TSRT14: Sensor Fusion
Lecture 6
Kalman filter (KF).
KF approximations (EKF, UKF)

Gustaf Hendeby

Spring 2015
Whiteboard:
- Derivation framework for KF, EKF, UKF

Slides:
- Kalman filter summary: main equations, robustness, sensitivity, divergence monitoring, user aspects
- Nonlinear transforms revisited
- Application to derivation of EKF and UKF
- User guidelines and interpretations
Standard models in global coordinates:

- Translation \( p_t^{(m)} = w_t^p \)
- 2D orientation for heading \( h_t^{(m)} = w_t^h \)
- Coordinated turn model

\[
\begin{align*}
\dot{X} &= v^X \\
\dot{v}^X &= -\omega v^Y \\
\dot{\omega} &= 0
\end{align*}
\]

Standard models in local coordinates \((x, y, \psi)\)

- Odometry and dead reckoning for \((x, y, \psi)\)

\[
\begin{align*}
X_t &= X_0 + \int_0^t v^x_t \cos(\psi_t) \, dt \\
Y_t &= Y_0 + \int_0^t v^y_t \sin(\psi_t) \, dt \\
\psi_t &= \psi_0 + \int_0^t \dot{\psi}_t \, dt
\end{align*}
\]

- Force models for \((\dot{\psi}, a_y, a_x, \ldots)\)
- 3D orientation \( \dot{q} = \frac{1}{2} \mathcal{S}(\omega)q = \frac{1}{2} \mathcal{S}(q)\omega \)
Chapter 7 Overview

Kalman filter

- Algorithms and derivation
- Practical issues
- Computational aspects
- Filter monitoring

The discussion and conclusions do usually apply to all nonlinear filters, though it is more concrete in the linear Gaussian case.
Kalman Filter (KF)

Time-varying state space model:

\[
x_{k+1} = F_k x_k + G_k v_k, \quad \text{cov}(v_k) = Q_k
\]
\[
y_k = H_k x_k + e_k, \quad \text{cov}(e_k) = R_k
\]

Time update:

\[
\hat{x}_{k+1|k} = F_k \hat{x}_{k|k}
\]
\[
P_{k+1|k} = F_k P_{k|k} F_k^T + G_k Q_k G_k^T
\]

Measurement update:

\[
\hat{x}_{k|k} = \hat{x}_{k|k-1} + P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} (y_k - H_k \hat{x}_{k|k-1})
\]
\[
P_{k|k} = P_{k|k-1} - P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} H_k P_{k|k-1}.
\]
KF Modifications

Auxiliary quantities: innovation $\epsilon_k$, innovation covariance $S_k$ and Kalman gain $K_k$

\[
\hat{y}_k = H_k \hat{x}_{k|k-1}
\]
\[
\epsilon_k = y_k - H_k \hat{x}_{k|k-1} = y_k - \hat{y}_k
\]
\[
S_k = H_k P_{k|k-1} H_k^T + R_k
\]
\[
K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} = P_{k|k-1} H_k^T S_k^{-1}
\]

Filter form

\[
\hat{x}_{k|k} = F_{k-1} \hat{x}_{k-1|k-1} + K_k (y_k - H_k F_{k-1} \hat{x}_{k-1|k-1})
\]
\[
= (F_{k-1} - K_k H_k F_{k-1}) \hat{x}_{k-1|k-1} + K_k y_k,
\]

Predictor form

\[
\hat{x}_{k+1|k} = F_k \hat{x}_{k|k-1} + F_k K_k (y_k - H_k \hat{x}_{k|k-1})
\]
\[
= (F_k - F_k K_k H_k) \hat{x}_{k|k-1} + F_k K_k y_k
\]
Create a constant velocity model, simulate and Kalman filter

```matlab
T = 0.5;
F = [1 0 T 0; 0 1 0 T; 0 0 1 0; 0 0 0 1];
G = [T^2/2 0; 0 T^2/2; T 0; 0 T];
H = [1 0 0 0; 0 1 0 0];
R = 0.03*eye(2);
m = lss(F, [], H, [], G*G', R, 1/T);
m.xlabel = {'X', 'Y', 'vX', 'vY'};
m.ylabel = {'X', 'Y'};
m.name = 'Constant velocity motion model';
z = simulate(m, 20);
xhat1 = kalman(m, z, 'alg', 2);
xplot2(z, xhat1, 'conf', 90, [1 2]);
```
Simulation Example (2/2)

Covariance illustrated as confidence ellipsoids in 2D plots or confidence bands in 1D plots.

```matlab
xplot(z, xhat1, 'conf', 99)
```
The SNR ratio $\|Q\|/\|R\|$ is the most crucial, it sets the filter speeds. Note difference of real system and model used in the KF.

