrt{L+\lambda})\sigma_i, \\ &= \bar{\mathbf{x}} \pm \tilde{\sigma}_i \end{aligned}$$

where σ_i denotes the ith column[12] of the matrix square root of $\mathbf{P}_\mathbf{x}$. This implies that $\sum_{i=1}^{L}(\sigma_i \sigma_i^T) = \mathbf{P}_\mathbf{x}$. Given this formulation of the sigma points, we can again write the propagation of each point through the nonlinear function as a Taylor series expansion about $\bar{\mathbf{x}}$:

$$\mathcal{Y}_i = f(\mathcal{X}_i) = f(\bar{\mathbf{x}}) + \mathbf{D}_{\tilde{\sigma}_i} f + \frac{1}{2}\mathbf{D}^2_{\tilde{\sigma}_i} f + \frac{1}{3!}\mathbf{D}^3_{\tilde{\sigma}_i} f + \frac{1}{4!}\mathbf{D}^4_{\tilde{\sigma}_i} f + \cdots.$$

Using Eq. (7.32), the UT predicted mean is

$$\begin{aligned} \bar{\mathbf{y}}_{UT} &= \frac{\lambda}{L+\lambda} f(\bar{\mathbf{x}}) + \frac{1}{2(L+\lambda)}\sum_{i=1}^{2L} \\ &\quad \times \left[f(\bar{\mathbf{x}}) + \mathbf{D}_{\tilde{\sigma}_i} f + \frac{1}{2}\mathbf{D}^2_{\tilde{\sigma}_i} f + \frac{1}{3!}\mathbf{D}^3_{\tilde{\sigma}_i} f + \frac{1}{4!}\mathbf{D}^4_{\tilde{\sigma}_i} f + \cdots \right] \\ &= f(\bar{\mathbf{x}}) + \frac{1}{2(L+\lambda)}\sum_{i=1}^{2L}\left(\mathbf{D}_{\tilde{\sigma}_i} f + \frac{1}{2}\mathbf{D}^2_{\tilde{\sigma}_i} f + \frac{1}{3!}\mathbf{D}^3_{\tilde{\sigma}_i} f + \frac{1}{4!}\mathbf{D}^4_{\tilde{\sigma}_i} f + \cdots \right). \end{aligned}$$

Since the sigma points are symmetrically distributed around $\bar{\mathbf{x}}$, all the odd moments are zero. This results in the simplification

$$\bar{\mathbf{y}}_{UT} = f(\bar{\mathbf{x}}) + \frac{1}{2(L+\lambda)}\sum_{i=1}^{2L}\left(\frac{1}{2}\mathbf{D}^2_{\tilde{\sigma}_i} f + \frac{1}{4!}\mathbf{D}^4_{\tilde{\sigma}_i} f + \frac{1}{6!}\mathbf{D}^6_{\tilde{\sigma}_i} f + \cdots \right),$$

[11]This includes probability distributions such as Gaussian, Student-t, etc.
[12]See Section 7.3 for details of exactly how the sigma points are calculated.

and since

$$\begin{aligned}\frac{1}{2(L+\lambda)}\sum_{i=1}^{2L}\frac{1}{2}\mathbf{D}_{\tilde{\sigma}_i}^2 f &= \frac{1}{2(L+\lambda)}(\nabla f)^T\left[\sum_{i=1}^{2L}(\sqrt{L+\lambda}\sigma_i\sigma_i^T\sqrt{L+\lambda})\right](\nabla f)\\ &= \frac{L+\lambda}{2(L+\lambda)}(\nabla f)^T\frac{1}{2}\left[\sum_{i=1}^{2L}\sigma_i\sigma_i^T\right](\nabla f)\\ &= \frac{1}{2}[(\nabla^T\mathbf{P_x}\nabla)f(\mathbf{x})]_{\mathbf{x}=\bar{\mathbf{x}}},\end{aligned}$$

the UT predicted mean can be further simplified to

$$\begin{aligned}\bar{\mathbf{y}}_{UT} &= f(\bar{\mathbf{x}}) + \frac{1}{2}[(\nabla^T\mathbf{P_x}\nabla)f(\mathbf{x})]_{\mathbf{x}=\bar{\mathbf{x}}}\\ &+ \frac{1}{2(L+\lambda)}\sum_{i=1}^{2L}\left(\frac{1}{4!}\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_i}^4 f + \frac{1}{6!}\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_i}^6 f + \cdots\right). \qquad (7.127)\end{aligned}$$

