Lecture 5:

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

Lecture 4: Summary

- Detection problems as hypothesis tests:

$$H_0 : \mathbf{y} = \mathbf{e},$$
$$H_1 : \mathbf{y} = \bar{x} + \mathbf{e} = \mathbf{h}(x) + \mathbf{e}.$$

- Neyman-Pearson's lemma: $T(y) = p_{\mathbf{e}}(y - \mathbf{h}(x^0))/p_{\mathbf{e}}(y)$ maximizes P_D for given P_{FA} (best ROC curve).
- In general case

$$T(y) = 2\log p_{\mathbf{e}}(y - \mathbf{h}(x^{ML})) - 2\log p_{\mathbf{e}}(y) \in \chi^2_{\dim(x)}(x^{0,T}\mathcal{I}(x^o)x^o).$$

- Bayes optimal filter

$$p(x_k|y_{1:k}) \propto p_{e_k}(y_k - h(x_k))p(x_k|y_{1:k-1})$$
$$p(x_{k+1}|y_{1:k}) = \int_{\mathcal{R}^{nx}} p_{v_k}(x_{k+1} - f(x_k))p(x_k|y_{1:k})\,dx_k$$

- Intuitive work flow of nonlinear filter:
  - MU: estimation from $y_k = h(x_k) + e_k$ and fusion with $\hat{x}_{k|k-1}$
  - TU: nonlinear transformation $z = f(x_k)$ and diffusion from $x_{k-1} = z_k + v_k$

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

The Kalman filter

Time-varying state space model:

$$\begin{aligned} x_{k+1} &= F_k x_k + G_k v_k, \quad \mathrm{Cov}(v_k) = Q_k \\ y_k &= H_k x_k + e_k, \quad \mathrm{Cov}(e_k) = R_k \end{aligned}$$

Time update:

$$\begin{aligned} \hat{x}_{k+1|k} &= F_k \hat{x}_{k|k} \\ P_{k+1|k} &= F_k P_{k|k} F_k^T + G_k Q_k G_k^T \end{aligned}$$

Measurement update:

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} (y_k - H_k \hat{x}_{k|k-1}) \\ P_{k|k} &= P_{k|k-1} - P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} H_k P_{k|k-1}. \end{aligned}$$

Modifications

Auxiliary quantities: innovation ε_k, innovation covariance S_k and Kalman gain K_k

$$\begin{aligned}
\varepsilon_k &= y_k - H_k \hat{x}_{k|k-1} \\
S_k &= H_k P_{k|k-1} H_k^T + R_k \\
K_k &= P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} = P_{k|k-1} H_k^T S_k^{-1}
\end{aligned}$$

Filter form

$$\begin{aligned}
\hat{x}_{k|k} &= F_{k-1} \hat{x}_{k-1|k-1} + K_k (y_k - H_k F_{k-1} \hat{x}_{k-1|k-1}) \\
&= (F_{k-1} - K_k H_k F_{k-1}) \hat{x}_{k-1|k-1} + K_k y_k,
\end{aligned}$$

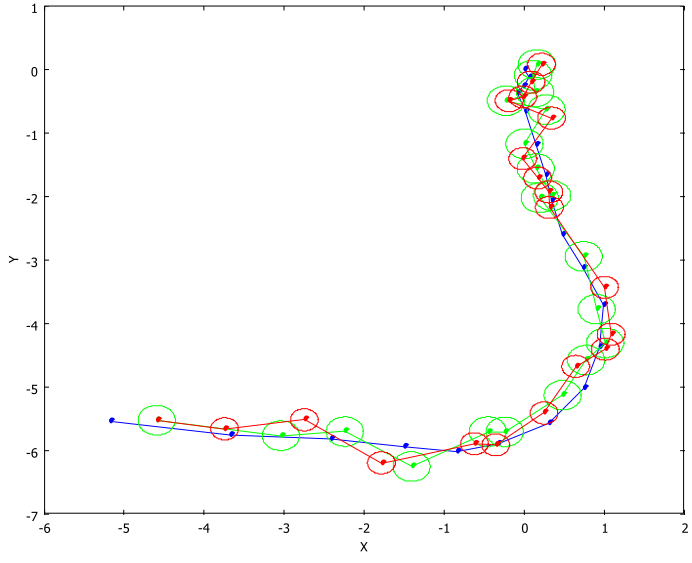
Predictor form

$$\begin{aligned}
\hat{x}_{k+1|k} &= F_k \hat{x}_{k|k-1} + F_k K_k (y_k - H_k \hat{x}_{k|k-1}) \\
&= (F_k - F_k K_k H_k) \hat{x}_{k|k-1} + F_k K_k y_k
\end{aligned}$$

