THE SQUARE-ROOT UNSCENTED KALMAN FILTER FOR STATE AND PARAMETER-ESTIMATION

Rudolph van der Merwe and Eric A. Wan

Oregon Graduate Institute of Science and Technology
20000 NW Walker Road, Beaverton, Oregon 97006, USA
{rvdmerwe,ericwan}@ece.ogi.edu

ABSTRACT

Over the last 20-30 years, the *extended Kalman filter* (EKF) has become the algorithm of choice in numerous nonlinear estimation and machine learning applications. These include estimating the state of a nonlinear dynamic system as well estimating parameters for nonlinear system identification (*e.g.,* learning the weights of a neural network). The EKF applies the standard linear Kalman filter methodology to a linearization of the true nonlinear system. This approach is sub-optimal, and can easily lead to divergence. Julier et al. [1] proposed the *unscented Kalman filter* (UKF) as a derivative-free alternative to the extended Kalman filter in the framework of state-estimation. This was extended to parameter-estimation by Wan and van der Merwe [2, 3]. The UKF consistently outperforms the EKF in terms of prediction and estimation error, at an equal computational complexity of $\mathcal{O}(L^3)$[1] for general state-space problems. When the EKF is applied to parameter-estimation, the special form of the state-space equations allows for an $\mathcal{O}(L^2)$ implementation. This paper introduces the *square-root unscented Kalman filter* (SR-UKF) which is also $\mathcal{O}(L^3)$ for general state-estimation and $\mathcal{O}(L^2)$ for parameter estimation (note the original formulation of the UKF for parameter-estimation was $\mathcal{O}(L^3)$). In addition, the square-root forms have the added benefit of numerical stability and guaranteed positive semi-definiteness of the state covariances.

1. INTRODUCTION

The EKF has been applied extensively to the field of nonlinear estimation for both *state-estimation* and *parameter-estimation*. The basic framework for the EKF (and the UKF) involves estimation of the state of a discrete-time nonlinear dynamic system,

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

$$\mathbf{y}_k = \mathbf{H}(\mathbf{x}_k) + \mathbf{n}_k, \quad (2)$$

where $\mathbf{x}_k$ represent the unobserved state of the system, $\mathbf{u}_k$ is a known exogenous 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 $\mathbf{n}_k$. The EKF involves the recursive estimation of the mean and covariance of the state under a Gaussian assumption.

In contrast, parameter-estimation, sometimes referred to as system identification, involves determining a nonlinear mapping $\mathbf{y}_k = \mathbf{G}(\mathbf{x}_k, \mathbf{w})$, 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 $\mathbf{w}$. 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 and Quasi-Newton methods), parameters can be efficiently estimated on-line by writing a new state-space representation

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{r}_k \quad (3)$$

$$\mathbf{d}_k = \mathbf{G}(\mathbf{x}_k, \mathbf{w}_k) + \mathbf{e}_k, \quad (4)$$

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). 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 [4].

2. THE UNSCENTED KALMAN FILTER

The inherent flaws of the EKF are due to its linearization approach for calculating the mean and covariance of a random variable which undergoes a nonlinear transformation. As shown in shown in [1, 2, 3], the UKF addresses these flaws by utilizing a deterministic "sampling" approach to calculate mean and covariance terms. Essentially, $2L+1$, *sigma* points (L is the state dimension), are chosen based on a square-root decomposition of the prior covariance. These sigma points are propagated through the true nonlinearity, without approximation, and then a weighted mean and covariance is taken. A simple illustration of the approach is shown in Figure 1 for a 2-dimensional system: the left plot shows the true mean and covariance propagation using Monte-Carlo sampling; the center plots show the results using a linearization approach as would be done in the EKF; the right plots show the performance of the new "sampling" approach (note only 5 sigma points are required). This approach results in approximations that are accurate to the third order (Taylor series expansion) for Gaussian inputs for all nonlinearities. For non-Gaussian inputs, approximations are accurate to at least the second-order [1]. In contrast, the linearization approach of the EKF results only in first order accuracy.

