Nonlinear Filtering
Unscented Kalman filtering with SVD

Tohru Katayama
Emeritus Professor of Kyoto University
Visiting Professor of Doshisha University
Chair Professor of Ritsumeikan University
Joint Paper

Michiaki Takeno and Tohru Katayama, On the selection of $\sigma$ points and the improvement of performance in Unscented Kalman filter based on singular value decomposition (in preparation).

In the Unscented Transformation (UT) used for Unscented Kalman Filter (UKF), we need to compute square roots of covariance matrices to select sigma points, which approximate the covariance information of conditional probabilities. We show by using a 2nd-order model that SVD-based matrix square roots better capture the covariance information than Cholesky decomposition. Simulation results for several discrete and continuous nonlinear systems are also included to show the applicability of the SVD-based UKF algorithm.
Nonlinear Filtering

- Nonlinear filtering has a long history, i.e. in 1960s, there published many papers, including
  Cox (IEEE 1964)
  Kushner (SIAM 1964)
  Bucy (IEEE 1965)
  Jazwinski (1970)

- Extended Kalman filter (EKF) has been tested and successfully used for Apollo project at NASA during 1960s (Grewal & Andrews, CS Magazine 2010).
## Nonlinear Filtering Techniques

<table>
<thead>
<tr>
<th>Local methods</th>
<th>EKF</th>
<th>Extended Kalman Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SLKF</td>
<td>Statistical Linearization Kalman Filter</td>
</tr>
<tr>
<td></td>
<td>UKF</td>
<td>Unscended Kalman Filter</td>
</tr>
<tr>
<td></td>
<td>EnKF</td>
<td>Ensemble Kalman Filter</td>
</tr>
<tr>
<td>Global methods</td>
<td>GSF</td>
<td>Gaussian Sum Filter</td>
</tr>
<tr>
<td></td>
<td>PMF</td>
<td>Point Mass Filter</td>
</tr>
<tr>
<td></td>
<td>PF</td>
<td>Particle Filter</td>
</tr>
</tbody>
</table>

- S. Lakshmivarahan and D. J. Stensrud, *CS Magazine*, June 2009
Outline

- Nonlinear Filtering
- Statistical Linearization
- Unscented Transformation
- A Motivating Example
- Cholesky decomposition vs. SVD
- Several Simulation Results
- Conclusions
Nonlinear Filtering

Consider a nonlinear stochastic system

\[
x_{t+1} = f_t(x_t) + w_t
\]
\[
y_t = h_t(x_t) + v_t
\]

\(x_t \in \mathbb{R}^n: \text{the state vector, } y_t \in \mathbb{R}^p: \text{the output vector}\)

\(f_t: \mathbb{R}^n \rightarrow \mathbb{R}^n, \ h_t: \mathbb{R}^n \rightarrow \mathbb{R}^p: \text{nonlinear transforms}\)

\(w_t, v_t: \text{Gaussian white noises}\)

\[
E[w_t^T \ v_t^T] \begin{bmatrix} w_s \\ v_s \end{bmatrix} = \begin{bmatrix} Q_t & 0 \\ 0 & R_t \end{bmatrix} \delta_{ts}
\]
Nonlinear Filtering

The problem is to compute the conditional expectation, or the conditional mean estimate

$$\hat{x}_{t+m|t} = E\{x_{t+m} \mid Y^t\} = \int_{\mathbb{R}^n} x_{t+m} p(x_{t+m} \mid Y^t) dx_{t+m}$$

$$m = 0, 1$$

$$Y^t = \{y_0, y_1, \cdots, y_t\}$$

which minimizes the conditional Bayes risk

$$J = E\{||x_{t+m} - \hat{x}_{t+m|t}||^2 \mid Y^t\}$$
Conditional Probabilities

- Observation update

\[ p(x_t | Y^t) = \frac{p(y_t | x_t)p(x_t | Y^{t-1})}{p(y_t | Y^{t-1})} \]