When we compare Eqs. (7.127) and (7.126), we can clearly see that the true posterior mean and the mean calculated by the UT agrees exactly to the third order and that errors are only introduced in the first and higher-order terms. The magnitudes of these errors depends on the choice of the composite scaling parameter λ as well as the higher-order derivatives of f. In contrast, a linearization approach calculates the posterior mean as

$$\bar{\mathbf{y}}_{\mathrm{LIN}} = f(\bar{\mathbf{x}}), \qquad (7.128)$$

which only agrees with the true posterior mean up to the first order. Julier and Uhlman [2] show that, on a term-by-term basis, the errors in the higher-order terms of the UT are consistently smaller than those for linearization.

Accuracy of the Covariance

The true posterior covariance is given by

$$\mathbf{P}_{\mathbf{y}} = \mathbb{E}[(\mathbf{y}-\bar{\mathbf{y}}_T)(\mathbf{y}-\bar{\mathbf{y}}_T)^T] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] - \bar{\mathbf{y}}\bar{\mathbf{y}}^T \qquad (7.129)$$

where the expectation is taken over the distribution of **y**. Substituting Eqs. (7.123) and (7.125) into (7.129), and recalling that all odd moments of $\delta\mathbf{x}$ are zero owing to symmetry, we can write the true posterior covariance as

$$\begin{aligned}\mathbf{P}_\mathbf{y} = \mathcal{A}_\mathbf{x}\mathbf{P}_\mathbf{x}\mathcal{A}_\mathbf{x}^T &= \frac{1}{4}\{[(\nabla^T\mathbf{P}_\mathbf{x}\nabla)f(\mathbf{x})][(\nabla^T\mathbf{P}_\mathbf{x}\nabla)\mathbf{f}(\mathbf{x})]^T\}_{\mathbf{x}=\bar{\mathbf{x}}} \\ &+ \mathbb{E}\underbrace{\left[\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{i!j!}\mathbf{D}_{\delta\mathbf{x}}^i f(\mathbf{D}_{\delta\mathbf{x}}^j f)^T\right]}_{ij>1} \\ &- \underbrace{\left[\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{(2i)!(2j)!}\mathbb{E}[\mathbf{D}_\mathbf{x}^{2i} f]\mathbb{E}[\mathbf{D}_{\sigma\mathbf{x}}^{2j} f]^T],\right]}_{ij>1}\end{aligned} \tag{7.130}$$

where $\mathcal{A}_\mathbf{x}$ is the Jacobian matrix of $f(\mathbf{x})$ evaluated at $\bar{\mathbf{x}}$. It can be shown (using a similar approach as for the posterior mean) that the posterior covariance calculated by the UT is given by

$$\begin{aligned}(\mathbf{P}_\mathbf{y})_{UT} &= \mathcal{A}_\mathbf{x}\mathbf{P}_\mathbf{x}\mathcal{A}_\mathbf{x}^T - \frac{1}{4}\{[(\nabla^T\mathbf{P}_\mathbf{x}\nabla)f(\mathbf{x})][(\nabla^T\mathbf{P}_\mathbf{x}\nabla)f(\mathbf{x})]^T\}_{\mathbf{x}=\bar{\mathbf{x}}} \\ &+ \frac{1}{2(L+\lambda)}\sum_{k=1}^{2L}\underbrace{\left[\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{i!j!}\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_k}^i f(\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_k}^j f)^T\right]}_{ij>1} \\ &- \underbrace{\left[\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{(2i)!(2j)!4(L+\lambda)^2}\sum_{k=1}^{2L}\sum_{m=1}^{2L}\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_k}^{2i} f(\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_m}^{2j} f)^T\right]}_{ij>1}.\end{aligned} \tag{7.131}$$

