The J-Orthogonal Square-Root MATLAB-Based Continuous-Discrete Unscented Kalman Filtering Method [⋆]

Maria V. Kulikova [∗], **Gennady Yu. Kulikov** [∗]

[∗] *CEMAT, Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001 LISBOA, Portugal (emails: maria.kulikova@ist.utl.pt, gkulikov@math.ist.utl.pt).*

Abstract: The paper suggests a general solution to the square-rooting problem existed for the Unscented Kalman Filter (UKF) since its appearance in the late 1990s. As properly noted in engineering literature, the previously suggested Cholesky-based UKF implementations are, in fact, the 'pseudo' square-root versions. Their key feature is the utilization of one-rank Cholesky update procedure required at each filtering step because of the possibly negative sigma points' weights. In a finite precision arithmetic, the resulting downdated matrix might be not a positive definite matrix. This yields a failure of the UKF estimator in practice. We resolve this problem by suggesting a novel square-root approach based on the J-orthogonal matrix utilization for updating the required Cholesky factors. Additionally, we explain how the MATLAB language with built-in numerical integration schemes developed for solving ordinary differential equations can be easily and effectively used for accurate calculations when implementing the continuous-discrete UKF time update stage.

Keywords: Unscented Kalman filter, Cholesky decomposition, square-root methods.

1. PROBLEM STATEMENT

Consider continuous-discrete stochastic system of the form

$$dx(t) = f\big(t, x(t)\big)dt + Gd\beta(t), \quad t > 0, \qquad (1)$$

$$z_k = h(k, x(t_k)) + v_k, \quad k = 1, 2, \dots \qquad (2)$$

where $x(t) \in \mathbb{R}^n$ is the unknown state vector to be estimated and the vector-function $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is the time-variant drift function. The process uncertainty is modelled by the additive noise term where $G \in \mathbb{R}^{n\times q}$ is the time-invariant diffusion matrix and $\beta(t)$ is the q-dimensional Brownian motion whose increment $d\beta(t)$ is independent of $x(t)$ and has the covariance $Q\,dt > 0$. Additionally, measurement equation (2) implies a nonlinear relationship $h : \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^m$ between the unknown dynamic state $x(t_k)$ and available data $z_k := z(t_k)$ measured at some discrete-time points t_k. This measured information $z_k \in \mathbb{R}^m$ comes with the sampling rate (sampling period) $\Delta_k = t_k - t_{k-1}$. The measurement noise term v_k is assumed to be white Gaussian noise with zero mean and known covariance $R_k > 0$, i.e. $\mathbf{E}\left\{v_k v_j^T\right\} = R_k\delta_{kj}$ and δ_{kj} is the Kronecker delta function. Finally, for solving the nonlinear Bayesian filtering problem, *a priori* information about the system state distribution should be available as well. The initial state $x(t_0)$ is assumed to be normally distributed with the mean $\bar{x}_0$ and covariance $\Pi_0 > 0$. The initial state and noise processes are all assumed to be statistically independent.

[⋆] The authors acknowledge the financial support of the Portuguese FCT — *Fundação para a Ciência e a Tecnologia*, through the projects UIDB/04621/2020 and UIDP/04621/2020 of CEMAT/IST-ID, Center for Computational and Stochastic Mathematics, Instituto Superior Técnico, University of Lisbon.

Following the *continuous-discrete* implementation framework for implementing any nonlinear KF-like Bayesian filtering method, the related *moment differential equations* (MDEs) for the mean and error covariance matrix should be derived, first. This problem has been solved for the UKF technique by Särkkä (2007). More precisely, the sampled-data UKF estimator is based on a set of $2n+1$ deterministically selected vectors called the sigma vectors; e.g., see a general representation in Wan and Van der Merwe (2001):

$$\xi_0 = \mathbf{0}_n, \quad \xi_i = \sqrt{n+\lambda}e_i, \quad \xi_{n+i} = -\sqrt{n+\lambda}e_i, i = \overline{1,n},$$
$$\text{where} \quad \mathcal{X}_i = \hat{x} + S_P\xi_i, \qquad i = 0,\dots,2n$$

where the vector $\mathbf{0}_n$ is the zero column of size n and e_i denotes the i-th unit coordinate vector in $\mathbb{R}^n$ where n is the dimension of the state vector. The term S_P stands for a square-root factor of the filter error covariance P, i.e. $P = S_P S_P^\top$. It is traditionally defined by using the Cholesky decomposition; e.g., S_P is assumed to be a lower triangular matrix with positive diagonal entries. We stress that Cholesky decomposition is required for generating the sigma vectors at each recursive filtering step. This operation is numerically unstable in a finite precision arithmetic because of the well known KF pitfall recognized since the 1960s; for more details, see the first ill-conditioned estimation scenario discussed in Examples 7.1 and 7.2 in Dyer and McReynolds (1969) and many other studies. To summarize, the roundoff errors may destroy the theoretical properties of error covariance matrix P, which are the symmetric form and positive definiteness. As a result, the Cholesky factorization fails to perform for such matrix as discussed in Kailath et al. (2000); Simon (2006); Grewal and Andrews (2015). Recall, the Cholesky decomposition exists and is unique when the symmetric matrix to be decomposed is positive definite; see Golub and Van Loan (1983). The general way to avoid the discussed numerical instability is to derive the *square-root* filtering methods. The key idea is to perform the factorization only once, i.e. at the initial filtering step. Additionally, the filtering equations should be re-derived in terms of propagating and updating the square-root factors S_P instead of the full matrix P. This approach ensures the symmetric form and positive (semi-) definiteness of the filter error covariance matrix $S_P S_P^\top := P$ although the roundoff errors influence the S_P calculation.