The full UKF involves the recursive application of this "sampling" approach to the state-space equations. The standard UKF implementation is given in Algorithm 2.1 for state-estimation, and uses the following variable definitions: $\{W_i\}$ is a set of scalar weights ($W_0^{(m)} = \lambda/(L+\lambda)$, $W_0^{(c)} = \lambda/(L+\lambda) + (1-\alpha^2+\beta)$

This work was sponsored in part by NSF under grant grant IRI-9712346, ECS-0083106, and DARPA under grant F33615-98-C-3516.

[1] L is the dimension of the state variable.

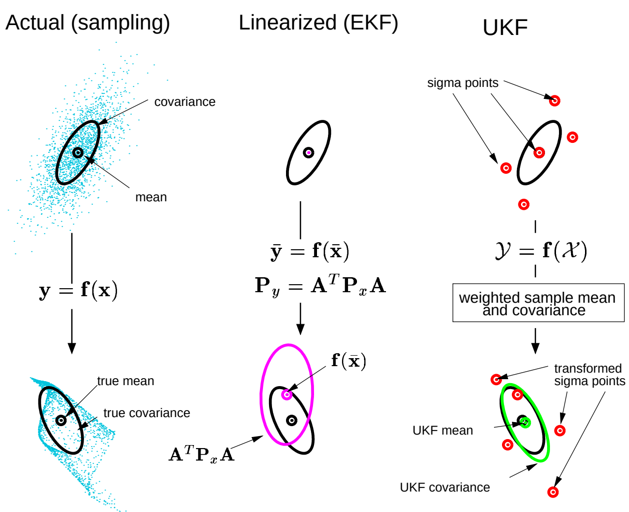

Figure 1: Example of mean and covariance propagation. a) actual, b) first-order linearization (EKF), c) new "sampling" approach (UKF).

, $W_i^{(m)} = W_i^{(c)} = 1/\{2(L+\lambda)\}$ $i = 1, \dots, 2L$). $\lambda = L(\alpha^2 - 1)$ and $\eta = \sqrt{(L+\lambda)}$ are scaling parameters. The constant α determines the spread of the sigma points around $\hat{\mathbf{x}}$ and is usually set to $1e-4 \leq \alpha \leq 1$. β is used to incorporate prior knowledge of the distribution of $\mathbf{x}$ (for Gaussian distributions, $\beta = 2$ is optimal). Also note that we define the linear algebra operation of adding a column vector to a matrix, *i.e.* $\mathbf{A} \pm \mathbf{u}$ as the addition of the vector to each column of the matrix. The superior performance of the UKF over the EKF has been demonstrated in a number of applications [1, 2, 3]. Furthermore, unlike the EKF, no explicit derivatives (*i.e.*, Jacobians or Hessians) need to be calculated.

3. EFFICIENT SQUARE-ROOT IMPLEMENTATION

The most computationally expensive operation in the UKF corresponds to calculating the new set of sigma points at each time update. This requires taking a matrix square-root of the state covariance matrix[2], $\mathbf{P} \in \mathbb{R}^{L\times L}$, given by $\mathbf{S}\mathbf{S}^T = \mathbf{P}$. An efficient implementation using a Cholesky factorization requires in general $\mathcal{O}(L^3/6)$ computations [5]. While the square-root of $\mathbf{P}$ is an integral part of the UKF, it is still the full covariance $\mathbf{P}$ which is recursively updated. In the SR-UKF implementation, $\mathbf{S}$ will be propagated directly, avoiding the need to refactorize at each time step. The algorithm will in general still be $\mathcal{O}(L^3)$, but with improved numerical properties similar to those of standard square-root Kalman filters [6]. Furthermore, for the special state-space formulation of parameter-estimation, an $\mathcal{O}(L^2)$ implementation becomes possible.