Comparing Eqs. (7.130) and (7.131), it is clear that the UT again calculates the posterior covariance accurately to the first two terms, with errors only introduced in the fourth- and higher-order moments. Julier and Uhlmann [2] show how the absolute term-by-term errors of these higher-order moments are again consistently smaller for the UT than for the linearized case that truncates the Taylor series after the first term, that is,

$$(\mathbf{P}_y)_{\mathrm{LIN}} = \mathcal{A}_\mathbf{x}\mathbf{P}_\mathbf{x}\mathcal{A}_\mathbf{x}^T. \tag{7.132}$$

For this derivation, we have assumed the value of the β parameter in the UT to be zero. If prior knowledge about the shape of the prior distribution of **x** is known, β can be set to a non-zero value that minimizes the error in some of the higher (≥ 4) order moments. Julier [53] shows how the error in the kurtosis of the posterior distribution is minimized for a Gaussian $\mathbf{x}$ when $\beta = 2$.

APPENDIX B: EFFICIENT SQUARE-ROOT UKF IMPLEMENTATIONS

In the standard Kalman implementation, the state (or parameter) covariance $\mathbf{P}_k$ is recursively calculated. The UKF requires taking the matrix square-root $\mathbf{S}_k\mathbf{S}_k^T = \mathbf{P}_k$, at each time step, which is $\mathcal{O}(\frac{1}{6}L^3)$ using a Cholesky factorization. In the square-root UKF (SR-UKF), $\mathbf{S}_k$ will be propagated directly, avoiding the need to refactorize at each time step. The algorithm will in general still be $\mathcal{O}(L^3)$ for state estimation, but with improved numerical properties (e.g., guaranteed positive-semidefiniteness of the state covariances), similar to those of standard square-root Kalman filters [20]. However, for the special state-space formulation of parameter estimation, an $\mathcal{O}(L^2)$ implementation becomes possible (equivalent complexity to EKF parameter estimation).

The square-root form of the UKF makes use of three powerful linear-algebra techniques,[13] *QR decomposition*, *Cholesky factor updating*, and *efficient least squares*, which we briefly review below:

- *QR decomposition* The QR decomposition or factorization of a matrix $\mathbf{A} \in \mathbb{R}^{L\times N}$ is given by, $\mathbf{A}^T = \mathbf{QR}$, where $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is orthogonal, $\mathbf{R} \in \mathbb{R}^{N\times L}$ is upper-triangular, and $N \geq L$. The upper-triangular part of $\mathbf{R}$, $\tilde{\mathbf{R}}$, is the transpose of the Cholesky factor of $\mathbf{P} = \mathbf{AA}^T$, that is, $\tilde{\mathbf{R}} = \mathbf{S}^T$, such that $\tilde{\mathbf{R}}^T\tilde{\mathbf{R}} = \mathbf{AA}^T$. We use the shorthand notation qr$\{\cdot\}$ to donate a QR decomposition of a matrix where only $\tilde{\mathbf{R}}$ is returned. The computational complexity of a QR decomposition is $\mathcal{O}(NL^2)$. Note that performing a Cholesky factorization directly on $\mathbf{P} = \mathbf{AA}^T$ is $\mathcal{O}(\frac{1}{6}L^3)$ plus $\mathcal{O}(NL^2)$ to form $\mathbf{AA}^T$.
- *Cholesky factor updating* If $\mathbf{S}$ is the original lower-triangular Cholesky factor of $\mathbf{P} = \mathbf{AA}^T$, then the Cholesky factor of the rank-1 update (or downdate) $\mathbf{P} \pm \sqrt{v}\mathbf{uu}^T$ is denoted by $\mathbf{S} =$ cholupdate$\{\mathbf{S}, \mathbf{u}, \pm v\}$. If $\mathbf{u}$ is a matrix and not a vector, then the result is M consecutive updates of the Cholesky factor using the M columns of $\mathbf{u}$. This algorithm (available in *Matlab* as `cholupdate`) is only $\mathcal{O}(L^2)$ per update.

[13] See [54] for theoretical and implementation details.