The first attempt to design numerically stable square-root methods for the UKF filtering strategy has been taken in Van der Merwe and Wan (2001). Since that time a wide variety of square-root UKF implementation methods has been derived; e.g., see Wan and Van der Merwe (2001). However, as correctly noted in Arasaratnam and Haykin (2009) all these methods are, in fact, the pseudo-square-root algorithms. All of them are based on the one-rank Cholesky update procedure required at each iterate when the negative UKF weights need to be processed. In a finite precision arithmetic, the resulting downdated matrix may possibly be non-positive definite. Consequently, it results in unexpected interruption of the computations because of a failure of the one-rank Cholesky update procedure. This means that the exact square-root solution for the advanced UKF estimation methodology still does not exist. In this paper, we solve this problem by using the so-called J-orthogonal transformations for updating the Cholesky square-root factors of the filter error covariance matrix. Additionally, we show how the build-in MATLAB functions for solving ordinary differential equations (ODEs) can be effectively used for elegant implementation of the *continuous-discrete* UKF. It is worth noting here that the build-in ODEs solvers' local error controlling mechanization provides an improved estimation accuracy compared to the *discrete-discrete* UKF implementation strategy.

The rest of the paper is organized as follows. In the next section we discuss the *conventional* UKF implementation strategy in a continuous-discrete manner. In Section 3, the new square-root UKF solution is proposed. Results of numerical experiments are presented in Section 4 and illustrate a performance of the novel UKF methods.

2. CONTINUOUS-DISCRETE UNSCENTED KALMAN FILTER

The sample-data UKF estimator depends on three parameters. This makes the estimator to be a flexible and advanced technique. Three pre-defined scalars α, β and κ define some sigma points' weights, which are constant and chosen in such a way that the sample mean and covariance of the state are determined precisely, e.g. Van der Merwe and Wan (2001):

$$w_0^{(m)} = \lambda/(n+\lambda), \qquad w_i^{(m)} = 1/(2n+2\lambda),$$
$$w_0^{(c)} = \lambda/(n+\lambda) + 1 - \alpha^2 + \beta, \qquad w_i^{(c)} = 1/(2n+2\lambda)$$

where $i = \overline{1,2n}$ and $\lambda = \alpha^2(\kappa+n) - n$ regulates the spread of the sigma points around the mean $\hat{x}$ while the secondary scaling parameters β and κ can be used for a further filter's tuning in order to match higher moments of the random variable at hand.

An excellent and the most comprehensive survey of alternative UKF parametrization variants existing in engineering literature can be found in Menegaz et al. (2015). To illustrate a capacity of novel square-root solution suggested in this paper, we focus on the so-called *classical* parametrization where $\alpha = 1$, $\beta = 0$ and $\kappa = 3 - n$; e.g., see Van der Merwe and Wan (2001). This set yields the negative sigma points' weights $w_0^{(m)}$ and $w_0^{(c)}$ when the number of states to be estimated is $n > 3$. Taking into account this scenario as an illustrative example, we explain how to manage the possibly negative weights while deriving the Cholesky factorization-based UKF methods.

To implement the *continuous-discrete* UKF easily and effectively by using the MATLAB language, one needs to represent the required operations in a matrix-vector manner for vectorizing the calculations. For that, the following notation is introduced: the vector $\mathbf{1}_{2n+1}$ is the unitary column of size $2n+1$ and I_{2n+1} is the identity matrix of that size, the symbol $\otimes$ is the Kronecker tensor product (the built-in function `kron` in MATLAB). Additionally, the UKF coefficient vectors and the related weighted matrix are, respectively, defined as follows:

$$w^{(m)} = \left[w_0^{(m)}, \dots, w_{2n}^{(m)}\right]^\top, \; w^{(c)} = \left[w_0^{(c)}, \dots, w_{2n}^{(c)}\right]^\top, \quad (3)$$
$$\mathbb{W} = \left[I_{2n+1} - \mathbf{1}_{2n+1}^\top \otimes w^{(m)}\right] \mathrm{diag}\left\{w_0^{(c)}, \dots, w_{2n}^{(c)}\right\} \times \left[I_{2n+1} - \mathbf{1}_{2n+1}^\top \otimes w^{(m)}\right]^\top \quad (4)$$

where $\mathrm{diag}\left\{w_0^{(c)}, \dots, w_{2n}^{(c)}\right\}$ is the diagonal matrix of size $2n+1$ with the given $w_i^{(c)}$, $i = 0, \dots, 2n$ entries.

Following the *continuous-discrete* implementation framework, one should solve the related UKF MDEs at the filter prediction step. In other words, at each sampling interval $[t_{k-1}, t_k]$ we should propagate the mean and covariance matrix by solving the MDEs derived in Särkkä (2007) and expressed in simple matrix-vector form:

$$\frac{d\hat{x}(t)}{dt} = f\big(t, \mathbb{X}(t)\big) w^{(m)},$$
$$\frac{dP(t)}{dt} = \mathbb{X}(t) \mathbb{W} f^\top\big(t, \mathbb{X}(t)\big) + f\big(t, \mathbb{X}(t)\big) \mathbb{W} \mathbb{X}^\top(t) + GQG^\top$$

where the term $\mathbb{X}(t)$ stands for the matrix collected from the sigma vectors $\mathcal{X}_i(t)$ defined by $\mathcal{X}_i(t) = \hat{x}(t) + S_P(t)\xi_i$. They are located by columns in the discussed matrix, i.e. $\mathbb{X}(t) = \Big[\mathcal{X}_0(t) | \dots | \mathcal{X}_{2n}(t)\Big]$ is of size $n \times (2n+1)$.

An alternative approach is to use the so-called *Sigma-Point Differential Equations* (SPDEs) instead of the UKF MDEs presented above. Following Särkkä (2007), the UKF SPDEs are given as follows:

$$\mathbb{X}_i'(t) = f\big(t, \mathbb{X}(t)\big) w^{(m)} + \sqrt{n+\lambda}\big[\mathbf{0},\; S(t)\Phi(M(t)),\; -S(t)\Phi(M(t))\big]_i \quad (5)$$

where the subscript i refers to the i-th column in the related matrices, $i = 0, \dots 2n$. The notation $\mathbf{0}$ stands for the first column, which is a zero column. The matrix $S(t)$ is the lower triangular Cholesky factor of $P(t)$ and matrix $M(t)$ is calculated by the formula

$$M(t) = S^{-1}(t) \left[\mathbb{X}(t) \mathbb{W} f^\top\big(t, \mathbb{X}(t)\big) + f\big(t, \mathbb{X}(t)\big) \mathbb{W} \mathbb{X}^\top(t) + GQG^\top\right] S^{-\top}(t) \quad (6)$$

with the mapping $\Phi(\cdot)$ that returns a lower triangular matrix defined as follows: (1) split the argument matrix

M as $M = \bar{L} + D + \bar{U}$ where $\bar{L}$ and $\bar{U}$ are, respectively, a strictly lower and upper triangular parts of M, and D is its main diagonal; (2) compute $\Phi(M) = \bar{L} + 0.5D$ for any argument matrix M.