Recomentation: fix $R$ according to sensor specification/performance, and tune $Q$ (motion models are anyway subjective approximation of reality).

High SNR in the model, gives fast filter that is quick in adapting to changes/maneuvers, but with larger uncertainty (small bias, large variance).

Conversely, low SNR in the model, gives slow filter that is slow in adapting to changes/maneuvers, but with small uncertainty (large bias, small variance).

$P_0$ reflects the belief in the prior $x_1 \sim \mathcal{N}(\hat{x}_{1|0}, P_0)$. Possible to choose $P_0$ very large (and $\hat{x}_{1|0}$ arbitrary), if no prior information exists.

Tune covariances in large steps (order of magnitudes)!
Optimality Properties

- For a linear model, the KF provides the WLS solution
- The KF is the best linear unbiased estimator (BLUE)
- It is the Bayes optimal filter for linear model when $x_0, v_k, e_k$ are Gaussian variables,

\[
\begin{align*}
    x_{k+1|y_{1:k}} & \sim \mathcal{N}(\hat{x}_{k+1|k}, P_{k+1|k}) \\
    x_k|y_{1:k} & \sim \mathcal{N}(\hat{x}_{k|k}, P_k|k) \\
    \varepsilon_k & \sim \mathcal{N}(0, S_k)
\end{align*}
\]
Robustness and Sensitivity

The following concepts are relevant for all filtering applications, but they are most explicit for KF

- **Observability** is revealed indirectly by $P_{k|k}$; monitor its rank or better condition number.
- **Divergence tests** Monitor performance measures and restart the filter after divergence.
- **Outlier rejection** monitor sensor observations.
- **Bias error** incorrect model gives bias in estimates.
- **Sensitivity analysis** uncertain model contributes to the total covariance.
- **Numerical issues** may give complex estimates.
1. Snapshot observability if $H_k$ has full rank. WLS can be applied to estimate $x$

2. Classical observability for time-invariant and time/varying case,

$$O = \begin{pmatrix} H \\ HF \\ HF^2 \\ \vdots \\ HF^{n-1} \end{pmatrix}$$

$$O_k = \begin{pmatrix} H_{k-n+1} \\ H_{k-n+2}F_{k-n+1} \\ H_{k-n+3}F_{k-n+2}F_{k-n+1} \\ \vdots \\ H_kF_{k-1} \cdots F_{k-n+1} \end{pmatrix}$$

3. The covariance matrix $P_{k|k}$ extends the observability condition by weighting with the measurement noise and to forget old information according to the process noise. Thus, (the condition number of) $P_{k|k}$ is the natural indicator of observability!
Divergence Tests

When is $\varepsilon_k \varepsilon_k^T$ significantly larger than its computed expected value $S_k = E(\varepsilon_k \varepsilon_k^T)$ (note that $\varepsilon_k \sim \mathcal{N}(0, S_k)$)?

Principal reasons:

- Model errors
- Sensor model errors: offsets, drifts, incorrect covariances, scaling factor in all covariances
- Sensor errors: outliers, missing data
- Numerical issues

Solutions:

- In the first two cases, the filter has to be redesigned
- In the last two cases, the filter has to be restarted
Outlier Rejection

Let $H_0 : \varepsilon_k \sim \mathcal{N}(0, S_k)$, then

$$T(y_k) = \varepsilon_k^T S_k^{-1} \varepsilon_k \sim \chi^2_{n_yk}$$

if everything works fine, and there is no outlier. If $T(y_k) > h_{P_{FA}}$, this is an indication of outlier, and the measurement update can be omitted.