- *Efficient least squares* The solution to the equation $(\mathbf{A}\mathbf{A}^T)\mathbf{x} = \mathbf{A}^T\mathbf{b}$ also corresponds to the solution of the overdetermined least-squares problem $\mathbf{A}\mathbf{x} = \mathbf{b}$. This can be solved efficiently using a QR decomposition with pivoting (implemented in *Matlab*'s "/" operator).

The complete specifications for the new square-root filters are given in Table 7.9 for state estimation and Table 7.10 for parameter estimation. Below we describe the key parts of the square-root algorithms, and how they contrast with the standard implementations. Experimental results and further discussion are presented in [7] and [55].

Square-Root State Estimation

As in the original UKF, the filter is initialized by calculating the matrix square root of the state covariance once via a Cholesky factorization, Eq. (7.133). However, the propagated and updated Cholesky factor is then used in subsequent iterations to directly form the sigma points. In Eq. (7.138) the *time update* of the Cholesky factor, $\mathbf{S}^-$, is calculated using a QR decomposition of the compound matrix containing the weighted propagated sigma points and the matrix square root of the additive process noise covariance. The subsequent Cholesky update (or downdate) in Eq. (7.137) is necessary since the zeroth weight, $W_0^{(c)}$, may be negative. These two steps replace the *time-update* of $\mathbf{P}^-$ in Eq. (7.55), and is also $\mathcal{O}(L^3)$.

The same two-step approach is applied to the calculation of the Cholesky factor, $\mathbf{S}_{\tilde{\mathbf{y}}}$, of the observation error covariance in Eqs. (7.142) and (7.143). This step is $\mathcal{O}(LM^2)$, where M is the observation dimension. In contrast to the way that Kalman gain is calculated in the standard UKF (see Eq. (7.61)), we now use two nested inverse (or *least-squares*) solutions to the following expansion of Eq. (7.60): $\mathcal{K}_k(\mathbf{S}_{\tilde{\mathbf{y}}_k}\mathbf{S}_{\tilde{\mathbf{y}}_k}^T) = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}$. Since $\mathbf{S}_{\tilde{\mathbf{y}}}$ is square and triangular, efficient "back-substitutions" can be used to solve for $\mathcal{K}_k$ directly without the need for a matrix inversion.

Finally, the posterior measurement update of the Cholesky factor of the state covariance is calculated in Eq. (7.147) by applying M sequential Cholesky downdates to $\mathbf{S}_k^-$. The downdate vectors are the columns of $\mathbf{U} = \mathcal{K}_k\mathbf{S}_{\tilde{\mathbf{y}}_k}$. This replaces the posterior update of $\mathbf{P}_k$ in Eq. (7.63), and is also $\mathcal{O}(LM^2)$.

Square-Root Parameter Estimation

The parameter-estimation algorithm follows a similar framework to that of the state-estimation square-root UKF. However, an $\mathcal{O}(ML^2)$ algorithm, as

Table 7.9 Square-Root UKF for state estimation

Initialize with

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \qquad \mathbf{S}_0 = \mathrm{chol}\left\{\mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]\right\}. \tag{7.133}$$

For $k \in \{1, \ldots, \infty\}$,
The sigma-point calculation and time update are given by

$$\mathcal{X}_{k-1} = [\hat{\mathbf{x}}_{k-1} \quad \hat{\mathbf{x}}_{k-1} + \gamma\mathbf{S}_k \quad \hat{\mathbf{x}}_{k-1} - \gamma\mathbf{S}_k], \tag{7.134}$$

$$\mathcal{X}^*_{k|k-1} = \mathbf{F}(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}), \tag{7.135}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}, \tag{7.136}$$

$$\mathbf{S}_k^- = \mathrm{qr}\left\{\left[\sqrt{W_1^{(c)}}(\mathcal{X}^*_{1:2L,k|k-1} - \hat{\mathbf{x}}_k^-) \quad \sqrt{\mathbf{R}^\mathbf{v}}\right]\right\} \tag{7.137}$$

$$\mathbf{S}_k^- = \mathrm{cholupdate}\{\mathbf{S}_k^-, \mathcal{X}^*_{0,k} - \hat{\mathbf{x}}_k^-, W_0^{(c)}\}, \tag{7.138}$$