Simulation example

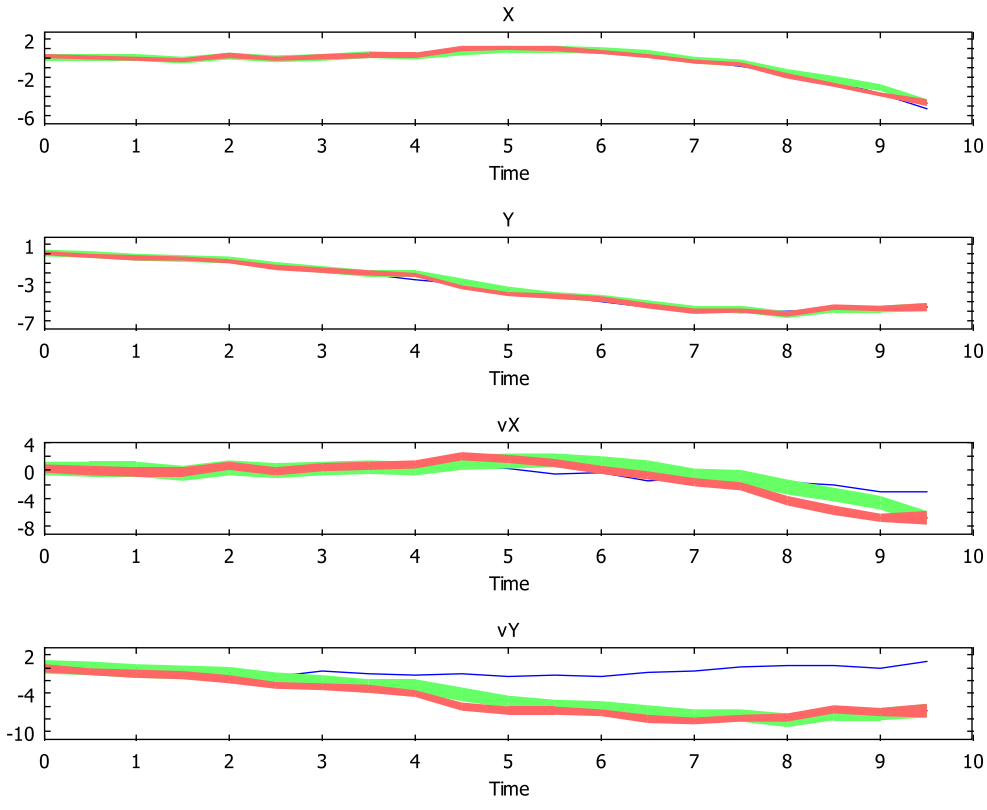
Create a constant velocity model, simulate and Kalman filter

```
T=0.5;
A=[1 0 T 0; 0 1 0 T; 0 0 1 0;
    0 0 0 1];
B=[T^2/2 0; 0 T^2/2; T 0; 0 T];
C=[1 0 0 0; 0 1 0 0];
R=0.03*eye(2);
m=lss(A,[],C,[],B*B',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',1); %
    Stationary
xhat2=kalman(m,z,'alg',4); %
    Smoother
xplot2(z,xhat1,xhat2,'conf',90,[1
    2])
```



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

```
xplot(z,xhat1,xhat2,'conf',99)
```



Tuning the KF

- 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.
- Recommentation: 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 $x_1 \in \mathcal{N}(\hat{x}_{1|0}, P_0)$. 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,

$$\begin{aligned} x_{k+1}|y_{1:k} &\in \mathcal{N}(\hat{x}_{k+1|k}, P_{k+1|k}) \\ x_k|y_{1:k} &\in \mathcal{N}(\hat{x}_{k|k}, P_{k|k}) \\ \varepsilon_k &\in \mathcal{N}(0, S_k) \end{aligned}$$

Robustness and sensitivity

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

- Observability.
- 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.