- Time update

\[ p(x_{t+1} | Y^t) = \int_{\mathbb{R}^n} p(x_{t+1} | x_t)p(x_t | Y^t)dx_t \]

Except for the Linear Gaussian case, it is impossible to find the optimal solution to filtering problems, though there are some results for obtaining exact solutions (Beneš 1981, Daum 1986).
Statistical Linearization

\[ g : \mathbb{R}^n \rightarrow \mathbb{R}^p: \text{a nonlinear transformation} \]
\[ x \in \mathbb{R}^n, y \in \mathbb{R}^p: \text{random variables} \]

The problem is to find the unbiased linear minimum variance estimate of \( g(\cdot) \), i.e.

\[ J = E\|e\|^2 = E\|y - Ax - b\|^2 \rightarrow \min_{A,b}, \ A \in \mathbb{R}^{p \times n}, \ b \in \mathbb{R}^p \]
Statistical Linearization

- Unbiased estimate

\[ E(y - Ax - b) = 0 \rightarrow \mu_y = A\mu_x + b \]

\[
\min J = \text{trace} E[(y - Ax - b)(y - Ax - b)^T] \\
= \text{trace} E[y - \mu_y - A(x - \mu_x)][y - \mu_y - A(x - \mu_x)]^T \\
= \text{trace} \left[ \Sigma_{yy} - A\Sigma_{xy} - \Sigma_{yx}A^T + A\Sigma_{xx}A^T \right] \\
A = \Sigma_{yx}\Sigma_{xx}^{-1}, \quad b = \mu_y - \Sigma_{yx}\Sigma_{xx}^{-1}\mu_x \rightarrow \hat{y} = \mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x-\mu_x)
\]

- However, computation of \( \mu_y, \Sigma_{yx} \) is almost impossible except for the case where \( x \sim N(\mu, \Sigma) \) and the nonlinearity \( g(\cdot) \) is a polynomial.
LS Problem

Given the data \((x_i, y_i), i = 1, \cdots, N\), we have a LS problem:

\[
J = \frac{1}{N} \sum_{i=1}^{N} e_i^T e_i = \frac{1}{N} \sum_{i=1}^{N} (y_i - Ax_i - b)^T (y_i - Ax_i - b) \rightarrow \min_{A,b}
\]

\[
\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^{N} g(x_i), \quad S_{yx} = \frac{1}{N} \sum_{i=0}^{N} [g(x_i) - \bar{y}][x_i - \bar{x}]^T
\]

\[
A = S_{yx} S_{xx}^{-1}, \quad b = \bar{y} - S_{yx} S_{xx}^{-1} \bar{x}
\]

\[
\rightarrow \quad \hat{y} = \bar{y} + S_{yx} S_{xx}^{-1} (x - \bar{x})
\]
Unscented Transformation

- Since $y = g(x)$, it suffices to specify the sample points $x_i, i = 1, \cdots, N$ to get data $(x_i, y_i), i = 1, \cdots, N$.
- Let $x \in \mathbb{R}^n$ be a random variable with mean $\mu$ and covariance matrix $\Sigma$.
- Define $2n + 1$ points and weighting coefficients: $(x_i, W_i), i = 0, 1, \cdots, 2n$ satisfying

$$\bar{x} = \sum_{i=0}^{2n} W_i x_i = \mu, \quad S_{xx} = \sum_{i=0}^{2n} W_i [x_i - \mu][x_i - \mu]^T = \Sigma$$

$(x_i, W_i), i = 0, 1, \cdots, 2n$ are called $\sigma$ points.
Unscented Transformation

\[
\begin{align*}
\text{mean } \bar{y} &= \sum_{i=0}^{2n} W_ig(X_i) \\
\text{cov } S_{yx} &= \sum_{i=0}^{2n} W_i[g(X_i) - \bar{y}][X_i - \bar{x}]^T \\
S_{yy} &= \sum_{i=0}^{2n} W_i[g(X_i) - \bar{y}][g(X_i) - \bar{y}]^T
\end{align*}
\]
Unscented Transformation