(augment sigma points)[14]

$$\mathcal{X}_{k|k-1} = [\mathcal{X}^*_{k|k-1} \quad \mathcal{X}^*_{0,k|k-1} + \gamma\sqrt{\mathbf{R}^\mathbf{v}} \quad \mathcal{X}^*_{0,k|k-1} - \gamma\sqrt{\mathbf{R}^\mathbf{v}}] \tag{7.139}$$

$$\mathcal{Y}_{k|k-1} = \mathbf{H}(\mathcal{X}_{k|k-1}) \tag{7.140}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \tag{7.141}$$

and the measurement update equations are

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \mathrm{qr}\left\{\left[\sqrt{W_1^{(c)}}(\mathcal{Y}_{1:2L,k} - \hat{\mathbf{y}}_k) \quad \sqrt{\mathbf{R}_k^\mathbf{n}}\right]\right\}, \tag{7.142}$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \mathrm{cholupdate}\{\mathbf{S}_{\tilde{\mathbf{y}}_k}, \mathcal{Y}_{0,k} - \hat{\mathbf{y}}_k, W_0^{(c)}\} \tag{7.143}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T, \tag{7.144}$$

$$\mathcal{K}_k = (\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}/\mathbf{S}_{\tilde{\mathbf{y}}_k}^T)/\mathbf{S}_{\tilde{\mathbf{y}}_k}, \tag{7.145}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-),$$

$$\mathbf{U} = \mathcal{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k}, \tag{7.146}$$

$$\mathbf{S}_k = \mathrm{cholupdate}\{\mathbf{S}_k^-, \mathbf{U}, -1\}, \tag{7.147}$$

opposed to $\mathcal{O}(L^3)$, is possible by taking advantage of the *linear* state transition function. Specifically, the time update of the state covariance is given simply by $\mathbf{P}_{\mathbf{w}_k}^- = \mathbf{P}_{\mathbf{w}_{k-1}} + \mathbf{R}_{k-1}^\mathbf{r}$ (see Section 7.4 for a discussion on selecting $\mathbf{R}_{k-1}^\mathbf{r}$). In the square-root filters $\mathbf{S}_{\mathbf{w}_k}$ may thus be updated directly in Eq. (7.150) using one of two options: (1) $\mathbf{S}_{\mathbf{w}_k}^- = \lambda_{RLS}^{-1/2}\mathbf{S}_{\mathbf{w}_{k-1}}$, correspond-

[14]Alternatively, *redraw* a new set of sigma points that incorporate the additive process noise, i.e., $\mathcal{X}_{k|k-1} = [\hat{\mathbf{x}}_k^- \quad \hat{\mathbf{x}}_k^- + \gamma\mathbf{S}_k^- \quad \hat{\mathbf{x}}_k^- - \gamma\mathbf{S}_k^-]$.

Table 7.10 Square-root UKF for parameter estimation

Initialize with

$$\hat{\mathbf{w}}_0 = E[\mathbf{w}], \qquad \mathbf{S}_{\mathbf{w}_0} = \mathrm{chol}\{E[(\mathbf{w} - \hat{\mathbf{w}}_0)(\mathbf{w} - \hat{\mathbf{w}}_0)^T]\}. \tag{7.148}$$

For $k \in \{1, \ldots, \infty\}$,
The time update and sigma point calculation are given by

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1}, \tag{7.149}$$

$$\mathbf{S}_{\mathbf{w}_k}^- = \lambda_{RLS}^{-1/2}\mathbf{S}_{\mathbf{w}_{k-1}} \quad \text{or} \quad \mathbf{S}_{\mathbf{w}_k}^- = \mathbf{S}_{\mathbf{w}_{k-1}} + \mathbf{D}_{\mathbf{r}_{k-1}}, \tag{7.150}$$

$$\mathcal{W}_{k|k-1} = [\hat{\mathbf{w}}_k^- \quad \hat{\mathbf{w}}_k^- + \gamma\mathbf{S}_{\mathbf{w}_k}^- \quad \hat{\mathbf{w}}_k^- - \gamma\mathbf{S}_{\mathbf{w}_k}^-], \tag{7.151}$$