For an accurate and efficient implementation of this time update step, one employs advanced numerical integration schemes designed for solving ordinary differential equations (ODEs). The key benefit of such an implementation strategy is that these include an automatic mechanization for bounding the discretization error committed in line with a user-supplied accuracy request. Previously, we have designed such self-adaptive continuous-discrete UKF methods grounded both in the variable-stepsize nested implicit Runge-Kutta formulas and in the MATLAB ODE solvers in Kulikov and Kulikova (2017) and Kulikov and Kulikova (2018). In this paper, we solve the square-rooting problem existing in these filters and improve their numerical robustness in a finite precision arithmetic underlying computational devices in use. We devise our novel square-rooting method in a general way, i.e. we show how any built-in MATLAB ODE solver (see Table 12.1 in Higham and Higham (2005)) can be applied for implementing the earlier- published accurate continuous-discrete UKF methods and their newly-proposed square-root versions as well.

The continuous-discrete UKF method based on solving the related SPDEs is summarized in Algorithm 1.

Algorithm 1. SPDE-BASED CD-UKF (*Conventional*)

```
   INITIALIZATION: (at time instance $t_0$)
1      Set $\xi_0 = \mathbf{0}$, $\xi_i = \sqrt{n+\lambda}e_i$, $\xi_{n+i} = -\sqrt{n+\lambda}e_i$
2          $(i = \overline{1,n})$ and find $w^{(m)}$, $w^{(c)}$, $\mathbb{W}$ by (3), (4);
3      Set solver's options with given tolerances $\epsilon_a$, $\epsilon_r$:
       options = odeset('AbsTol',$\epsilon_a$,'RelTol',$\epsilon_r$,
           'MaxStep',0.1);
4      Set initial values $\hat{x}_{0|0} = \bar{x}_0$ and $P_{0|0} = \Pi_0$;
   TIME UPDATE: (at interval $[t_{k-1}, t_k]$) $\triangleright$ PRIORI EST.
5      Cholesky dec.: $P_{k-1|k-1} = P^{1/2}_{k-1|k-1}P^{\top/2}_{k-1|k-1}$;
6      $\mathcal{X}_{i,k-1|k-1} = \hat{x}_{k-1|k-1} + P^{1/2}_{k-1|k-1}\xi_i$, $(i = \overline{0,2n})$;
7      Form $\mathbb{X}_{k-1|k-1} = \left[\mathcal{X}_{0,k-1|k-1}, \ldots, \mathcal{X}_{2n,k-1|k-1}\right]$;
8      Reshape in MATLAB: $\widetilde{\mathbb{X}}_{k-1|k-1} = \mathbb{X}_{k-1|k-1}$(:);
9      $\widetilde{\mathbb{X}}_{k|k-1} \leftarrow$ odesolver[SPDEs, $\widetilde{\mathbb{X}}_{k-1|k-1}, [t_{k-1}, t_k]$];
10     $\mathbb{X}_{k|k-1} \leftarrow$ reshape($\widetilde{\mathbb{X}}_{k|k-1}(t_k), n, 2n+1$);
11     Recover the state $\hat{x}_{k|k-1} = \mathcal{X}_{0,k|k-1} = [\mathbb{X}_{k|k-1}]_1$;
12     $S_{k|k-1} = \texttt{tril}([\mathbb{X}_{k|k-1}]_{2:2n+1} - \hat{x}_{k|k-1})/\sqrt{n+\lambda}$;
13     Recover the covariance $P_{k|k-1} = S_{k|k-1}S^{\top}_{k|k-1}$;
   MEASUREMENT UPDATE: (at $t_k$) $\triangleright$ POSTERIORI EST.
14     Propagate $\mathcal{Z}_{i,k|k-1} = h\big(k, \mathbb{X}_{k|k-1}\big)$, $(i = \overline{0,2n})$;
15     Collect $\mathbb{Z}_{k|k-1} = \left[\mathcal{Z}_{0,k|k-1}, \ldots, \mathcal{Z}_{2n,k|k-1}\right]$;
16     Compute the predicted $\hat{z}_{k|k-1} = \mathbb{Z}_{k|k-1}w^{(m)}$;
17     Find $R_{e,k} = \mathbb{Z}_{k|k-1}\mathbb{W}\mathbb{Z}^{\top}_{k|k-1} + R_k$;
18     Compute $P_{xz,k} = \mathbb{X}_{k|k-1}\mathbb{W}\mathbb{Z}^{\top}_{k|k-1}$;
19     Calculate the filter gain $\mathbb{K}_k = P_{xz,k}R^{-1}_{e,k}$;
20     Update $\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbb{K}_k(z_k - \hat{z}_{k|k-1})$;
21     Update $P_{k|k} = P_{k|k-1} - \mathbb{K}_k R_{e,k}\mathbb{K}^{\top}_k$.
```

In summary, the set of $2n+1$ SPDE equations (5) should be solved at each sampling interval $[t_{k-1}, t_k]$ for propagating the sigma vectors and, then, they are utilized at the measurement update step for computing the state estimate and the related error covariance matrix. We indicate the utilized numerical integration scheme as `odesolver` where values ϵ_a and ϵ_r are, respectively, absolute and relative tolerances given by users for solving the ODEs.