Usually, $\sigma$ points are determined as

\[ X_0 = \mu, \quad W_0 = \frac{\lambda}{n + \lambda} \]

\[ X_i = \mu + \left[ \sqrt{(n + \lambda)\Sigma} \right]_i, \quad W_i = \frac{1}{2(n + \lambda)}, \quad i = 1, \cdots, n \]

\[ X_{i+n} = \mu - \left[ \sqrt{(n + \lambda)\Sigma} \right]_i, \quad W_{i+n} = \frac{1}{2(n + \lambda)}, \quad i = 1, \cdots, n \]

$[\sqrt{\cdot}]_i$: the $i$th column vector of matrix square root

UKF algorithm

1. Let $\hat{x}_{0|0} = \bar{x}_0$ and $P_{0|0} = \Sigma_0$, and generate the initial $\sigma$ points $(i = 1, \cdots, n)$

$$\hat{x}^{(0)}_{0|0} = \hat{x}_{0|0}, \quad \hat{x}^{(i)}_{0|0} = \hat{x}_{0|0} + \left[\sqrt{(n + \lambda)P_{0|0}}\right]_i$$

$$\hat{x}^{(i+n)}_{0|0} = \hat{x}_{0|0} - \left[\sqrt{(n + \lambda)P_{0|0}}\right]_i$$

Set $t = 0$.

2. Observation update

**Input:** $[\hat{x}^{(i)}_{t|t-1}, P_{t|t-1}, y_t] \rightarrow \textbf{Output:} [\hat{x}_{t|t}, P_{t|t}]$

(a) Predicted estimate of output

$$\hat{y}_{t|t-1} = \sum_{i=0}^{2n} W_h^{(i)} h_1(\hat{x}^{(i)}_{t|t-1})$$
(b) Conditional covariances

\[ V_{t|t-1} = \sum_{i=0}^{2n} W_h^{(i)} [h_t(\hat{x}_{t|t-1}^{(i)}) - \hat{y}_{t|t-1}][h_t(\hat{x}_{t|t-1}^{(i)}) - \hat{y}_{t|t-1}]^T + R_t \]

\[ U_{t|t-1} = \sum_{i=0}^{2n} W_h^{(i)} [(\hat{x}_{t|t-1}^{(i)} - \hat{x}_{t|t-1})[h_t(\hat{x}_{t|t-1}^{(i)}) - \hat{y}_{t|t-1}]^T \]

(c) UKF gain \( K_t = U_{t|t-1} V_{t|t-1}^{-1} \)

(d) Filtered estimate

\[ \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t[y_t - \hat{y}_{t|t-1}] \]

(e) Filtered covariance matrix

\[ P_{t|t} = P_{t|t-1} - U_{t|t-1} V_{t|t-1}^{-1} U_{t|t-1}^T \]

3. Time update

Input: \([\hat{x}_{t|t}, P_{t|t}] \) → Output: \([\hat{x}_{t+1|t}^{(i)}, P_{t+1|t}] \)
(a) Generation of σ points \((i = 1, \cdots, n)\)
\[
\hat{x}_{i|t}^{(0)} = \hat{x}_{t|t}, \quad \hat{x}_{i|t}^{(i)} = \hat{x}_{t|t} + \left[ \sqrt{(n + \lambda)P_{t|t}} \right]_i, \quad \hat{x}_{i|t}^{(i+n)} = \hat{x}_{t|t} - \left[ \sqrt{(n + \lambda)P_{t|t}} \right]_i
\]

(b) Transformation of σ points
\[
\hat{x}_{t+1|t}^{(i)} = f_t(\hat{x}_{t|t}^{(i)}), \quad i = 0, 1, \cdots, 2n
\]

(c) Predicted estimate
\[
\hat{x}_{t+1|t} = \sum_{i=0}^{2n} W_f^{(i)} f_t(\hat{x}_{t|t}^{(i)})
\]