In the case of several sensors, each sensor $i$ should be monitored for outliers

$$T(y_k^i) = (\varepsilon_k^i)^T S_k^{-1} \varepsilon_k^i \sim \chi^2_{n_{y_k^i}}$$
Sensitivity analysis: parameter uncertainty

**Sensitivity analysis** can be done with respect to uncertain parameters with known covariance matrix using for instance Gauss approximation formula.

- Assume $F(\theta)$, $G(\theta)$, $H(\theta)$, $Q(\theta)$, $R(\theta)$ have uncertain parameters $\theta$ with $E(\theta) = \hat{\theta}$ and $\text{cov}(\theta) = P_\theta$.
- The state estimate $\hat{x}_k$ is a stochastic variable with four stochastic sources, $v_k, e_k, x_1$ at one hand, and $\theta$ on the other hand.
- The law of total variance ($\text{var}(X) = E\text{var}(X|Y) + \text{var} E(X|Y)$) and Gauss approximation formula ($\text{var}(h(Y)) \approx h'_Y(\bar{Y}) \text{var}(Y)(h'_Y(\bar{Y}))^T$) gives

$$\text{cov}(\hat{x}_{k|k}) \approx P_{k|k} + \frac{d\hat{x}_{k|k}}{d\theta} P_\theta \left( \frac{d\hat{x}_{k|k}}{d\theta} \right)^T.$$ 

- The gradient $d\hat{x}_{k|k}/d\theta$ can be computed numerically by simulations.
Numerical Issues

Some simple fixes if problem occurs:

- Assure that the covariance matrix is symmetric $P = 0.5P + 0.5P'$
- Use the more numerically stable Joseph's form for the measurement update of the covariance matrix:

$$P_{k|k} = (I - K_k H_k)P_{k|k-1}(I - K_k H_k)^T + K_k R_k K_k^T$$

- Assure that the covariance matrix is positive definite by setting negative eigenvalues in $P$ to zero or small positive values
- Avoid singular $R = 0$, even for constraints
- Dithering. Increase $Q$ and $R$ if needed; this can account for all kind of model errors
Chapter 8 Overview

- Nonlinear transformations
- Details of the EKF algorithms
- Numerical methods to compute Jacobian and Hessian in the Taylor expansion
- An alternative EKF version without the Ricatti equation
- The unscented Kalman filter (UKF)
Apply TT1 and TT2, respectively, to the dynamic and observation models. For instance,

\[ x_{k+1} = f(x_k) + v_k = f(\hat{x}) + g'(\hat{x})(x - \hat{x}) + \frac{1}{2}(x - \hat{x})^T g''(\xi)(x - \hat{x}). \]

- EKF1 neglects the rest term.
- EKF2 compensates with the mean and covariance of the rest term using \( \xi = \hat{x} \).
EKF1 and EKF2 Algorithm

\[ S_k = h'_x(\hat{x}_{k|k-1})P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T + h'_e(\hat{x}_{k|k-1})R_k(h'_e(\hat{x}_{k|k-1}))^T \]

\[ + \frac{1}{2} \left[ \text{tr}(h''_{i,x}(\hat{x}_{k|k-1})P_{k|k-1}h''_{j,x}(\hat{x}_{k|k-1})P_{k|k-1}) \right]_{ij} \]

\[ K_k = P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T S_k^{-1} \]

\[ \varepsilon_k = y_k - h(\hat{x}_{k|k-1}, 0) - \frac{1}{2} \left[ \text{tr}(h''_{i,x} P_{k|k-1}) \right]_i \]

\[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \varepsilon_k \]

\[ P_{k|k} = P_{k|k-1} - P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T S_k^{-1} h'_x(\hat{x}_{k|k-1})P_{k|k-1} \]

\[ + \frac{1}{2} \left[ \text{tr}(h''_{i,x}(\hat{x}_{k|k-1})P_{k|k-1}h''_{j,x}(\hat{x}_{k|k-1})P_{k|k-1}) \right]_{ij} \]

\[ \hat{x}_{k+1|k} = f(\hat{x}_{k|k}, 0) + \frac{1}{2} \left[ \text{tr}(f''_{i,x} P_{k|k}) \right]_i \]

\[ P_{k+1|k} = f'_x(\hat{x}_{k|k})P_{k|k}(f'_x(\hat{x}_{k|k}))^T + f'_v(\hat{x}_{k|k})Q_k(f'_v(\hat{x}_{k|k}))^T \]

\[ + \frac{1}{2} \left[ \text{tr}(f''_{i,x}(\hat{x}_{k|k})P_{k|k}f''_{j,x}(\hat{x}_{k|k})P_{k|k}) \right]_{ij} \]

NB!