We stress that Algorithm 1 is, in fact, the conventional implementation of the UKF estimator because of the full error covariance matrix update at each iteration step; see lines 13 and 21 in Algorithm 1. As a result, the Cholesky decomposition is required at each filtering step for generating the sigma vectors at line 5 of Algorithm 1. It is worth noting here that the examined SPDE-based implementation of the continuous-discrete UKF estimator allows one of the demanded Cholesky factorizations to be avoided compared to the MDEs-based implementations. In fact, it is clearly seen that one does not need to generate the sigma vectors at the measurement update step of Algorithm 1 because the propagated sigma vectors are directly available after the time update step; see line 10 in Algorithm 1. Thus, the Cholesky decomposition is avoided at the measurement update step in each filter iterate.

Taking into account that the square-root factor $S_{k|k-1}$ is recovered when the SPDEs are solved, it makes sense to re-derive the measurement update equations in terms of the square-root factors $S_{k|k-1}$ and $S_{k|k}$, only. A possible solution yields the square-root implementation of the discussed SPDE-based continuous-discrete UKF estimator. We propose a novel solution in the next section.

To conclude this section, we give the pseudo-code expressed by MATLAB language notation for computing the right-hand side of the SPDEs function, which is to be sent to the chosen ODEs solver at the time update step.

```
$[\widetilde{\mathbb{X}}'(t)] \leftarrow$ SPDEs($\widetilde{\mathbb{X}}(t), t, n, \lambda, \mathbb{W}, w^{(m)}, G, Q$)
1      Get matrix $\mathbb{X}(t) \leftarrow$ reshape($\widetilde{\mathbb{X}}(t), n, 2n+1$);
2      Recover $\hat{x}(t) = [\mathbb{X}(t)]_1$;
3      Recover $S(t) = \texttt{tril}\big([\mathbb{X}(t)]_{2:2n+1} - \hat{x}(t)\big)/\sqrt{n+\lambda}$;
4      Propagate $f\big(t, \mathbb{X}(t)$ and find $\mathbb{X}(t)\mathbb{W}f^{\top}\big(t, \mathbb{X}(t)\big)$;
5      Find $M(t)$ by (6) and split $M = \bar{L} + D + \bar{U}$;
6      Compute $\Phi(M) = \bar{L} + 0.5D$;
7      Set matrix $[\mathbb{X}'(t)]$ by columns defined in (5);
8      Reshape into a vector form $[\widetilde{\mathbb{X}}'(t)] = [\mathbb{X}'(t)]$(:).
```

Finally, the built-in MATLAB function `tril` used both in the pseudo-code above and Algorithm 1 is required for taking a lower triangular part of the computed matrices. It is done for the following reason. In fact, when the covariance square-root factor is recovered by formula $S(t) = \big([\mathbb{X}(t)]_{2:2n+1} - \hat{x}(t)\big)/\sqrt{n+\lambda}$, the resulted matrix $S(t)$ should be a lower triangular matrix. However, in a finite precision arithmetic, this is not always the case. To avoid any unexpected non-zero entries in the computed square-root factor, we skip such entries by implementing the operation `tril`. Theoretically, it is not required, but preferable while implementing in practice.

3. SQUARE-ROOT SOLUTION

Derivation of a square-root solution means that we should re-derive the SPDE-based UKF equations in terms of propagating and updating the Cholesky factors $S_{k|k-1}$ and $S_{k|k}$ of the filter error covariance matrix. The new equations should be equivalent to the conventional ones summarized in Algorithm 1, but the numerical properties of these two computational approaches will no longer agree. The square-root methods are numerically stable with respect to roundoff errors. Taking into account the condition number of covariance matrices, Cholesky-based implementations are proved to maintain the computations with a double precision as shown in Kaminski et al. (1971).

The key problem for finding the square-root solution for the discussed UKF is the possibly negative weights $w^{(c)}$ and unavailable square root operation for negative real numbers. To resolve this problem, we suggest the following elegant strategy. First, we define the square-root matrix $|\mathbb{W}|^{1/2}$ where the required square root operation is taken for the absolute values $|w^{(c)}|$. In addition, we determine the corresponding signature matrix in order to save the information about the sign of these entries[1]. In summary,

$$|\mathbb{W}|^{1/2} = \left[I_{2n+1} - \mathbf{1}_{2n+1}^{\top} \otimes w^{(m)}\right] \times \operatorname{diag}\left\{\sqrt{|w_0^{(c)}|}, \ldots, \sqrt{|w_{2n}^{(c)}|}\right\},$$
$$S = \operatorname{diag}\left\{\operatorname{sgn}(w_0^{(c)}), \ldots, \operatorname{sgn}(w_{2n}^{(c)})\right\}.$$

Next, the appearance of a signature matrix in our square-root solution demands the utilization of a hyperbolic QR factorization instead of traditionally used standard QR factorization. Following Higham (2003), the J-orthogonal matrix Q is defined as one, which satisfies $Q^{\top}JQ = QJQ^{\top} = J$ where $J = \operatorname{diag}(\pm 1)$. Clearly, if the matrix J equals to identity matrix, i.e. $J = I$, then the J-orthogonal matrix Q reduces to usual orthogonal one.

Following Higham (2003), the J-orthogonal transformations are used for computing the Cholesky factorization of a positive definite matrix $C = A^{\top}A - B^{\top}B$, where $A \in \mathbb{R}^{p\times n}$ $(p \geq n)$ and $B \in \mathbb{R}^{q\times n}$. If we can find a J-orthogonal matrix Q such that

$$Q\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} \qquad (7)$$

with $J = \operatorname{diag}\{I_p, -I_q\}$, $R \in \mathbb{R}^{n\times n}$ upper triangular, then

$$C = \begin{bmatrix} A \\ B \end{bmatrix}^{\top} J \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix}^{\top} Q^{\top}JQ \begin{bmatrix} A \\ B \end{bmatrix} = R^{\top}R$$

so R is the Cholesky factor, which we are looking for. The factorization in (7) is called the hyperbolic QR factorization and its effective implementation methods are discussed in Bojanczyk et al. (2003).