The square-root form of the UKF makes use of three powerful linear algebra techniques[3], *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, $\mathbf{A}^T = \mathbf{Q}\mathbf{R}$, 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 $\mathbf{P} = \mathbf{A}\mathbf{A}^T$, *i.e.*, $\tilde{\mathbf{R}} = \mathbf{S}^T$, such that $\tilde{\mathbf{R}}^T\tilde{\mathbf{R}} = \mathbf{A}\mathbf{A}^T$. 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 $\mathbf{P} = \mathbf{A}\mathbf{A}^T$ is $\mathcal{O}(L^3/6)$ plus $\mathcal{O}(NL^2)$ to form $\mathbf{A}\mathbf{A}^T$.
- *Cholesky factor updating.* If $\mathbf{S}$ is the original Cholesky factor of $\mathbf{P} = \mathbf{A}\mathbf{A}^T$, then the Cholesky factor of the rank-1 update (or downdate) $\mathbf{P} \pm \sqrt{\nu}\mathbf{u}\mathbf{u}^T$ is denoted as $\mathbf{S} =$ cholupdate$\{\mathbf{S}, \mathbf{u}, \pm\nu\}$. 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.
- *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 specification of the new square-root filters is given in Algorithm 3.1 for state-estimation and 3.2 for paramater-

[2] For notational clarity, the time index k has been omitted.
[3] See [5] for theoretical and implementation details.

Initialize with:

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0] \quad \mathbf{P}_0 = \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] \tag{5}$$

For $k \in \{1, \dots, \infty\}$,

Calculate sigma points:

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\mathbf{x}}_{k-1} \quad \hat{\mathbf{x}}_{k-1} + \eta\sqrt{\mathbf{P}_{k-1}} \quad \hat{\mathbf{x}}_{k-1} - \eta\sqrt{\mathbf{P}_{k-1}}\right] \tag{6}$$

Time update:

$$\boldsymbol{\mathcal{X}}_{k|k-1} = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}, \mathbf{u}_{k-1}] \tag{7}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1} \tag{8}$$

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

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}] \tag{9}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1} \tag{10}$$

Measurement update equations:

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

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T \tag{11}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}^{-1} \tag{12}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \tag{13}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}_k\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}\mathcal{K}_k^T \tag{14}$$

where $\mathbf{R}^{\mathbf{v}}$=process noise cov., $\mathbf{R}^{\mathbf{n}}$=measurement noise cov.

Algorithm 2.1: Standard UKF algorithm.

estimation. Below we describe the key parts of the square-root algorithms, and how they contrast with the standard implementations.

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 (Eqn. 16). However, the propagted and updated Cholesky factor is then used in subsequent iterations to directly form the sigma points. In Eqn. 20 the *time-update* of the Cholesky factor, $\mathbf{S}^-$, is calculated using a QR decompostion 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 Eqn. 21 is necessary since the the zero'th weight, $W_0^{(c)}$, may be negative. These two steps replace the *time-update* of $\mathbf{P}^-$ in Eqn. 9, 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 Eqns. 24 and 25. This step is $\mathcal{O}(LM^2)$, where M is the observation dimension. In contrast to the way the Kalman gain is calculated in the standard UKF (see Eqn. 12), we now use two nested inverse (or *least squares*) solutions to the following expansion of Eqn. 12, $\mathcal{K}_k(\mathbf{S}_{\tilde{\mathbf{y}}_k}\mathbf{S}_{\tilde{\mathbf{y}}_k}^T) = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}$. 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 Eqn. 29 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 Eqn. 14, and is also $\mathcal{O}(LM^2)$.

Square-Root Parameter-Estimation: The parameter-estimation algorithm follows a similar framework as that of the state-estimation square-root UKF. However, an $\mathcal{O}(ML^2)$ algorithm, as 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 $\mathbf{P}_{\mathbf{w}_k}^- = \mathbf{P}_{\mathbf{w}_{k-1}} + \mathbf{R}_{k-1}^{\mathbf{r}}$. Now, if we apply an exponential weighting on past data[4], the process noise covariance is given by $\mathbf{R}_k^{\mathbf{r}} = (\gamma^{-1}-1)\mathbf{P}_{\mathbf{w}_k}$, and the time update of the state covariance becomes,

$$\mathbf{P}_{\mathbf{w}_k}^- = \mathbf{P}_{\mathbf{w}_{k-1}} + (\gamma^{-1}-1)\mathbf{P}_{\mathbf{w}_{k-1}} = \gamma^{-1}\mathbf{P}_{\mathbf{w}_{k-1}}. \tag{15}$$

This translates readily into the factored form, $\mathbf{S}_{\mathbf{w}_k}^- = \gamma^{-1/2}\mathbf{S}_{\mathbf{w}_{k-1}}$ (see Eqn. 32), and avoids the costly $\mathcal{O}(L^3)$ QR and Cholesky based updates necessary in the state-estimation filter.