This form of the EKF2 (as given in the book) is disregarding second order terms of the process noise! See, e.g., my thesis for the full expressions.
The EKF1, using the TT1 transformation, is obtained by letting both Hessians $f''_x$ and $h''_x$ be zero.

- Analytic Jacobian and Hessian needed. If not available, use numerical approximations (done in Signal and Systems Lab by default!)

- The complexity of EKF1 is as in KF $n_x^3$ due to the $FPF^T$ operation.

- The complexity of EKF2 is $n_x^5$ due to the $F_iPF_j^T$ operation for $i, j = 1, \ldots, n_x$.

- Dithering is good! That is, increase $Q$ and $R$ from the simulated values to account for the approximation errors.
EKF Variants

- The standard EKF linearizes around the current state estimate
- The *linearized Kalman filter* linearizes around some reference trajectory
- The *error state Kalman filter*, also known as the *complementary Kalman filter*, estimates the state error \( \tilde{x}_k = x_k - \hat{x}_k \) with respect to some approximate or reference trajectory. Feedforward or feedback configurations

linearized Kalman filter = feedforward error state Kalman filter
EKF = feedback error state Kalman filter
Numeric derivatives are preferred in the following cases:

- The nonlinear function is too complex
- The derivatives are too complex functions
- A user-friendly algorithm is desired, with as few user inputs as possible

This can be achieved with either numerical approximation or using sigma points!
Nonlinear transformations (NLT)

Consider a second order Taylor expansion of a function $z = g(x)$:

$$z = g(x) = g(\hat{x}) + g'(\hat{x})(x - \hat{x}) + \frac{1}{2}(x - \hat{x})^T g''(\xi)(x - \hat{x})$$

The rest term is negligible and EKF works fine if

- the model is almost linear
- or the SNR is high, so $\|x - \hat{x}\|$ can be considered small

The size of the rest term can be approximated \textit{a priori}.

\textbf{Note:} the size may depend on the choice of state coordinates!

If the rest term is large, use either of

- the second order compensated EKF that compensates for the mean and covariance of $r(x; \hat{x}, g''(\xi)) \approx r(x; \hat{x}, g''(\hat{x}))$
- the unscented KF (UKF)
The first order Taylor term gives a contribution to the covariance:

\[ x \sim \mathcal{N}(\hat{x}, P) \rightarrow \mathcal{N}(g(\hat{x}), [g'_i(\hat{x})P(g'_j(\hat{x}))^T]_{ij}) = \mathcal{N}(g(\hat{x}), g'(\hat{x})P(g'(\hat{x}))^T) \]

- This is sometimes called Gauss’ approximation formula.
- Here \([A]_{ij}\) means element \(i, j\) in the matrix \(A\). This is used in EKF1 (EKF with first order Taylor expansion). Leads to a KF where nonlinear functions are approximated with their Jacobians.
- Compare with the linear transformation rule

\[ z = Gx, \quad x \sim \mathcal{N}(\hat{x}, P) \quad \rightarrow \quad z \sim \mathcal{N}(G\hat{x}, GPG^T) \]

- Note that \(GPG^T\) can be written \([G_iPG_j^T]_{ij}\), where \(G_i\) denotes row \(i\) of \(G\).
The second order Taylor term contributes both to the mean and covariance as follows:

\[
x \sim \mathcal{N}(\hat{x}, P) \rightarrow \mathcal{N}(g(\hat{x}) + \frac{1}{2}[\text{tr}(g''(\hat{x})P)]_i, [g'_i(\hat{x})P(g'_j(\hat{x}))^T + \frac{1}{2} \text{tr}(Pg''(\hat{x})Pg''(\hat{x}))]_{ij})
\]