Thus, we propose the following square-root solution for the examined UKF estimator with the classical parametrization $\alpha = 1$, $\beta = 0$ and $\kappa = 3 - n$: (i) negative weights in $w^{(c)}$ are sorted in such a way that they are located at the end of the column; (ii) the column $w^{(m)}$ is sorted at the same way as $w^{(c)}$; (iii) the matrix $\mathbb{W}$ is defined by (4); (iv) finally, define the square-root and related signature matrices as follows:

$$|\mathbb{W}|_{cUKF}^{1/2} = \left[I_{2n+1} - \mathbf{1}_{2n+1}^{\top} \otimes w^{(m)}\right] \times \operatorname{diag}\left\{\sqrt{\frac{1}{6}}, \ldots, \sqrt{\frac{|3-n|}{3}}\right\} \qquad (8)$$

with the signature matrix

$$S_{cUKF} = \operatorname{diag}\{1, \ldots, 1, \operatorname{sgn}((3-n)/3)\}. \qquad (9)$$

The above steps yield the signature matrix of the form $J = \operatorname{diag}\{I_p, -I_q\}$ required in hyperbolic transformation (7) with $p = 2n$ and $q = 1$ for the estimation problems of size $n > 3$. Now, we are ready to formulate the first square-root implementation method for the discussed SPDE-based continuous-discrete UKF estimator.

Algorithm 1a. SPDE-BASED SR CD-UKF (*Cholesky*)

```
    INITIALIZATION: Repeat lines 1-3 of Algorithm 1;
1       Rearrange $w^{(c)}$ and, then, $w^{(m)}$ similarly;
2       Find $|\mathbb{W}|_{cUKF}^{1/2}$ and signature $S_{cUKF}$ by (8), (9);
3       Cholesky factorization: $\Pi_0 = \Pi_0^{1/2}\Pi_0^{\top/2}$;
4       Set initial values $\hat{x}_{0|0} = \bar{x}_0$ and $P_{0|0}^{1/2} = \Pi_0^{1/2}$;
    TIME UPDATE: (at interval $[t_{k-1}, t_k]$) ▷ PRIORI EST.
5       Repeat lines 6-12 of Algorithm 1, but note that
        here $\mathbb{W} := \mathbb{W}_{cUKF} = |\mathbb{W}|_{cUKF}^{1/2} S_{cUKF} |\mathbb{W}|_{cUKF}^{\top/2}$;
    MEASUREMENT UPDATE: (at $t_k$) ▷ POSTERIORI EST.
6       Propagate $\mathcal{Z}_{i,k|k-1} = h(k, \mathbb{X}_{k|k-1})$, $(i = \overline{0, 2n})$;
7       Collect $\mathbb{Z}_{k|k-1} = [\mathcal{Z}_{0,k|k-1}, \ldots, \mathcal{Z}_{2n,k|k-1}]$;
8       Compute the predicted $\hat{z}_{k|k-1} = \mathbb{Z}_{k|k-1} w^{(m)}$;
9       Build pre-array $\mathbb{A}_k$ and triangularize it
        $\underbrace{\begin{bmatrix} R_k^{1/2} & \mathbb{Z}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2} \\ \mathbf{0} & \mathbb{X}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2} \end{bmatrix}}_{\text{Pre-array } \mathbb{A}_k} \mathbb{Q} = \underbrace{\begin{bmatrix} R_{e,k}^{1/2} & \mathbf{0}_{m\times n} & \mathbf{0}_{m\times(n+1)} \\ \bar{P}_{xz,k} & P_{k|k}^{1/2} & \mathbf{0}_{n\times(n+1)} \end{bmatrix}}_{\text{Post-array } \mathbb{R}_k}$
        with $\mathbb{Q}$ is any $J = \operatorname{diag}\{I_m, S_{cUKF}\}$-orthogonal
        transformation that block lower triangulates;
10      Extract and save blocks $R_{e,k}^{1/2}$, $\bar{P}_{xz,k}$ and $P_{k|k}^{1/2}$;
11      Calculate the filter gain $\mathbb{K}_k = \bar{P}_{xz,k} R_{e,k}^{-1/2}$;
12      Update $\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbb{K}_k(z_k - \hat{z}_{k|k-1})$.
```

Following the definition of hyperbolic QR factorization and the fact that $\mathbb{A}_k J \mathbb{A}_k^{\top} = \mathbb{A}_k \mathbb{Q}_k J \mathbb{Q}_k^{\top} \mathbb{A}_k^{\top} = \mathbb{R}_k J \mathbb{R}_k^{\top}$, one may justify the algebraic equivalence between equations of Algorithm 1 and 1a by computing $\mathbb{A}_k J \mathbb{A}_k^{\top}$ and comparing with $\mathbb{R}_k J \mathbb{R}_k^{\top}$. Indeed, let us consider the transformation in line 9 of Algorithm 1a that implies

$$\begin{aligned}
\mathbb{A}_k J \mathbb{A}_k^{\top} &= \left[R_k^{1/2}, \mathbb{Z}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}\right]\left[\operatorname{diag}\{I_m, S_{cUKF}\}\right] \\
&\quad \times \left[R_k^{1/2}, \mathbb{Z}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}\right]^{\top} \\
&= R_k^{1/2} I_m R_k^{\top/2} + \mathbb{Z}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2} S_{cUKF} |\mathbb{W}|^{\top/2} \mathbb{Z}_{k|k-1}^{\top} \\
&= R_k + \mathbb{Z}_{k|k-1} \mathbb{W} \mathbb{Z}_{k|k-1}^{\top} \overset{\text{line 17, Alg. 1}}{=} R_{e,k} \\
=\mathbb{R}_k J \mathbb{R}_k^{\top} &= \left[R_{e,k}^{1/2}, \mathbf{0}_{m\times n}, \mathbf{0}_{m\times(n+1)}\right]\left[\operatorname{diag}\{I_m, S_{cUKF}\}\right] \\
&\quad \times \left[R_{e,k}^{1/2}, \mathbf{0}_{m\times n}, \mathbf{0}_{m\times(n+1)}\right]^{\top}.
\end{aligned}$$