Initialize with:

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0] \quad \mathbf{S}_0 = \text{chol}\left\{\mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]\right\} \tag{16}$$

For $k \in \{1, \dots, \infty\}$,

Sigma point calculation and time update:

$$\boldsymbol{\mathcal{X}}_{k-1} = [\hat{\mathbf{x}}_{k-1} \quad \hat{\mathbf{x}}_{k-1} + \eta\mathbf{S}_k \quad \hat{\mathbf{x}}_{k-1} - \eta\mathbf{S}_k] \tag{17}$$

$$\boldsymbol{\mathcal{X}}_{k|k-1} = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}, \mathbf{u}_{k-1}] \tag{18}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)}\mathcal{X}_{i,k|k-1} \tag{19}$$

$$\mathbf{S}_k^- = \text{qr}\left\{\left[\sqrt{W_1^{(c)}}\left(\boldsymbol{\mathcal{X}}_{1:2L,k|k-1} - \hat{\mathbf{x}}_k^-\right) \quad \sqrt{\mathbf{R}^{\mathbf{v}}}\right]\right\} \tag{20}$$

$$\mathbf{S}_k^- = \text{cholupdate}\left\{\mathbf{S}_k^-\ ,\ \mathcal{X}_{0,k} - \hat{\mathbf{x}}_k^-\ ,\ W_0^{(c)}\right\} \tag{21}$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}] \tag{22}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)}\mathcal{Y}_{i,k|k-1} \tag{23}$$

Measurement update equations:

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \text{qr}\left\{\left[\sqrt{W_1^{(c)}}[\boldsymbol{\mathcal{Y}}_{1:2L,k} - \hat{\mathbf{y}}_k] \quad \sqrt{\mathbf{R}_k^{\mathbf{n}}}\right]\right\} \tag{24}$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \text{cholupdate}\left\{\mathbf{S}_{\tilde{\mathbf{y}}_k}\ ,\ \mathcal{Y}_{0,k} - \hat{\mathbf{y}}_k\ ,\ W_0^{(c)}\right\} \tag{25}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)}[\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T \tag{26}$$

$$\mathcal{K}_k = (\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}/\mathbf{S}_{\tilde{\mathbf{y}}_k}^T)/\mathbf{S}_{\tilde{\mathbf{y}}_k} \tag{27}$$

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

$$\mathbf{U} = \mathcal{K}_k\mathbf{S}_{\tilde{\mathbf{y}}_k} \tag{28}$$

$$\mathbf{S}_k = \text{cholupdate}\left\{\mathbf{S}_k^-\ ,\ \mathbf{U}\ ,\ \text{-1}\right\} \tag{29}$$

where $\mathbf{R}^{\mathbf{v}}$=process noise cov., $\mathbf{R}^{\mathbf{n}}$=measurement noise cov.

Algorithm 3.1: Square-Root UKF for state-estimation.

4. EXPERIMENTAL RESULTS

The improvement in error performance of the UKF over that of the EKF for both state and parameter-estimation is well documented [1, 2, 3]. The focus of this section will be to simply verify the equivalent error performance of the UKF and SR-UKF, and show the reduction in computational cost achieved by the SR-UKF for parameter-estimation. Figure 2 shows the superior performance of UKF and SR-UKF compared to that of the EKF on estimating the Mackey-Glass-30 chaotic time series corrupted by additive white noise (3dB SNR). The error performance of the SR-UKF and UKF are indistinguishable and are both superior to the EKF. The computational complexity of all three filters are of the same order but the SR-UKF is about 20% faster than the UKF and about 10% faster than the EKF.