- This is used in EKF2 (EKF with second order Taylor expansion). Leads to a KF where nonlinear functions are approximated with their Jacobians and Hessians.
- UKF tries to do this approximation numerically, \textit{without} forming the Hessian \(g''(x)\) explicitly. This reduces the \(n_x^5\) complexity in \([\text{tr}(Pg''(\hat{x})Pg''(\hat{x}))]_{ij}\) to \(n_x^3\) complexity.
MC: Monte Carlo sampling

Generate $N$ samples, transform them, and fit a Gaussian distribution

$$x^{(i)} \sim \mathcal{N}(\hat{x}, P)$$

$$z^{(i)} = g(x^{(i)})$$

$$\mu_z = \frac{1}{N} \sum_{i=1}^{N} z^{(i)}$$

$$P_z = \frac{1}{N-1} \sum_{i=1}^{N} (z^{(i)} - \mu_z)(z^{(i)} - \mu_z)^T$$

Not commonly used in nonlinear filtering, but a valid and solid approach!
UT: the unscented transform

At first sight, similar to MC:
Generate $2n_x + 1$ sigma points, transform these, and fit a Gaussian distribution:

$$x^{(0)} = \hat{x}$$

$$x^{(\pm i)} = \hat{x} \pm \sqrt{n_x + \lambda} P^{1/2}_{\hat{x}, i}, \quad i = 1, 2, \ldots, n_x$$

$$z^{(i)} = g(x^{(i)})$$

$$E(z) \approx \frac{\lambda}{2(n_x + \lambda)} z^{(0)} + \sum_{i=-n_x}^{n_x} \frac{1}{2(n_x + \lambda)} z^{(i)}$$

$$\text{cov}(z) \approx \left( \frac{\lambda}{2(n_x + \lambda)} + (1 - \alpha^2 + \beta) \right) (z^{(0)} - E(z)) (z^{(0)} - E(z))^T + \sum_{i=-n_x}^{n_x} \frac{1}{2(n_x + \lambda)} (z^{(i)} - E(z)) (z^{(i)} - E(z))^T$$
UT: design parameters

- $\lambda$ is defined by $\lambda = \alpha^2 (n_x + \kappa) - n_x$
- $\alpha$ controls the spread of the sigma points and is suggested to be chosen around $10^{-3}$
- $\beta$ compensates for the distribution, and should be chosen to $\beta = 2$ for Gaussian distributions
- $\kappa$ is usually chosen to zero

Note

- $n_x + \lambda = \alpha^2 n_x$ when $\kappa = 0$
- The weights sum to one for the mean, but sum to $2 - \alpha^2 + \beta \approx 4$ for the covariance. Note also that the weights are not in $[0, 1]$
- The mean has a large negative weight!
- If $n_x + \lambda \to 0$, then UT and TT2 (and hence UKF and EKF2) are identical for $n_x = 1$, otherwise closely related!
Example 1: squared norm

Squared norm of a Gaussian vector has a known distribution:

\[ z = g(x) = x^T x, \quad x \sim \mathcal{N}(0, I_n) \Rightarrow z \sim \chi^2(n). \]

Theoretical distribution is \( \chi^2(n) \) with mean \( n \) and variance \( 2n \). The mean and variance are below summarized as a Gaussian distribution. The number of Monte Carlo simulations is 10 000.