(d) Predicted covariance matrix
\[
P_{t+1|t} = \sum_{i=0}^{2n} W_f^{(i)} [f_t(\hat{x}_{t|t}^{(i)}) - \hat{x}_{t+1|t}][f_t(\hat{x}_{t|t}^{(i)}) - \hat{x}_{t+1|t}]^T + Q_t
\]

4. Put \(t := t + 1\), and go to Step 2.
Classical Ballistic Missile Model

Consider the problem of tracking a body falling freely through the atmosphere (Gelb 1974).

\[ \begin{align*}
\dot{x}_1 &= x_2, & \dot{x}_2 &= d - g, & \dot{x}_3 &= 0, & y &= x_1 + v \\
\beta &= x_3 \\
d &= \frac{\rho_0 e^{-x_1/k_\rho}}{2\beta} x_2^2
\end{align*} \]

\[ x_1 = \text{height of the missle} \quad \quad x_2 = \text{velocity} \]
\[ g = \text{acceleration of gravity} \quad \quad \beta = \text{ballistic coefficient} \]
\[ d = \text{drag deceleration} \quad \quad k_\rho = \text{decay constant} \]
\[ \rho_0 = \text{atomsheric density at sea level} \]
• Data for simulation

\( g = 9.8, \ \rho_0 = 2.202, \ k_\rho = 1000/0.1558, \ \beta = 2000 \)
\( \Delta = 0.1, \ N = 300 \)

\[
\begin{bmatrix}
  x_1(0) \\
  x_2(0) \\
  x_3(0)
\end{bmatrix} =
\begin{bmatrix}
  40000 \\
  -3000 \\
  2000
\end{bmatrix}, \quad
Q =
\begin{bmatrix}
  0 & 0 & 0 \\
  0 & 0 & 0 \\
  0 & 0 & 10
\end{bmatrix}, \quad
R = 100
\]

\[
x(0|0) =
\begin{bmatrix}
  40100 \\
  -3100 \\
  3000
\end{bmatrix}, \quad
P(0|0) =
\begin{bmatrix}
  100 & 0 & 0 \\
  0 & 10000 & 0 \\
  0 & 0 & 10000
\end{bmatrix}
\]

• 4th-order Runge-Kutta is used for simulation.
Estimation of parameter $\beta$ (Chol - SVD)
Estimation of parameter $\beta$ (Chol - SVD)
Comments

- Motivated by the encouraging results for a ballistic model, we have started a study of UT through examples for static nonlinearities and then simulated several dynamical models.

- We have used SVD and QR decomposition rather than Cholesky decomposition for matrix factorizations. Reading many papers for UKF, however, we noticed that they are based on mostly Cholesky factorization.
### UT: Example n=1

\[ x \sim N(\mu, \sigma^2) \quad x \rightarrow g(\cdot) \quad y \]

<table>
<thead>
<tr>
<th>( y = x^2 )</th>
<th>( \mu_y )</th>
<th>( \sigma_{yx} )</th>
<th>( \sigma_y^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True</strong></td>
<td>( \mu^2 + \sigma^2 )</td>
<td>( 2\mu\sigma^2 )</td>
<td>( 4\mu^2\sigma^2 + 2\sigma^4 )</td>
</tr>
<tr>
<td><strong>Linear</strong></td>
<td>( \mu^2 )</td>
<td>( 2\mu\sigma^2 )</td>
<td>( 4\mu^2\sigma^2 )</td>
</tr>
<tr>
<td><strong>UT</strong></td>
<td>( \mu^2 + \sigma^2 )</td>
<td>( 2\mu\sigma^2 )</td>
<td>( 4\mu^2\sigma^2 + \lambda\sigma^4 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( y = x^3 )</th>
<th>( \mu_y )</th>
<th>( \sigma_{yx} )</th>
<th>( \sigma_y^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True</strong></td>
<td>( \mu^3 + 3\mu\sigma^2 )</td>
<td>( 3\mu^2\sigma^2 + 3\sigma^4 )</td>
<td>( 9\mu^4\sigma^2 + 36\mu^2\sigma^4 + 15\sigma^6 )</td>
</tr>
<tr>
<td><strong>Linear</strong></td>
<td>( \mu^3 )</td>
<td>( 3\mu^2\sigma^2 )</td>
<td>( 9\mu^4\sigma^2 )</td>
</tr>
<tr>
<td><strong>UT</strong></td>
<td>( \mu^3 + 3\mu\sigma^2 )</td>
<td>( 3\mu^2\sigma^2 + (1 + \lambda)\sigma^4 )</td>
<td>( 9\mu^4\sigma^2 + (6 + 15\lambda)\mu^2\sigma^4 + (1 + \lambda)^2\sigma^6 )</td>
</tr>
</tbody>
</table>
UT: Cholesky \ n=2