Similarly, we justify other formulas as follows:

$$\begin{aligned}
\mathbb{A}_k J \mathbb{A}_k^{\top} &= \left[R_k^{1/2}, \mathbb{Z}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}\right]\left[\operatorname{diag}\{I_m, S_{cUKF}\}\right] \\
&\quad \times \left[\mathbf{0}, \mathbb{X}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}\right]^{\top} = \mathbb{Z}_{k|k-1} \mathbb{W} \mathbb{X}_{k|k-1}^{\top} \overset{\text{Alg.1,18}}{=} P_{xz,k}^{\top} \\
=\mathbb{R}_k J \mathbb{R}_k^{\top} &= \left[R_{e,k}^{1/2}, \mathbf{0}_{m\times n}, \mathbf{0}_{m\times(n+1)}\right]\left[\operatorname{diag}\{I_m, S_{cUKF}\}\right] \\
&\quad \times \left[\bar{P}_{xz,k}, P_{k|k}^{1/2}, \mathbf{0}_{n\times(n+1)}\right]^{\top} = R_{e,k}^{1/2} I_m \bar{P}_{xz,k}^{\top},
\end{aligned}$$

[1] For zero entries $w_i^{(c)} = 0$ we set $\operatorname{sgn}(w_i^{(c)}) = 1$.

i.e. $R_{e,k}^{1/2}\bar{P}_{xz,k}^{\top} = P_{xz,k}^{\top}$ and, hence, $\bar{P}_{xz,k} = P_{xz,k}R_{e,k}^{-\top/2}$. Thus, the gain matrix is calculated by using the available post-array blocks $\bar{P}_{xz,k}$ and $R_{e,k}^{1/2}$ as follows:

$$\mathbb{K}_k = P_{xz,k}R_{e,k}^{-1} = \bar{P}_{xz,k}R_{e,k}^{-1/2}.$$

Finally, we get

$$\begin{aligned}
\mathbb{A}_k J \mathbb{A}_k^{\top} &= \left[\mathbf{0}, \mathbb{X}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}\right]\left[\mathrm{diag}\{I_m, S_{cUKF}\}\right] \\
&\times \left[\mathbf{0}, \mathbb{X}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}\right]^{\top} = \mathbb{X}_{k|k-1}\mathbb{W}\mathbb{X}_{k|k-1}^{\top} = P_{k|k-1} \\
=\mathbb{R}_k J \mathbb{R}_k^{\top} &= \left[\bar{P}_{xz,k}, P_{k|k}^{1/2}, \mathbf{0}_{n\times(n+1)}\right]\left[\mathrm{diag}\{I_m, S_{cUKF}\}\right] \\
&\times \left[\bar{P}_{xz,k}, P_{k|k}^{1/2}, \mathbf{0}_{n\times(n+1)}\right]^{\top} = \bar{P}_{xz,k}\bar{P}_{xz,k}^{\top} \\
&+ P_{k|k}^{1/2}[S_{cUKF}]_{n\times n}P_{k|k}^{\top/2} = \mathbb{K}_k R_{e,k}\mathbb{K}_k^{\top} + P_{k|k}^{1/2} I_n P_{k|k}^{\top/2}.
\end{aligned}$$

This completes the proof of algebraic equivalence between Algorithms 1 and 1a. The numerical properties of these methods no longer agree. For instance, the inverse of $R_{e,k}^{1/2}$ is required in the square-root method instead of $R_{e,k}$.

Finally, one more square-root method can be derived by using a symmetric UKF equation that is similar to a symmetric Joseph stabilized equation existing for the classical KF; see Simon (2006); Grewal and Andrews (2015). To derive it, we consider the equation

$$P_{k|k} = P_{k|k-1} - \mathbb{K}_k R_{e,k}\mathbb{K}_k^{\top} = P_{k|k-1} - P_{xz,k}\mathbb{K}_k^{\top} \qquad (10)$$

and get

$$\begin{aligned}
P_{k|k} &= P_{k|k-1} - P_{xz,k}\mathbb{K}_k^{\top} + \mathbb{K}_k R_{e,k}\mathbb{K}_k^{\top} - \mathbb{K}_k P_{xz,k}^{\top} \\
&= \mathbb{X}_{k|k-1}\mathbb{W}\mathbb{X}_{k|k-1}^{\top} - \mathbb{X}_{k|k-1}\mathbb{W}\mathbb{Z}_{k|k-1}^{\top}\mathbb{K}_k^{\top} \\
&- \mathbb{K}_k\mathbb{Z}_{k|k-1}\mathbb{W}\mathbb{X}_{k|k-1}^{\top} + \mathbb{K}_k(\mathbb{Z}_{k|k-1}\mathbb{W}\mathbb{Z}_{k|k-1}^{\top} + R_k)\mathbb{K}_k^{\top} \\
&= \left[\mathbb{X}_{k|k-1} - \mathbb{K}_k\mathbb{Z}_{k|k-1}\right]\mathbb{W}\left[\mathbb{X}_{k|k-1} - \mathbb{K}_k\mathbb{Z}_{k|k-1}\right]^{\top} \\
&+ \mathbb{K}_k R_k \mathbb{K}_k^{\top}
\end{aligned}$$

The resulted symmetric formula allows for the hyperbolic QR factorization and, hence, we summarize an alternative square-root UKF implementation method as follows.