<table>
<thead>
<tr>
<th>( n )</th>
<th>TT1</th>
<th>TT2</th>
<th>UT1</th>
<th>UT2</th>
<th>MCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \mathcal{N}(0, 0) )</td>
<td>( \mathcal{N}(1, 2) )</td>
<td>( \mathcal{N}(1, 2) )</td>
<td>( \mathcal{N}(1, 2) )</td>
<td>( \mathcal{N}(1.02, 2.15) )</td>
</tr>
<tr>
<td>2</td>
<td>( \mathcal{N}(0, 0) )</td>
<td>( \mathcal{N}(2, 4) )</td>
<td>( \mathcal{N}(2, 2) )</td>
<td>( \mathcal{N}(2, 8) )</td>
<td>( \mathcal{N}(2.02, 4.09) )</td>
</tr>
<tr>
<td>3</td>
<td>( \mathcal{N}(0, 0) )</td>
<td>( \mathcal{N}(3, 6) )</td>
<td>( \mathcal{N}(3, 0) )</td>
<td>( \mathcal{N}(3, 18) )</td>
<td>( \mathcal{N}(3.03, 6.3) )</td>
</tr>
<tr>
<td>4</td>
<td>( \mathcal{N}(0, 0) )</td>
<td>( \mathcal{N}(4, 8) )</td>
<td>( \mathcal{N}(4, -4) )</td>
<td>( \mathcal{N}(4, 32) )</td>
<td>( \mathcal{N}(4.03, 8.35) )</td>
</tr>
<tr>
<td>5</td>
<td>( \mathcal{N}(0, 0) )</td>
<td>( \mathcal{N}(5, 10) )</td>
<td>( \mathcal{N}(5, -10) )</td>
<td>( \mathcal{N}(5, 50) )</td>
<td>( \mathcal{N}(5.08, 10.4) )</td>
</tr>
<tr>
<td>Theory</td>
<td>( \mathcal{N}(0, 0) )</td>
<td>( \mathcal{N}(n, 2n) )</td>
<td>( \mathcal{N}(n, (3 - n)n) )</td>
<td>( \mathcal{N}(n, 2n^2) )</td>
<td>( \to \mathcal{N}(n, 2n) )</td>
</tr>
</tbody>
</table>

**Conclusion:** TT2 works, not the unscented transforms.
Example 2: radar

Conversion of polar measurements to Cartesian position:

\[ z = g(x) = \begin{pmatrix} x_1 \cos(x_2) \\ x_1 \sin(x_2) \end{pmatrix} \]

<table>
<thead>
<tr>
<th>X</th>
<th>TT1</th>
<th>TT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3.0)</td>
<td>(1.0 0.0)</td>
<td>(3.0)</td>
</tr>
<tr>
<td>(0.0)</td>
<td>(0.0 1.0)</td>
<td>(0.0)</td>
</tr>
<tr>
<td>(3.0)</td>
<td>(1.0 0.0)</td>
<td>(2.6)</td>
</tr>
<tr>
<td>(0.5)</td>
<td>(0.0 1.0)</td>
<td>(1.5)</td>
</tr>
<tr>
<td>(3.0)</td>
<td>(1.0 0.0)</td>
<td>(2.1)</td>
</tr>
<tr>
<td>(0.8)</td>
<td>(0.0 1.0)</td>
<td>(2.1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UT1</th>
<th>UT2</th>
<th>MCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.8)</td>
<td>(3.7 0.0)</td>
<td>(1.5)</td>
</tr>
<tr>
<td>(0.0)</td>
<td>(0.0 2.9)</td>
<td>(0.0)</td>
</tr>
<tr>
<td>(1.6)</td>
<td>(3.5 0.3)</td>
<td>(1.3)</td>
</tr>
<tr>
<td>(0.9)</td>
<td>(0.3 3.1)</td>
<td>(0.8)</td>
</tr>
<tr>
<td>(1.3)</td>
<td>(3.3 0.4)</td>
<td>(1.1)</td>
</tr>
<tr>
<td>(1.3)</td>
<td>(0.4 3.3)</td>
<td>(1.1)</td>
</tr>
</tbody>
</table>

Conclusion: UT works better than TT1 and TT2.
Example 3: standard sensor networks measurements

**Standard measurements:**

\[ g_{\text{TOA}}(x) = \|x\| = \sqrt{\sum_{i=1}^{n} x_i^2} \]

\[ g_{\text{DOA}}(x) = \arctan2(x_1, x_2), \]