Observability

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,

$$\mathcal{O} = \begin{pmatrix} H \\ HF \\ HF^2 \\ \vdots \\ HF^{n-1} \end{pmatrix} \qquad \mathcal{O}_k = \begin{pmatrix} H_{k-n+1} \\ H_{k-n+2}F_{k-n+1} \\ H_{k-n+3}F_{k-n+2}F_{k-n+1} \\ \vdots \\ H_kF_{k-1}\dots F_{k-n+1} \end{pmatrix}.$$

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 $S_k = \mathrm{E}(\varepsilon_k \varepsilon_k^T)$ (note that $\varepsilon_k \in \mathcal{N}(0, S_k)$)?
Principal reasons:

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

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

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

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

$$T(y_k^i) = \varepsilon_k^{i,T} S_k^{-1} \varepsilon_k^i \sim \chi^2(\dim(y_k^i)).$$

Divergence monitoring

Related to outlier detection, but performance is monitored on a longer time horizon.
One way to modify the chi2-test to a Gaussian test using the central limit theorem:

$$T = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{\dim(y_k)}\varepsilon_k^T S_k^{-1}\varepsilon_k \sim \mathcal{N}\left(1, \frac{2}{\sum_{k=1}^{N}\dim(y_k)}\right),$$

If

$$(T-1)\sqrt{\sum_{k=1}^{N}\dim(y_k)/2} > h_{P_{FA}},$$

filter divergence can be concluded, and the filter must be restarted. Instead of all data, a long sliding window or an exponential window (forgetting factor) can be used.

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 θ with $\mathrm{E}(\theta) = \hat{\theta}$ and $\mathrm{Cov}(\theta) = P_\theta$.
The state estimate $\hat{x}_k$ is a stochastic variable with four stochastic sources, v_k, e_k, x_1 at one hand, and θ on the other hand. . The law of total variance ($\mathrm{Var}(X) = \mathrm{E}\,\mathrm{Var}(X|Y) + \mathrm{Var}\,\mathrm{E}(X|Y)$) and Gauss approximation formula ($\mathrm{Var}(h(Y)) \approx h'_Y(\bar{Y})\mathrm{Var}(Y)(h'_Y(\bar{Y}))^T$) gives

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

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=0.5*P+0.5*P'`
- 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.

EKF and UKF

Theory from Chapter 8:

- 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).

Applications from Chapter 16:

- Automotive applications: yaw rate, roll ange, friction estimation.
- Rocket application: integration of IMU and GPS.

Non-linear transformations

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

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 *a priori*. 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).

TT1: first order Taylor approximation

Gauss' approximation formula

$$x \in \mathcal{N}(\hat{x}, P) \rightarrow \mathcal{N}(g(\hat{x}), \left[g_i'(\hat{x}) P (g_j'(\hat{x}))^T\right]_{ij}) = \mathcal{N}(g(\hat{x}), g'(\hat{x}) P (g'(\hat{x}))^T$$

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

$$\begin{aligned} z &= Gx, \\ x &\in \mathcal{N}(\hat{x}, P) \rightarrow \\ z &\in \mathcal{N}(G\hat{x}, GPG^T) \end{aligned}$$

Note that GPG^T can be written $[G_i P G_j^T]_{ij}$, where G_i denotes row i of G.

TT2: second order Taylor approximation

$$x \in \mathcal{N}(\hat{x}, P) \rightarrow$$

$$\mathcal{N}\Big(g(\hat{x}) + [\mathrm{tr}(g_i''(\hat{x})P)]_i, \left[g_i'(\hat{x})P(g_j'(\hat{x}))^T + \frac{1}{2}\mathrm{tr}(Pg_i''(\hat{x})Pg_j''(\hat{x}))\right]_{ij}\Big)$$

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, *without* forming the Hessian $g''(x)$ explicitly. This reduces the n_x^5 complexity in $\left[Pg_i''(\hat{x})Pg_j''(\hat{x}))\right]_{ij}$ to n_x^3 complexity.