Algorithm 1b. SPDE-based SR CD-UKF (*Cholesky*)

```
  Initialization: Repeat from Algorithm 1a;
  Time Update: (at interval $[t_{k-1}, t_k]$) $\triangleright$ Priori est.
1     Repeat from Algorithm 1a;
  Measurement Update: (at $t_k$) $\triangleright$ Posteriori est.
2     Propagate $\mathcal{Z}_{i,k|k-1} = h\big(k, \mathbb{X}_{k|k-1}\big)$, $(i = \overline{0, 2n})$;
3     Collect $\mathbb{Z}_{k|k-1} = \left[\mathcal{Z}_{0,k|k-1}, \ldots, \mathcal{Z}_{2n,k|k-1}\right]$;
4     Compute the predicted $\hat{z}_{k|k-1} = \mathbb{Z}_{k|k-1}w^{(m)}$;
5     Build pre-array $\mathbb{A}_k$ and triangularize it
      $\underbrace{\left[R_k^{1/2} \quad \mathbb{Z}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}\right]}_{\mathrm{Pre-array}\ \mathbb{A}_k}\mathbb{Q} = \underbrace{\left[R_{e,k}^{1/2} \quad \mathbf{0}_{m\times(2n+1)}\right]}_{\mathrm{Post-array}\ \mathbb{R}_k}$;
      with $\mathbb{Q}$ is any $J = \mathrm{diag}\{I_m, S_{cUKF}\}$-orthogonal
      transformation that block lower triangulates;
6     $P_{xz,k} = \mathbb{X}_{k|k-1}|\mathbb{W}|_{cUKF}^{1/2}S_{cUKF}|\mathbb{W}|_{cUKF}^{\top/2}\mathbb{Z}_{k|k-1}^{\top}$;
7     Calculate the filter gain $\mathbb{K}_k = P_{xz,k}R_{e,k}^{-\top/2}R_{e,k}^{-1/2}$;
8     Update $\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbb{K}_k(z_k - \hat{z}_{k|k-1})$;
9     Build pre-array $\mathbb{A}_k$ and lower triangularize it
      $\underbrace{\left[\mathbb{K}_k R_k^{1/2},\ \left[\mathbb{X}_{k|k-1} - \mathbb{K}_k\mathbb{Z}_{k|k-1}\right]|\mathbb{W}|_{cUKF}^{1/2}\right]}_{\mathrm{Pre-array}\ \mathbb{A}_k}\mathbb{Q}$
      $= \underbrace{\left[P_{k|k}^{1/2} \ 0\right]}_{\mathrm{Post-array}\ \mathbb{R}_k}$ where $J = \mathrm{diag}\{I_m, S_{cUKF}\}$.
```

4. NUMERICAL EXPERIMENTS

Example 1. Following Arasaratnam et al. (2010), consider the aircraft's dynamics when performing a coordinated turn in the horizontal plane with the following drift function and diffusion matrix

$$f(\cdot) = \left[\dot{\epsilon}, -\omega\dot{\eta}, \dot{\eta}, \omega\dot{\epsilon}, \dot{\zeta}, 0, 0\right], G = \mathrm{diag}\left[0, \sigma_1, 0, \sigma_1, 0, \sigma_1, \sigma_2\right]$$

where $\sigma_1 = \sqrt{0.2}$, $\sigma_2 = 0.007$ and $\beta(t)$ is the *standard* Brownian motion, i.e. $Q = I_7$. The state vector consists of seven entries, i.e. $x(t) = [\epsilon, \dot{\epsilon}, \eta, \dot{\eta}, \zeta, \dot{\zeta}, \omega]^{\top}$, where ϵ, η, ζ and $\dot{\epsilon}$, $\dot{\eta}$, $\dot{\zeta}$ stand for positions and corresponding velocities in the Cartesian coordinates at time t, and $\omega(t)$ is the (nearly) constant turn rate. The initial conditions are $\bar{x}_0 = [1000\,\mathrm{m}, 0\,\mathrm{m/s}, 2650\,\mathrm{m}, 150\,\mathrm{m/s}, 200\,\mathrm{m}, 0\,\mathrm{m/s}, \omega^{\circ}/\mathrm{s}]^{\top}$ and $\Pi_0 = \mathrm{diag}(0.01\, I_7)$. We fix the turn rate to $\omega = 3^{\circ}/\mathrm{s}$.

Following Examples 7.1 and 7.2 in Dyer and McReynolds (1969), simulate the ill-conditioned measurement scheme in a similar manner as follows:

$$z_k = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1+\delta \end{bmatrix} x_k + \begin{bmatrix} v_k^1 \\ v_k^2 \end{bmatrix}, \; R_k = \delta^2 I_2$$

where parameter δ is used for simulating roundoff effect, i.e. it is assumed to tend $\delta \to \epsilon_{roundoff}$ where $\epsilon_{roundoff}$ stands for the unit roundoff error.

When the ill-conditioning parameter δ in Example 1 tends to machine precision limit, $\delta \to \epsilon_{roundoff}$, one observes a degradation of any KF-like nonlinear Bayesian filtering methods due to roundoff errors that make the residual covariance matrix $R_{e,k}$ to be ill-conditioned although the matrix H_k is always of a full rank (Grewal and Andrews, 2015, p. 288). In this paper, we perform the following set of numerical experiments. First, we fix the ill-conditioning parameter to $\delta = 10^{-1}$. Next, we solve the state estimation problem on interval $[0s, 150s]$ with various sampling periods $\Delta = 1, 2, \ldots, 10(s)$ by the SPDE-based continuous-discrete UKF methods in Algorithms 1, 1a and 1b under examination. For their implementation, we utilize the built-in MATLAB ODEs solver `ode113` with the given tolerance $\epsilon_a = \epsilon_r = 10^{-8}$. We stress that all filtering methods are tested at the same conditions, i.e. with the same simulated "true" state trajectory, the same measurement data and the same initial filter conditions.

To simulate the "true" stochastic aircraft's dynamics, we solve the given SDEs by the Euler-Maruyama method with the small step size $0.0005(s)$. Next, for each fixed value of Δ, we get x_k^{true}, $k = 1, \ldots, K$ and simulate the related measurements. Finally, given the available measurements, the inverse (filtering) problem is solved, i.e. we estimate the unknown dynamic state $\hat{x}_{k|k}$, $k = 1, \ldots, K$ by all filtering methods to be examined. To justify the estimation quality of the UKF estimators, the square errors are calculated at all sampling points by taking the norm of the difference between the "true" solution x_k^{true} and the