\[
\Sigma = LL^T = \begin{bmatrix}
  * & 0 \\
  * & * \\
\end{bmatrix} \begin{bmatrix}
  * & * \\
  0 & * \\
\end{bmatrix}
\]

\[
L = \begin{bmatrix}
  l_{11} & 0 \\
  l_{21} & l_{22} \\
\end{bmatrix}
= \begin{bmatrix}
  l_1 \\
  l_2 \\
\end{bmatrix}
\]

\[
X_0 = \mu_x
\]

\[
X_i = \mu_x \pm (n + \lambda)^{1/2} l_i
\]

\[
(x - \mu)^T \Sigma^{-1} (x - \mu) = C
\]

\[
(C = 1, \mu_1 = 5, \mu_2 = 2)
\]

Elipsoid of constant density
$\Sigma = USU^T = U \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} U^T$

$U^s = U \begin{bmatrix} \sqrt{s_1} & 0 \\ 0 & \sqrt{s_2} \end{bmatrix} = [\sqrt{s_1}u_1 \quad \sqrt{s_2}u_2]$

$X_0 = \mu_x$

$X_i = \mu_x \pm (n + \lambda)^{1/2} \sqrt{s_i}u_i$

$(x - \mu)^T \Sigma^{-1} (x - \mu) = C$

$(C = 1, \mu_1 = 5, \mu_2 = 2)$

Elipsoid of constant density
UT: Example n=2

\[ y_1 = x_1 - x_1 x_2 \]  \hspace{1cm} \text{(Lotka-Volterra)}
\[ y_2 = -0.8x_2 + 1.2x_1 x_2 \]

\[ \mu_x = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma_x = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix} \]

\[ \sigma_1^2 = \sigma_2^2 = 1, \quad \sigma_{12} = 0.5 \]

We compute the 1st- and 2nd-order moments of the output \( y \), where the true, the linear approximation and the UT transformation are shown.
### UT: Example n=2