MC: Monte Carlo sampling

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

$$
\begin{aligned}
x^{(i)} &\in \mathcal{N}\left(\hat{x}, P\right), \\
z^{(i)} &= g(x^{(i)}), \\
\mu_z &= \frac{1}{N} \sum_{i=1}^{N} z^{(i)}, \\
P_z &= \frac{1}{N-1} \sum_{i=1}^{N} \left(z^{(i)} - \mu_z\right)\left(z^{(i)} - \mu_z\right)^T.
\end{aligned}
$$

Not commonly used in nonlinear filtering, but a valid and solid approach!

UT: the unscented transform

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

$$
\begin{aligned}
x^0 &= \hat{x}, \\
x^{\pm i} &= \hat{x} \pm \sqrt{n_x + \lambda} P^{1/2}_{:,i}, \quad i = 1, 2, \ldots, n_x, \\
z^i &= g(x^i), \\
\mathrm{E}(z) &\approx \frac{\lambda}{n_x + \lambda} z^0 + \sum_{i=\pm 1}^{\pm N} \frac{1}{2(n_x + \lambda)} z^i, \\
\mathrm{Cov}(z) &\approx \left(\frac{\lambda}{n_x + \lambda} + (1 - \alpha^2 + \beta) \right) \left(z^0 - \mathrm{E}(z)\right)\left(z^0 - \mathrm{E}(z)\right)^T \\
&\quad + \sum_{i=\pm 1}^{\pm N} \frac{1}{2(n_x + \lambda)} \left(z^i - \mathrm{E}(z)\right)\left(z^i - \mathrm{E}(z)\right)^T.
\end{aligned}
$$

UT: design parameters

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

Note that

- $n_x + \lambda = \alpha^2 n_x$ 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

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

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.

n	TT1	TT2	UT1	UT2	MCT
1	N(0,0)	N(1,2)	N(1,2)	N(1,2)	N(1.02,2.15)
2	N(0,0)	N(2,4)	N(2,2)	N(2,8)	N(2.02,4.09)
3	N(0,0)	N(3,6)	N(3,0)	N(3,18)	N(3.03,6.3)
4	N(0,0)	N(4,8)	N(4,-4)	N(4,32)	N(4.03,8.35)
5	N(0,0)	N(5,10)	N(5,-10)	N(5,50)	N(5.08,10.4)
Theory	N(0,0)	$N(n, 2n)$	$N(n, (3-n)n)$	$N(n, 2n^2)$	$\rightarrow N(n, 2n)$

Conclusion: TT2 works, not the unscented transforms.

Example 2: radar

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

X	TT1	TT2
$\begin{pmatrix} 3.0 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 3.0 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 9.0 \end{pmatrix}$	$\begin{pmatrix} 2.0 \\ -0.0 \end{pmatrix}, \begin{pmatrix} 3.0 & 0.0 \\ 0.0 & 10.0 \end{pmatrix}$
$\begin{pmatrix} 3.0 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 2.6 \\ 1.5 \end{pmatrix}, \begin{pmatrix} 3.0 & -3.5 \\ -3.5 & 7.0 \end{pmatrix}$	$\begin{pmatrix} -1.4 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 27.0 & 2.5 \\ 2.5 & 9.0 \end{pmatrix}$
$\begin{pmatrix} 3.0 \\ 0.8 \end{pmatrix}, \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 2.1 \\ 2.1 \end{pmatrix}, \begin{pmatrix} 5.0 & -4.0 \\ -4.0 & 5.0 \end{pmatrix}$	$\begin{pmatrix} 2.1 \\ 2.1 \end{pmatrix}, \begin{pmatrix} 9.0 & 0.0 \\ 0.0 & 13.0 \end{pmatrix}$