The next experiment shows the reduction in computational cost achieved by the square-root unscented Kalman filters and how that compares to the computational complexity of the EKF for parameter-estimation. For this experiment, we use an EKF, UKF and SR-UKF to train a 2-12-2 MLP neural network on the well known *Mackay-Robot-Arm*[5] benchmark problem of mapping the joint angles of a robot arm to the Cartesian coordinates of the hand. The learning curves (mean square error (MSE) vs. learning epoch) of the different filters are shown in Figure 3. Figure 4 shows how the computational complexity of the different filters scale as a function of the number of parameters (weights in neural network). While the standard UKF is $\mathcal{O}(L^3)$, both the EKF and SR-UKF are $\mathcal{O}(L^2)$.

[4] This is identical to the approach used in weighted recursive least squares (W-RLS). γ is a scalar weighting factor chosen to be slightly less than 1, *i.e.* $\gamma = 0.9995$.

[5] http://wol.ra.phy.cam.ac.uk/mackay

Initialize with:

$$\hat{\mathbf{w}}_0 = E[\mathbf{w}] \quad \mathbf{S}_{\mathbf{w}_0} = \text{chol}\left\{E[(\mathbf{w}-\hat{\mathbf{w}}_0)(\mathbf{w}-\hat{\mathbf{w}}_0)^T]\right\} \tag{30}$$

For $k \in \{1, \dots, \infty\}$,

Time update and sigma point calculation:

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1} \tag{31}$$

$$\mathbf{S}_{\mathbf{w}_k}^- = \gamma^{-1/2}\mathbf{S}_{\mathbf{w}_{k-1}} \tag{32}$$

$$\boldsymbol{\mathcal{W}}_{k|k-1} = \left[\hat{\mathbf{w}}_k^- \quad \hat{\mathbf{w}}_k^- + \eta\mathbf{S}_{\mathbf{w}_k}^- \quad \hat{\mathbf{w}}_k^- - \eta\mathbf{S}_{\mathbf{w}_k}^-\right] \tag{33}$$

$$\boldsymbol{\mathcal{D}}_{k|k-1} = \mathbf{G}[\mathbf{x}_k, \boldsymbol{\mathcal{W}}_{k|k-1}] \tag{34}$$

$$\hat{\mathbf{d}}_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{D}_{i,k|k-1} \tag{35}$$

Measurement update equations:

$$\mathbf{S}_{\mathbf{d}_k} = \text{qr}\left\{\left[\sqrt{W_1^{(c)}}\left[\boldsymbol{\mathcal{D}}_{1:2L,k} - \hat{\mathbf{d}}_k\right] \quad \sqrt{\mathbf{R^e}}\right]\right\} \tag{36}$$

$$\mathbf{S}_{\mathbf{d}_k} = \text{cholupdate}\left\{\mathbf{S}_{\mathbf{d}_k}\ ,\ \mathcal{D}_{0,k} - \hat{\mathbf{d}}_k\ ,\ W_0^{(c)}\right\} \tag{37}$$

$$\mathbf{P}_{\mathbf{w}_k\mathbf{d}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{W}_{i,k|k-1} - \hat{\mathbf{w}}_k^-][\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k]^T \tag{38}$$

$$\mathcal{K}_k = (\mathbf{P}_{\mathbf{w}_k\mathbf{d}_k}/\mathbf{S}_{\mathbf{d}_k}^T)/\mathbf{S}_{\mathbf{d}_k} \tag{39}$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathcal{K}_k(\mathbf{d}_k - \hat{\mathbf{d}}_k) \tag{40}$$

$$\mathbf{U} = \mathcal{K}_k\mathbf{S}_{\mathbf{d}_k} \tag{41}$$

$$\mathbf{S}_{\mathbf{w}_k} = \text{cholupdate}\left\{\mathbf{S}_{\mathbf{w}_k}^-\ ,\ \mathbf{U}\ ,\ \text{-1}\right\} \tag{42}$$

where $\mathbf{R^e}$=measurement noise cov (this can be set to an arbitrary value, *e.g.*, $.5\mathbf{I}$.)

Algorithm 3.2: Square-Root UKF for parameter-estimation.