$$\mathcal{D}_{k|k-1} = \mathbf{G}(\mathbf{x}_k, \mathcal{W}_{k|k-1}), \tag{7.152}$$

$$\hat{\mathbf{d}}_k = \sum_{i=0}^{2L} W_i^{(m)}\mathcal{D}_{i,k|k-1}, \tag{7.153}$$

and the measurement-update equations are

$$\mathbf{S}_{\mathbf{d}_k} = \mathrm{qr}\left\{\left[\sqrt{W_1^{(c)}}(\mathcal{D}_{1:2L,k} - \hat{\mathbf{d}}_k) \quad \sqrt{\mathbf{R}^{\mathrm{e}}}\right]\right\}, \tag{7.154}$$

$$\mathbf{S}_{\mathbf{d}_k} = \mathrm{cholupdate}\{\mathbf{S}_{\mathbf{d}_k}, \mathcal{D}_{0,k} - \hat{\mathbf{d}}_k, W_0^{(c)}\}, \tag{7.155}$$

$$\mathbf{P}_{\mathbf{w}_k\mathbf{d}_k} = \sum_{i=0}^{2L} W_i^{(c)}(\mathcal{W}_{i,k|k-1} - \hat{\mathbf{w}}_k^-)(\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k)^T, \tag{7.156}$$

$$\mathcal{K}_k = (\mathbf{P}_{\mathbf{w}_k\mathbf{d}_k}/\mathbf{S}_{\mathbf{d}_k}^T)/\mathbf{S}_{\mathbf{d}_k}, \tag{7.157}$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathcal{K}_k(\mathbf{d}_k - \hat{\mathbf{d}}_k), \tag{7.156}$$

$$\mathbf{U} = \mathcal{K}_k\mathbf{S}_{\mathbf{d}_k}, \tag{7.158}$$

$$\mathbf{S}_{\mathbf{w}_k} = \mathrm{cholupdate}\{\mathbf{S}_{\mathbf{w}_k}^-, \mathbf{U}, -1\}, \tag{7.159}$$

where

$$\mathbf{D}_{\mathbf{r}_{k-1}} = -\mathrm{Diag}\{\mathbf{S}_{\mathbf{w}_{k-1}}\} + \sqrt{\mathrm{Diag}\{\mathbf{S}_{\mathbf{w}_{k-1}}\}^2 + \mathrm{Diag}\{\mathbf{R}_{k-1}^{\mathbf{r}}\}}.$$

ing to an exponential weighting on past data; (2) $\mathbf{S}_{\mathbf{w}_k}^- = \mathbf{S}_{\mathbf{w}_{k-1}} + \mathbf{D}_{\mathbf{r}_{k-1}}$, where the diagonal matrix $\mathbf{D}_{\mathbf{r}_{k-1}}$, is chosen to approximate the effects of annealing a diagonal process noise covariance $\mathbf{R}_k^{\mathbf{r}}$.[15] Both options avoid the costly $\mathcal{O}(L^3)$ QR and Cholesky-based updates necessary in the state-estimation filter.

[15]This update ensures that the main diagonal of $\mathbf{P}_{\mathbf{w}_k}^-$ is exact. However, additional off-diagonal cross-terms $\mathbf{S}_{\mathbf{w}_{k-1}}\mathbf{D}_{\mathbf{r}_{k-1}}^T + \mathbf{D}_{\mathbf{r}_{k-1}}\mathbf{S}_{\mathbf{w}_{k-1}}^T$ are also introduced (though the effect appears negligible).

REFERENCES

[1] S.J. Julier, J.K. Uhlmann, and H. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *Proceedings of the American Control Conference*, 1995, pp. 1628–1632.

[2] S.J. Julier and J.K. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Technical Report, RRG, Department of Engineering Science, University of Oxford, November 1996. http://www.robots.ox.ac.uk/siju/work/publications/letter_size/Unscented.zip.