#### Means of $y_1$ and $y_2$

<table>
<thead>
<tr>
<th></th>
<th>$\mu_1 = 0$</th>
<th>$\mu_1 = 1$</th>
<th>$\mu_1 = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_y_1$</td>
<td>$\mu_2 = 0$</td>
<td>$\mu_2 = 1$</td>
<td>$\mu_2 = 2$</td>
</tr>
<tr>
<td>True</td>
<td>$-0.5$</td>
<td>$-0.5$</td>
<td>$-2.5$</td>
</tr>
<tr>
<td>Linear</td>
<td>$0.0$</td>
<td>$0.0$</td>
<td>$-2.0$</td>
</tr>
<tr>
<td>Chol</td>
<td>$-0.4330$</td>
<td>$-0.4330$</td>
<td>$-2.4330$</td>
</tr>
<tr>
<td>SVD</td>
<td>$-0.5000$</td>
<td>$-0.5000$</td>
<td>$-2.5000$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\mu_1 = 0$</th>
<th>$\mu_1 = 1$</th>
<th>$\mu_1 = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_y_2$</td>
<td>$\mu_2 = 0$</td>
<td>$\mu_2 = 1$</td>
<td>$\mu_2 = 2$</td>
</tr>
<tr>
<td>True</td>
<td>$0.6$</td>
<td>$1.0$</td>
<td>$3.8$</td>
</tr>
<tr>
<td>Linear</td>
<td>$0.0$</td>
<td>$0.4$</td>
<td>$3.2$</td>
</tr>
<tr>
<td>Chol</td>
<td>$0.5196$</td>
<td>$0.9196$</td>
<td>$3.7196$</td>
</tr>
<tr>
<td>SVD</td>
<td>$0.6000$</td>
<td>$1.0000$</td>
<td>$3.8000$</td>
</tr>
<tr>
<td></td>
<td>Var($y_1$)</td>
<td>( \mu_1 = 0 )</td>
<td>( \mu_1 = 1 )</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>True</strong> ( \sigma^2_{y_1} )</td>
<td></td>
<td>2.25</td>
<td>2.25</td>
</tr>
<tr>
<td><strong>Linear</strong> ( \sigma^2_{y_1L} )</td>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>( \lambda_c = 4.5 )</td>
<td>2.2812</td>
<td>1.7812</td>
<td>7.0133</td>
</tr>
<tr>
<td>( \lambda_c = 6.0 )</td>
<td>2.5625</td>
<td>2.0625</td>
<td>7.2946</td>
</tr>
<tr>
<td>( \lambda_c = 7.0 )</td>
<td>2.7500</td>
<td>2.2500</td>
<td>7.4821</td>
</tr>
<tr>
<td>( \lambda_c = 8.0 )</td>
<td>2.9375</td>
<td>2.4375</td>
<td>7.6696</td>
</tr>
<tr>
<td>( \lambda_c = 11.0 )</td>
<td>3.5000</td>
<td>3.0000</td>
<td>8.2321</td>
</tr>
<tr>
<td>( \lambda_s = 0.30 )</td>
<td>2.1875</td>
<td>2.1875</td>
<td>8.1875</td>
</tr>
<tr>
<td>( \lambda_s = 0.35 )</td>
<td>2.2188</td>
<td>2.2188</td>
<td>8.2188</td>
</tr>
<tr>
<td>( \lambda_s = 0.40 )</td>
<td>2.2500</td>
<td>2.2500</td>
<td>8.2500</td>
</tr>
<tr>
<td>( \lambda_s = 0.45 )</td>
<td>2.2812</td>
<td>2.2812</td>
<td>8.2812</td>
</tr>
<tr>
<td></td>
<td>$\mu_1 = 0$</td>
<td>$\mu_1 = 1$</td>
<td>$\mu_1 = 2$</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>$\mu_2 = 0$</td>
<td>$\mu_2 = 1$</td>
<td>$\mu_2 = 2$</td>
</tr>
<tr>
<td><strong>True $\sigma^2_{y_2}$</strong></td>
<td>2.44</td>
<td>3.88</td>
<td>13.96</td>
</tr>
<tr>
<td><strong>Linear $\sigma^2_{y_2L}$</strong></td>
<td>0.64</td>
<td>2.08</td>
<td>12.16</td>
</tr>
<tr>
<td>$\lambda_c = 3.5$</td>
<td>1.6950</td>
<td>3.5507</td>
<td>13.6605</td>
</tr>
<tr>
<td>$\lambda_c = 4.5$</td>
<td>1.9650</td>
<td>3.8207</td>
<td>13.9305</td>
</tr>
<tr>
<td>$\lambda_c = 5.5$</td>
<td>2.2350</td>
<td>4.0907</td>
<td>14.2005</td>
</tr>
<tr>
<td>$\lambda_c = 6.5$</td>
<td>2.5050</td>
<td>4.3607</td>
<td>14.4705</td>
</tr>
<tr>
<td>$\lambda_s = 0.30$</td>
<td>2.3500</td>
<td>3.7900</td>
<td>13.8700</td>
</tr>
<tr>
<td>$\lambda_s = 0.35$</td>
<td>2.3950</td>
<td>3.8350</td>
<td>13.9150</td>
</tr>
<tr>
<td>$\lambda_s = 0.40$</td>
<td>2.4400</td>
<td>3.8800</td>
<td>13.9600</td>
</tr>
<tr>
<td>$\lambda_s = 0.45$</td>
<td>2.4850</td>
<td>3.9250</td>
<td>14.0050</td>
</tr>
<tr>
<td></td>
<td>( \mu_1 = 0 )</td>
<td>( \mu_1 = 1 )</td>
<td>( \mu_1 = 2 )</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>True ( \sigma_{y_{12}} )</td>
<td>−1.9</td>
<td>−2.5</td>
<td>−10.3</td>
</tr>
<tr>
<td>Linear ( \sigma_{y_{12L}} )</td>
<td>−0.4</td>
<td>−1.0</td>
<td>−8.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \lambda_c = 6.0 )</th>
<th>( \lambda_c = 6.5 )</th>
<th>( \lambda_c = 7.0 )</th>
<th>( \lambda_c = 8.5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chol</td>
<td>−1.9214</td>
<td>−2.3946</td>
<td>−9.7463</td>
</tr>
<tr>
<td></td>
<td>−2.0339</td>
<td>−2.5071</td>
<td>−9.8588</td>
</tr>
<tr>
<td></td>
<td>−2.1464</td>
<td>−2.6196</td>
<td>−9.9713</td>
</tr>
<tr>
<td></td>
<td>−2.4839</td>
<td>−2.9571</td>
<td>−10.3088</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \lambda_s = 0.30 )</th>
<th>( \lambda_s = 0.35 )</th>
<th>( \lambda_s = 0.40 )</th>
<th>( \lambda_s = 0.45 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>−1.8250</td>
<td>−2.4250</td>
<td>−10.2250</td>
</tr>
<tr>
<td></td>
<td>−1.8625</td>
<td>−2.4625</td>
<td>−10.2625</td>
</tr>
<tr>
<td></td>
<td>−1.9000</td>
<td>−2.5000</td>
<td>−10.3000</td>
</tr>
<tr>
<td></td>
<td>−1.9375</td>
<td>−2.5375</td>
<td>−10.3375</td>
</tr>
</tbody>
</table>
Comments