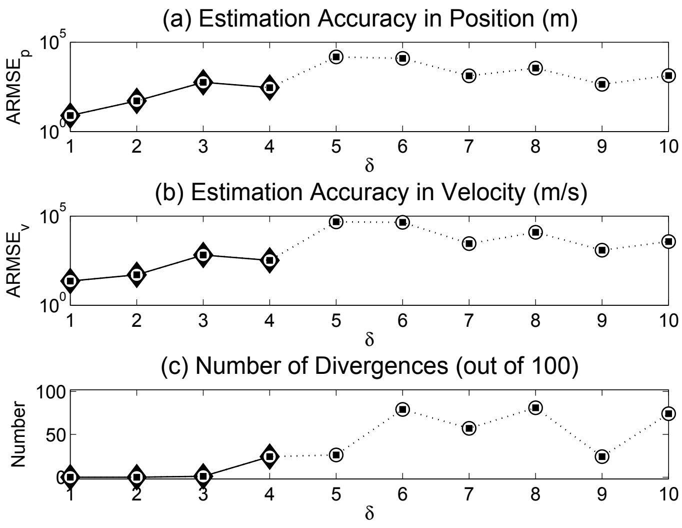

Fig. 1. Accuracy degradation in Example 1: Algorithm 1 (solid line with ◆), Algorithm 1a (dotted line with ∘), Algorithm 1b (dotted line with ■)

estimated one $\hat{x}_{k|k}$ for all $k = 1, \dots, K$. We repeat the experiment for 100 Monte Carlo simulations and compute the accumulated root mean square error (ARMSE) by averaging over 100 Monte Carlo runs. Following Arasaratnam et al. (2010), we compute the ARMSE in position ARMSE$_p$, the ARMSE in velocity ARMSE$_v$ and the number of divergences when ARMSE$_p > 500(m)$ (out of 100 runs), respectively, by the formulas given in the cited paper. The obtained results are illustrated by Fig. 1.

Having analyzed Fig. 1, we conclude that the conventional implementation summarized in Algorithm 1 rapidly fails to solve this moderate ill-conditioned test problem for large sampling intervals. Indeed, it is easily seen that all filtering algorithms under examination work with the same precision while sampling intervals are small enough, i.e. for $\Delta \leq 4(s)$. This justifies our theoretical derivations of the novel square-root methods presented in this paper and illustrates in practice the proved mathematical equivalence of all three UKF implementation methods examined. However, in contrast to the conventional implementation in Algorithm 1, which fails for $\Delta > 5(s)$, the novel Cholesky-based square-root methods treat in a robust way the long sampling interval scenario. Indeed, the roundoff errors destroy the theoretical properties of the error covariance matrix and, hence, the conventional Algorithm 1 fails because of the unavailable Cholesky factorization for the computed covariance. Meanwhile, the novel square-root Algorithms 1a and 1b are robust with respect to roundoff errors. In particular, they do not require the Cholesky factorization at each iteration filtering step since the square-root matrices are propagated and updated while estimation procedure. Thus, the obtained results justify that the newly-suggested Cholesky factorization-based continuous-discrete UKF methods outperform their conventional counterpart for the numerical robustness.

5. CONCLUSION

The problem of designing square-root methods for the UKF estimator with possibly negative sigma points' weights is solved. The solution is based on the hyperbolic QR factorization for updating the required Cholesky factors. The new methods are derived for the classical UKF variant. Numerical experiments indicate a superior performance of the novel square-root UKF algorithms.

REFERENCES

Arasaratnam, I. and Haykin, S. (2009). Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6), 1254–1269.

Arasaratnam, I., Haykin, S., and Hurd, T.R. (2010). Cubature Kalman filtering for continuous-discrete systems: Theory and simulations. *IEEE Transactions on Signal Processing*, 58(10), 4977–4993.

Bojanczyk, A., Higham, N.J., and Patel, H. (2003). Solving the indefinite least squares problem by hyperbolic QR factorization. *SIAM Journal on Matrix Analysis and Applications*, 24(4), 914–931.

Dyer, P. and McReynolds, S. (1969). Extensions of square root filtering to include process noise. *Journal of Optimization Theory and Applications*, 3(6), 444–459.

Golub, G.H. and Van Loan, C.F. (1983). *Matrix Computations.* Johns Hopkins University Press, Baltimore, Maryland.

Grewal, M.S. and Andrews, A.P. (2015). *Kalman Filtering: Theory and Practice using MATLAB.* John Wiley & Sons, New Jersey, 4-th edition edition.

Higham, D.J. and Higham, N.J. (2005). *MatLab Guide.* SIAM, Philadelphia.

Higham, N.J. (2003). J-orthogonal matrices: properties and generalization. *SIAM Review*, 45(3), 504–519.

Kailath, T., Sayed, A.H., and Hassibi, B. (2000). *Linear Estimation.* Prentice Hall, New Jersey.

Kaminski, P.G., Bryson, A.E., and Schmidt, S.F. (1971). Discrete square-root filtering: a survey of current techniques. *IEEE Transactions on Automatic Control*, AC-16(6), 727–735.

Kulikov, G.Yu. and Kulikova, M.V. (2017). Accurate continuous-discrete unscented Kalman filtering for estimation of nonlinear continuous-time stochastic models in radar tracking. *Signal Process.*, 139, 25–35.

Kulikov, G.Yu. and Kulikova, M.V. (2018). Stability analysis of Extended, Cubature and Unscented Kalman filters for estimating stiff continuous-discrete stochastic systems. *Automatica*, 90, 91–97.

Menegaz, H.M., Ishihara, J.Y., Borges, G.A., and Vargas, A.N. (2015). A systematization of the unscented Kalman filter theory. *IEEE Transactions on Automatic Control*, 60(10), 2583–2598.

Särkkä, S. (2007). On unscented Kalman filter for state estimation of continuous-time nonlinear systems. *IEEE Transactions on Automatic Control*, 52(9), 1631–1641.

Simon, D. (2006). *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches.* John Wiley & Sons.

Van der Merwe, R. and Wan, E.A. (2001). The square-root unscented Kalman filter for state and parameter-estimation. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings*, volume 6, 3461–3464.

Wan, E.A. and Van der Merwe, R. (2001). The unscented Kalman filter. In S. Haykin ed. *Kalman Filtering and Neural Networks*, 221–280. John Wiley & Sons, Inc., New York.