[3] S.J. Julier and J.K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls, 1997.*

[4] E.A. Wan, R. van der Merwe, and A.T. Nelson, "Dual estimation and the unscented transformation," in S.A. Solla, T.K. Leen, and K.-R. Müller, Eds. *Advances in Neural Information Processing Systems 12*, Cambridge, MA: MIT Press, 2000, pp. 666–672.

[5] E.A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation, in *Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control (AS-SPCC), IEEE, Lake Louise, Alberta, Canada, October 2000.*

[6] R. van der Merwe, J.F.G. de Freitas, D. Doucet, and E.A. Wan, "The unscented particle filter," Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, August 2000.

[7] R. van der Merwe and E.A. Wan, "Efficient derivative-free Kalman filters for online learning," in *Proceedings of European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, April 2001.*

[8] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman filter," in *Advances in Neural Information Processing Systems 1.* San Mateo, CA: Morgan Kauffman, 1989, pp. 133–140.

[9] G.V. Puskorius and L.A. Feldkamp, "Decoupled extended Kalman filter training of feedforward layered networks," in *Proceedings of IJCNN*, Vol. 1, International Joint Conference on Neural Networks, 1991, pp. 771–777.

[10] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME, Ser. D, Journal of Basic Engineering*, **82**, 35–45 (1960).

[11] A. Jazwinsky, *Stochastic Processes and Filtering Theory.* New York: Academic Press, 1970.

[12] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Transactions on Automatic Control*, **45**, 910–927 (2000).

[13] M. Nørgaard, N.K. Poulsen, and O. Ravn, "Advances in derivative-free state estimation for nonlinear systems," Technical Report IMM-REP-1998-15,

Department of Mathematical Modelling/Department of Automation, Technical University of Denmark, Lyngby, April 2000.

[14] J.R. Cloutier, C.N. D'Souza, and C.P. Mracek, "Nonlinear regulation and nonlinear H-infinity controls via the state-dependent Riccati equation technique: Part 1, Theory," in *Proceedings of the International Conference on Nonlinear Problems in Aviation and Aerospace, Daytona Beach, FL, May 1996*.

[15] M. Mackey and L. Glass, "Oscillation and chaos in a physiological control system," *Science*, **197**, 287–289 1977.

[16] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modelling," Technical Report LAUR 872662, Los Alamos National Laboratory, 1987.

[17] R.H. Shumway and D.S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Time Series Analysis*, **3**, 253–264 (1982).

[18] Z. Ghahramani and S.T. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," in M.J. Kearns, S.A. Solla, and D.A. Cohn, Eds., *Advances in Neural Information Processing Systems 11: Proceedings of the 1998 Conference*. Cambridge, MA: MIT Press, 1999.

[19] F.L. Lewis, *Optimal Estimation*. New York: Wiley, 1986.

[20] A.H. Sayed and T. Kailath, "A state-space approach to adaptive RLS filtering," *IEEE Signal Processing Magazine*, pp. 18–60 (July 1994).

[21] S. Haykin. *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1996.

[22] A.T. Nelson, "Nonlinear estimation and modeling of noisy time-series by dual Kalman filtering methods," PhD Thesis, Oregon Graduate Institute, 2000.

[23] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.

[24] D.J.C. MacKay, http://wol.ra.phy.cam.ac.uk/mackay/sourcedata. html.www.

[25] D.J.C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Computation*, **4**, 448–472 (1992).

[26] K. Ikeda, "Multiple-valued stationary state and its instability of light by a ring cavity system," *Optics Communications* **30**, 257–261 (1979).

[27] H.H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *Computer Journal*, **3**, 175–184 (1960).

[28] G.V. Puskorius and L.A. Feldkamp, "Extensions and enhancements of decoupled extended Kalman filter training," in *Proceedings of ICNN*, Vol. 3, International Conference on Neural Networks 1997, pp. 1879–1883.

[29] E.A. Wan and A.T. Nelson, "Neural dual extended Kalman filtering: Applications in speech enhancement and monaural blind signal separation," in *Proceedings of Neural Networks for Signal Processing Workshop, IEEE*, 1997.