• Numerical results for static nonlinearities above shows that the SVD-based UT method is better than Cholesky-based one.

• We show some more examples for dynamic systems. Computational load for SVD-based method is 10% higher than Cholesky decomposition-based method in MATLAB.
Lotka-Volterra Model

Consider the state and parameter estimation problem for Lotka-Volterra model

\[
\begin{align*}
\dot{x}_1 &= r_1 x_1 - a_{12} x_1 x_2 + w_1 \\
\dot{x}_2 &= -r_2 x_2 + a_{21} x_1 x_2 + w_2 \\
y_1 &= x_1 + v_1 \\
y_2 &= x_2 + v_2 \\
x_1 &= \text{number of prey} \\
r_1 &= \text{coefficient of growth rate} \\
r_2 &= \text{coefficient of death rate} \\
a_{12}, a_{21} &= \text{coefficients of interaction}
\end{align*}
\]
• Data for simulation

\[ a_{12} = 1, \ a_{21} = 1, \ r_1 = 0.8, \ r_2 = 1.2 \]
\[ x_3 = r_1, \ x_4 = r_2, \ \Delta = 0.01, \ N = 3000 \]
\[ Q = 10^{-5}I_4, \ R = 0.05I_2 \]

\[
x(0) = \begin{bmatrix}
10^{-2} \\
10^{-2} \\
10^{-2} \\
10^{-2}
\end{bmatrix}, \quad P(0) = \begin{bmatrix}
1.0 & 0 & 0 & 0 \\
0 & 1.0 & 0 & 0 \\
0 & 0 & 0.1 & 0 \\
0 & 0 & 0 & 0.1
\end{bmatrix}
\]

• 4th-order Runge-Kutta is used for simulation.
True states (top) and estimates (bottom) by Cholesky
True states (top) and estimates (bottom) by SVD
State estimation errors