UT1	UT2	MCT
$\begin{pmatrix} 1.8 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 3.7 & 0.0 \\ 0.0 & 2.9 \end{pmatrix}$	$\begin{pmatrix} 1.5 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 5.5 & 0.0 \\ 0.0 & 9.0 \end{pmatrix}$	$\begin{pmatrix} 1.8 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 2.5 & 0.0 \\ 0.0 & 4.4 \end{pmatrix}$
$\begin{pmatrix} 1.6 \\ 0.9 \end{pmatrix}, \begin{pmatrix} 3.5 & 0.3 \\ 0.3 & 3.1 \end{pmatrix}$	$\begin{pmatrix} 1.3 \\ 0.8 \end{pmatrix}, \begin{pmatrix} 6.4 & -1.5 \\ -1.5 & 8.1 \end{pmatrix}$	$\begin{pmatrix} 1.6 \\ 0.9 \end{pmatrix}, \begin{pmatrix} 2.9 & -0.8 \\ -0.8 & 3.9 \end{pmatrix}$
$\begin{pmatrix} 1.3 \\ 1.3 \end{pmatrix}, \begin{pmatrix} 3.3 & 0.4 \\ 0.4 & 3.3 \end{pmatrix}$	$\begin{pmatrix} 1.1 \\ 1.1 \end{pmatrix}, \begin{pmatrix} 7.2 & -1.7 \\ -1.7 & 7.2 \end{pmatrix}$	$\begin{pmatrix} 1.3 \\ 1.3 \end{pmatrix}, \begin{pmatrix} 3.4 & -1.0 \\ -1.0 & 3.4 \end{pmatrix}$

Conclusion: UT works better than TT1 and TT2

Example 3

$$g_{\mathrm{TOA}}(x) = \|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

$$g_{\mathrm{RSS}}(x) = c_0 - c_2 \cdot 10 \log_{10}(\|x\|^2)$$

$$g_{\mathrm{DOA}}(x) = \mathrm{arctan2}(x_1, x_2),$$

Conclusion from these standard sensor network examples: UT works slightly better than TT1 and TT2

TOA2d $g(x) = \|x\|$	N([3;0],[1 0;0 10])
TT1	N(3,1)
TT2	N(4.67,6.56)
UT2	N(4.08,3.34)
MCT	N(4.08,1.94)

DOA $g(x) = \mathrm{arctan2}(x_1, x_2)$	N([3;0],[10,0;0,1])
TT1	N(0,0.111)
TT2	N(0,0.235)
UT2	N(0.524,1.46)
MCT	N(0.0702,1.6)

EKF1 and EKF2 principle

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

$$
\begin{aligned}
S_k &= h'_x(\hat{x}_{k|k-1})P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T \\
&\quad + h'_e(\hat{x}_{k|k-1})R_k(h'_e(\hat{x}_{k|k-1}))^T \\
K_k &= P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T S_k^{-1} \\
\varepsilon_k &= y_k - h(\hat{x}_{k|k-1}, 0) - \tfrac{1}{2}\left[\mathrm{tr}(h''_{i,x}P_{k|k-1})\right]_i \\
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k\varepsilon_k \\
P_{k|k} &= P_{k|k-1} \\
&\quad - P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T S_k^{-1} h'_x(\hat{x}_{k|k-1})P_{k|k-1} \\
&\quad \tfrac{1}{2}\left[\mathrm{tr}(h''_{i,x}(\hat{x}_{k|k-1})P_{k|k-1}h''_{j,x}(\hat{x}_{k|k-1})P_{k|k-1})\right]_{ij} \\
\hat{x}_{k+1|k} &= f(\hat{x}_{k|k}, 0) \\
P_{k+1|k} &= f'_x(\hat{x}_{k|k})P_{k|k}(f'_x(\hat{x}_{k|k}))^T \\
&\quad + f'_v(\hat{x}_{k|k})Q_k(f'_v(\hat{x}_{k|k}))^T. \\
&\quad \tfrac{1}{2}\left[\mathrm{tr}(f''_{i,x}(\hat{x}_{k|k})P_{k|k}f''_{j,x}(\hat{x}_{k|k})P_{k|k})\right]_{ij}.
\end{aligned}
$$

Comments

- The EKF1, using the TT1transformation, 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 as 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 $F_i P F_j^T$ 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

Derivative-free algorithms

Numeric derivatives are preferred in the following cases:

- The non-linear 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!