<table>
<thead>
<tr>
<th>TOA 2D: ( g(x) = |x| )</th>
<th>DOA: ( g(x) = \arctan2(x_2, x_1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>X ( \sim \mathcal{N}([3; 0], [1, 0; 0, 10]) )</td>
<td>X ( \sim \mathcal{N}([3; 0], [10, 0; 0, 1]) )</td>
</tr>
<tr>
<td>TT1 ( \sim \mathcal{N}(3, 1) )</td>
<td>TT1 ( \sim \mathcal{N}(0, 0.111) )</td>
</tr>
<tr>
<td>TT2 ( \sim \mathcal{N}(4.67, 6.56) )</td>
<td>TT2 ( \sim \mathcal{N}(0, 0.235) )</td>
</tr>
<tr>
<td>UT2 ( \sim \mathcal{N}(4.08, 3.34) )</td>
<td>UT2 ( \sim \mathcal{N}(0.524, 1.46) )</td>
</tr>
<tr>
<td>MCT ( \sim \mathcal{N}(4.08, 1.94) )</td>
<td>MCT ( \sim \mathcal{N}(0.0702, 1.6) )</td>
</tr>
</tbody>
</table>

**Conclusion:** UT works slightly better than TT1 and TT2. Studying RSS measurements,

\[ g_{\text{RSS}}(x) = c_0 - c_2 \cdot 10 \log_{10}(\|x\|^2), \]

gives similar results.
Lemma 7.1 If

\[
\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N}\left( \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} P_{xx} & P_{xy} \\ P_{xy} & P_{yy} \end{pmatrix} \right) = \mathcal{N}\left( \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, P \right)
\]

Then, the conditional distribution for \( X \), given the observed \( Y = y \), is Gaussian distributed:

\[
(X|Y = y) \sim \mathcal{N}(\mu_x + P_{xy}P_{yy}^{-1}(y - \mu_y), P_{xx} - P_{xy}P_{yy}^{-1}P_{yx})
\]

Connection to the Kalman filter

The Kalman gain is in this notation given by

\[
K_k = P_{xy}P_{yy}^{-1}
\]
Time update: Let

\[ \tilde{x} = \begin{pmatrix} x_k \\ v_k \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \hat{x}_{k|k} \\ 0 \end{pmatrix}, \begin{pmatrix} P_{k|k} & 0 \\ 0 & Q_k \end{pmatrix} \right) \]

\[ z = x_{k+1} = f(x_k, u_k, v_k) = g(\tilde{x}). \]

The transformation approximation (UT, MC, TT1, TT2) gives

\[ z \sim \mathcal{N}(\hat{x}_{k+1|k}, P_{k+1|k}). \]
Measurement update: Let

\[
\bar{x} = \begin{pmatrix} x_k \\ e_k \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \hat{x}_{k|k-1} \\ 0 \end{pmatrix}, \begin{pmatrix} P_{k|k-1} & 0 \\ 0 & R_k \end{pmatrix} \right)
\]

\[
z = \begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} x_k \\ h(x_k, u_k, e_k) \end{pmatrix} = g(\bar{x})
\]

The transformation approximation (UT, MC, TT1, TT2) gives

\[
z \sim \mathcal{N} \left( \begin{pmatrix} \hat{x}_{k|k-1} \\ \hat{y}_{k|k-1} \end{pmatrix}, \begin{pmatrix} P_{xx}^{k|k-1} & P_{xy}^{k|k-1} \\ P_{yx}^{k|k-1} & P_{yy}^{k|k-1} \end{pmatrix} \right)
\]

The measurement update is now

\[
K_k = P_{xy}^{k|k-1} \left( P_{yy}^{k|k-1} \right)^{-1},
\]

\[
\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - \hat{y}_{k|k-1}).
\]
The filter obtained using TT1 is equivalent to the standard EKF1
The filter obtained using TT2 is equivalent to EKF2
The filter obtained using UT is equivalent to UKF
The Monte Carlo approach should be the most accurate, since it asymptotically computes the correct first and second order moments
There is a freedom to mix transform approximations in the time and measurement update
Choice of Nonlinear Filter

- Depends mainly on:
  1. SNR;
  2. the degree of nonlinearity
  3. the degree of non-Gaussian noise, in particular if any distribution is multi-modal (has several local maxima)

- SNR and degree of nonlinearity is connected through the rest term, whose expected value is

  \[
  E((x - \hat{x})^T g''(\xi)(x - \hat{x}) = E\left(\text{tr}(g''(\xi)(x - \hat{x})(x - \hat{x})^T)\right) = \text{tr}(g''(\xi)P)
  \]

- Small rest term requires either high SNR (small \(P\)) or almost linear functions (small \(f''\) and \(h''\)).