[30] M.B. Matthews, "A state-space approach to adaptive nonlinear filtering using recurrent neural networks," in *Proceedings of IASTED International Symposium on Artificial Intelligence Application and Neural Networks, 1990*, pp. 197–200.

[31] R.W. Brumbaugh, "An Aircraft Model for the AIAA Controls Design Challenge," PRC Inc., Edwards, CA.

[32] J.P. Dutton, "Development of a Nonlinear Simulation for the McDonnell Douglas F-15 Eagle with a Longitudinal TECS Control Law," Master Thesis, Department of Aeronautics and Astronautics, University of Washington, 1994.

[33] A Doucet, "On sequential simulation-based methods for Bayesian filtering," Technical Report CUED/F-INFENG/TR 310, Cambridge University Engineering Department, 1998.

[34] A. Doucet, J.F.G. de Freitas, and N.J. Gordon, "Introduction to sequential Monte Carlo methods," in A. Doucet, J.F.G. de Freitas, and N.J. Gordon, Eds. *Sequential Monte Carlo Methods in Practice*. Berlin: Springer-Verlag, 2000.

[35] N.J. Gordon, D.J. Salmond, and A.F.M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings, Part F*, **140**, 107–113 (1993).

[36] B. Efron, *The Bootstrap Jacknife and other Resampling Plans*. Philadelphia: SIAM, 1982.

[37] D.B. Rubin, "Using the SIR algorithm to simulate posterior distributions," in J.M. Bernardo, M.H. DeGroot, D.V. Lindley, and A.F.M. Smith, Eds., *Bayesian Statistics 3*. Oxford University Press, 1988, pp. 395–402.

[38] A.F.M. Smith and A.E. Gelfand, "Bayesian statistics without tears: a sampling–resampling perspective," *American Statistician*, **46**, 84–88, 1992.

[39] T. Higuchi, "Monte Carlo filter using the genetic algorithm operators," *Journal of Statistical Computation and Simulation*, **59**, 1–23 (1997).

[40] A. Kong, J.S. Liu, and W.H. Wong, "Sequential imputations and Bayesian missing data problems," *Journal of the American Statistical Association*, **89**, 278–288 (1994).

[41] J.S. Liu and R. Chen, "Blind deconvolution via sequential imputations," *Journal of the American Statistical Association*, **90**, 567–576 (1995).

[42] J.S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, **93**. 1032–1044 (1998).

[43] V.S. Zaritskii, V.V. Svetnik, and L.I. Shimelevich, "Monte-Carlo techniques in problems of optimal information processing," *Automation and Remote Control*, **36**, 2015–2022 (1975).

[44] D. Avitzour, "A stochastic simulation Bayesian approach to multitarget tracking," *IEE Proceedings on Radar, Sonar and Navigation*, **142**, 41–44 (1995).

[45] E.R. Beadle and P.M. Djurić, "A fast weighted Bayesian bootstrap filter for nonlinear model state estimation," *IEEE Transactions on Aerospace and Elecytronic Systems*, **33**, 338–343 (1997).

[46] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *Proceedings of European Conference on Computer Vision*, *Cambridge, UK*, 1996, pp. 343–356.

[47] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space model," *Journal of Computational and Graphical Statistics*, **5**, 1–25 (1996).

[48] J.F.G. de Freitas, "Bayesian Methods for Neural Networks," PhD Thesis, Cambridge University Engineering Department, 1999.

[49] J.C. Hull, *Options, Futures, and Other Derivatives*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1997.

[50] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, **81**, 637–659 (1973).

[51] M. Niranjan, "Sequential tracking in pricing financial options using model based and neural network approaches," in M.C. Mozer, M.I. Jordan, and T. Petsche, Eds. *Advances in Neural Information Processing Systems* **8**, 1996, 960–966.

[52] J.F.G. de Freitas, M. Niranjan, A.H. Gee, and A. Doucet, "Sequential Monte Carlo methods to train neural network models," *Neural Computation*, **12**, 955–993 (2000).

[53] S.J. Julier, "The scaled unscented transformation." In preparation.

[54] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

[55] R. van der Merwe and E.A. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, UT, May 2001.*