KF, EKF and UKF in one framework

First, recall the lemma

$$\begin{pmatrix} X \\ Y \end{pmatrix} \in \mathrm{N}\left(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} P_{xx} & P_{xy} \\ P_{xy} & P_{yy} \end{pmatrix} \right) = \mathrm{N}\left(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, P \right)$$

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

$$(X|Y = y) \in N(\mu_x + P_{xy}P_{yy}^{-1}(y - \mu_y), P_{xx} - P_{xy}P_{yy}^{-1}P_{yx})$$

The Kalman gain is in this notation given by

$$K_k = P_{xy}P_{yy}^{-1}.$$

Algorithm

Time update: Let

$$\bar{x} = \begin{pmatrix} x_k \\ v_k \end{pmatrix} \in \mathcal{N}\left(\begin{pmatrix} \hat{x}_{k|k} \\ 0 \end{pmatrix}, \begin{pmatrix} P_{k|k} & 0 \\ 0 & Q_k \end{pmatrix} \right)$$
$$z = x_{k+1} = f(x_k, u_k, v_k) = g(\bar{x}).$$

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

$$z \sim \mathcal{N}\left(\hat{x}_{k+1|k}, P_{k+1|k}\right).$$

Measurement update: Let

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

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

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

$$K_k = P^{xy}_{k|k-1} \left(P^{yy}_{k|k-1} \right)^{-1},$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \left(y_k - \hat{y}_{k|k-1} \right).$$

Linear case

Time update: $f(x_k, u_k, v_k) = Ax_k + v_k$ gives

$$z \sim \mathcal{N}\left(A\hat{x}_{k|k}, AP_{k|k}A^T + Q_k\right) = \mathcal{N}\left(\hat{x}_{k+1|k}, P_{k+1|k}\right)$$

This is the KF time update.

Measurement update: $h(x_k, e_k) = Hx_k + e_k$ gives

$$z \sim \mathcal{N}\left(\begin{pmatrix} \hat{x}_{k|k-1} \\ H\hat{x}_{k|k-1} \end{pmatrix}, \begin{pmatrix} P_{k|k-1} & HP_{k|k-1} \\ P_{k|k-1}H^T & HP_{k|k-1}H^T + R \end{pmatrix}\right)$$

This gives the KF measurement update!

Comments

- 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.

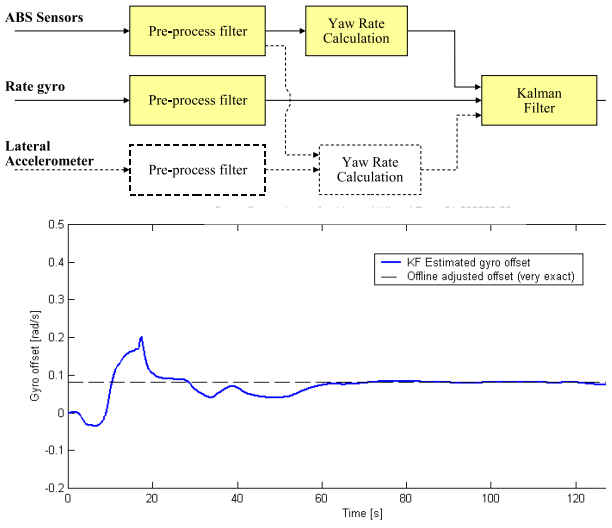
Choice of nonlinear filter

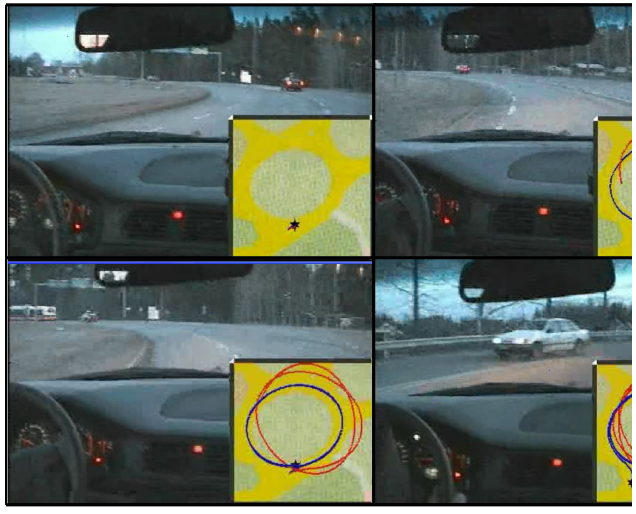
- Depends mainly on (i) SNR, (ii) the degree on nonlinarity and (iii) the degree of non-Gaussian noise, in particular if any distribution is multi-modal (has several local maxima).
- SNR and degree of nonlinearity is connected through the rest term, whose expected value is