- If the rest term is small, use EKF1

- If the rest term is large, and the nonlinearities are essentially quadratic (example \(x^T x\)) use EKF2

- If the rest term is large, and the nonlinearities are not essentially quadratic try UKF

- If the functions are severely nonlinear or any distribution is multi-modal, consider filterbanks or particle filter
Virtual Yaw Rate Sensor

- Yaw rate subject to bias, orientation error increases linearly in time
- Wheel speeds also give a gyro, where the orientation error grows linearly in distance

Model, with state vector $x_k = (\dot{\psi}_k, \ddot{\psi}_k, b_k, r_k, 3 r_k, 4 r_k)$ and the measurements

\[
\begin{align*}
y_1^k &= \dot{\psi}_k + b_k + e_1^k \\
y_2^k &= \frac{\omega_3 r_{nom} + \omega_4 r_{nom}}{2} \frac{2}{B} \left( \frac{\omega_3}{\omega_4} \frac{r_k}{r_{nom}} - 1 \right) + e_2^k.
\end{align*}
\]
Navigation grade IMU gives accurate dead-reckoning, but drift may cause return at bad places. GPS is restricted for high speeds and high accelerations.

Fusion of IMU and GPS when pseudo-ranges are available, with IMU support to tracking loops inside GPS.

- Loose integration: direct fusion approach \( y_k = p_k + e_k \).
- Tight integration: TDOA fusion approach \( y_k^i = |p_k - p_k^i|/c + t_k + e_k \).

http://youtu.be/zRHFXzLQ64
Sounding Rocket

Navigation grade IMU gives accurate dead-reckoning, but drift may cause return at bad places. GPS is restricted for high speeds and high accelerations. Fusion of IMU and GPS when pseudo-ranges are available, with IMU support to tracking loops inside GPS.

- Loose integration: direct fusion approach \( y_k = p_k + e_k \).
- Tight integration: TDOA fusion approach \( y^i_k = |p_k - p^i_k|/c + t_k + e_k \).
MC Leaning Angle

- Headlight steering, ABS and anti-spin systems require leaning angle.
- Gyro very expensive for this application.
- Combination of accelerometers investigated, lateral and downward acc worked fine in EKF.

Model, where $z_y, z_z, a_1, a_2, J$ are constants relating to geometry and inertias of the motorcycle, $u = v_x$

$$x = \left( \begin{array}{ccccccc} \varphi & \dot{\varphi} & \ddot{\varphi} & \dot{\psi} & \ddot{\psi} & \delta_{ay} & \delta_{az} & \delta_{\dot{\varphi}} \end{array} \right)^T$$

$$y = h(x) = \begin{pmatrix} a_y \\ a_z \\ \dot{\varphi} \end{pmatrix} = \begin{pmatrix} u x_4 - z_y x_3 + z_y x_4^2 \tan(x_1) + g \sin(x_1) + x_6 \\ -u x_4 \tan(x_1) - z_z (x_2^2 + x_4^2 \tan^2(x_1)) + g \cos(x_1) + x_7 \\ -a_1 x_3 + a_2 x_4^2 \tan(x_1) - u x_4 J + x_6 \end{pmatrix}$$
Key tool for a unified derivation of KF, EKF, UKF.

\[
\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} P_{xx} & P_{xy} \\ P_{xy} & P_{yy} \end{pmatrix} \right)
\]

\[\Rightarrow (X|Y = y) \sim \mathcal{N}(\mu_x + P_{xy}P_{yy}^{-1}(y - \mu_y), P_{xx} - P_{xy}P_{yy}^{-1}P_{yx})\]

The Kalman gain is in this notation given by \( K_k = P_{xy}P_{yy}^{-1} \).

- In KF, \( P_{xy} \) and \( P_{yy} \) follow from a linear Gaussian model
- In EKF, \( P_{xy} \) and \( P_{yy} \) can be computed from a linearized model (requires analytic gradients)
- In EKF and UKF, \( P_{xy} \) and \( P_{yy} \) computed by NLT for transformation of \((x^T, v^T)^T\) and \((x^T, e^T)^T\), respectively. No gradients required, just function evaluations.