Mean square error of states

MSE of states

Time

Chol
SVD

0 5 10 15 20 25 30

0 0.5 1 1.5 2
Parameter estimation by SVD & Cho

Parameter $r$

![Graph 1](#)

Parameter $r$

![Graph 2](#)

Estimates

Number of steps $k$
Models for Simulation

1. 1st-order linear system
   \[ x_{t+1} = ax_t + bu_t + w_t \]
   \[ y_t = x_t + v_t \]

2. 1st-order NL system
   \[ \dot{x} = \gamma(u - ax + b) \]
   \[ y = x + \sin x + v \]

3. Ballistic missile model

4. Lotka-Volterra model

5. Lorenz model
   \[ \dot{x}_1 = \sigma(x_2 - x_1) + w_1 \]
   \[ \dot{x}_2 = \rho x_1 - x_2 - x_1 x_3 + w_2 \]
   \[ \dot{x}_3 = x_1 x_2 - \beta x_3 + w_3 \]
   \[ y = I_3 x + v \]

6. Two-dim tracking model
   \[ \dot{x}_1 = x_2 + w_1 \]
   \[ \dot{x}_2 = -x_1 + w_2 \]
   \[ y_1 = \sqrt{(x_1 - \alpha)^2 + x_2^2} + v_1 \]
   \[ y_2 = \tan^{-1} \left( \frac{x_2}{x_1 - \alpha} \right) + v_2 \]
## Estimation Results for Six Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>EKF</th>
<th>UKF&lt;sub&gt;s&lt;/sub&gt;</th>
<th>UKF&lt;sub&gt;c&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(a)</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>(b)</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>(a)</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>(b)</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>3</td>
<td>(\beta)</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>4</td>
<td>(r_1, r_2)</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>(a_{12}, a_{21})</td>
<td>○</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>5</td>
<td>(\sigma, \rho, \beta)</td>
<td>×</td>
<td>○</td>
<td>×</td>
</tr>
<tr>
<td>6</td>
<td>—</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Euler</th>
<th>R-K</th>
<th>R-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
Two-dim Tracking Model

\[ \dot{x}_1 = x_2 + w_1, \quad y_1 = r = \sqrt{(x_1 - \alpha)^2 + x_2^2} + v_1 \]

\[ \dot{x}_2 = -x_1 + w_2, \quad y_2 = \theta = \tan^{-1} \left( \frac{x_2}{x_1 - \alpha} \right) + v_2 \]

\[ \dot{x}_3 = 0, \quad (x_3 = \alpha) \]
• **Data for simulation**

\[
\begin{bmatrix}
    x_1(0) \\
    x_2(0) \\
    x_3(0)
\end{bmatrix} =
\begin{bmatrix}
    1 \\
    1 \\
    4 (= \alpha)
\end{bmatrix}, \quad Q =
\begin{bmatrix}
    10^{-6} & 0 & 0 \\
    0 & 10^{-6} & 0 \\
    0 & 0 & 0
\end{bmatrix}
\]

\[R = 10^{-2} I_2\]

\[
\hat{x}(0) =
\begin{bmatrix}
    10^{-1} \\
    10^{-1} \\
    10^{-1}
\end{bmatrix}, \quad P(0) =
\begin{bmatrix}
    1.0 & 0 & 0 \\
    0 & 1.0 & 0 \\
    0 & 0 & 2.0
\end{bmatrix}
\]

• **4th-order Runge-Kutta is used for simulation.**

• **T = 15 (sec), \Delta = 0.01, \lambda = 1**
Two-dim Tracking Model

True trajectory

Estimates
Two-dim Tracking Model

State vector (true)

Number of steps k

State vector (estimation)

Number of steps k

True states

Estimates
Two-dim Tracking Model
Conclusions

- As shown in examples of UT for static nonlinear transformation as well as for many nonlinear dynamical systems, the SVD-based method shows better performance than Cholesky-based method.

- Computational load for SVD is 10% higher than that for Cholesky decomposition in MATLAB.

- It is however not easy to give a recipe for selecting a suitable value for the parameter $\lambda$. 