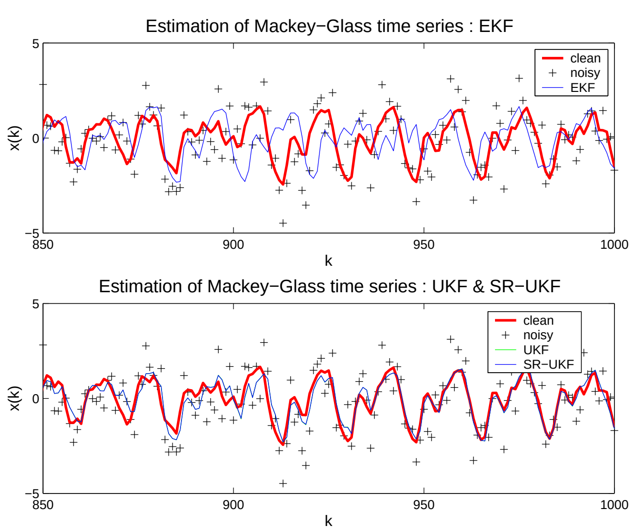

Figure 2: Estimation of the Mackey-Glass chaotic time-series with the EKF, UKF and SR-UKF.

5. CONCLUSIONS

The UKF consistently performs better than or equal to the well known EKF, with the added benefit of ease of implementation in that no analytical derivatives (Jacobians or Hessians) need to be calculated. For state-estimation, the UKF and EKF have equal complexity and are in general $\mathcal{O}(L^3)$. In this paper, we introduced square-root forms of the UKF. The square-root UKF has better numerical properties and guarantees positive semi-definiteness of the underlying state covariance. In addition, for parameter-estimation an efficient $\mathcal{O}(L^2)$ implementation is possible for the square-root form, which is again of the same complexity as efficient EKF parameter-estimation implementations. In this light, the SR-UKF is the logical replacement for the EKF in all state and parameter-estimation applications.

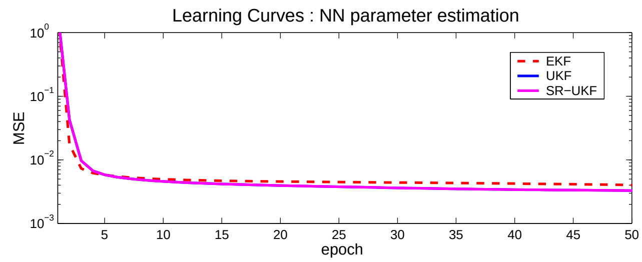

Figure 3: Learning curves for Mackay-Robot-Arm neural network parameter-estimation problem.

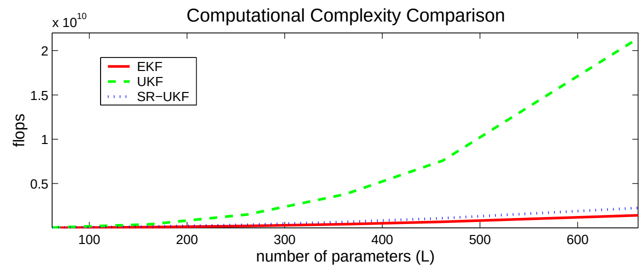

Figure 4: Computational complexity (flops/epoch) of EKF, UKF and SR-UKF for parameter-estimation (Mackay-Robot-Arm problem).

6. REFERENCES

[1] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," in *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls.*, 1997.

[2] E. Wan, R. van der Merwe, and A. T. Nelson, "Dual Estimation and the Unscented Transformation," in *Neural Information Processing Systems 12*. 2000, pp. 666–672, MIT Press.

[3] E. A. Wan and R. van der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *Proc. of IEEE Symposium 2000 (AS-SPCC)*, Lake Louise, Alberta, Canada, Oct. 2000.

[4] G.V. Puskorius and L.A. Feldkamp, "Decoupled Extended Kalman Filter Training of Feedforward Layered Networks," in *IJCNN*, 1991, vol. 1, pp. 771–777.

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

[6] A. H. Sayed and T. Kailath, "A State-Space Approach to Adaptive RLS Filtering," *IEEE Sig. Proc. Mag.*, pp. 18–60, July 1994.