$$\mathrm{E}(x-\hat{x})^T g''(\xi)(x-\hat{x}) = \mathrm{E}\,\mathrm{trace} g''(\xi)(x-\hat{x})(x-\hat{x})^T = \mathrm{trace} g''(\xi)P.$$

  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 severly nonlinear or any distribution is multi-modal, consider filterbanks or particle filter.

Course estimation in cars





Car with course gyro and wheel speed sensors. Gyro has drift. Wheel speeds can be converted to course angular rate, but with a (speed dependent) drift. EKF can correct the drifts!
http://youtu.be/d9rzCCIBS9I

$$x_k = (\dot{\psi}_k, \ddot{\psi}_k, b_k, \frac{r_{k,3}}{r_{k,4}}),$$

$$y_k^1 = \dot{\psi}_k + b_k + e_k^1,$$

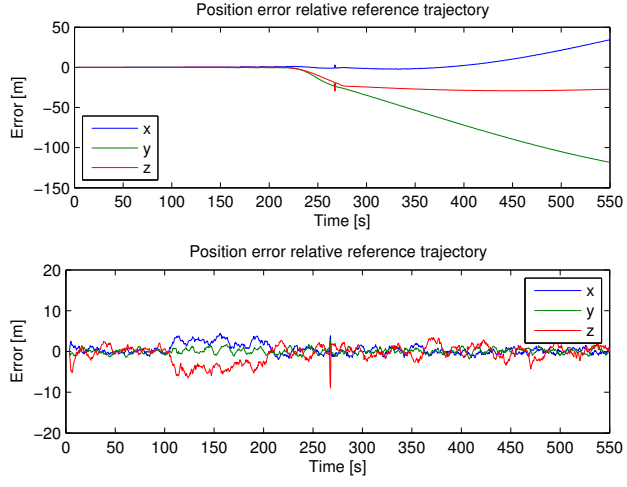
$$y_k^2 = \frac{\omega_3 r_{nom} + \omega_4 r_{nom}}{2} \frac{2}{B} \frac{\frac{\omega_3}{\omega_4}\frac{r_{k,3}}{r_{k,4}} - 1}{\frac{\omega_3}{\omega_4}\frac{r_{k,3}}{r_{k,4}} + 1} + e_k^2.$$

Fusion of GPS and IMU in a Sounding Rocket

Possibilities:

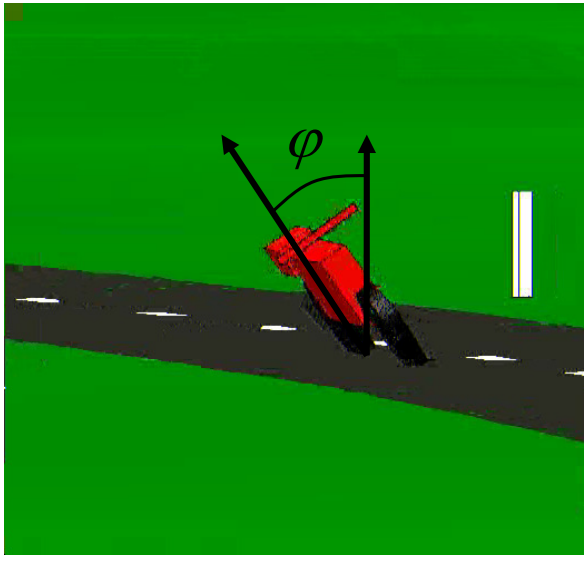
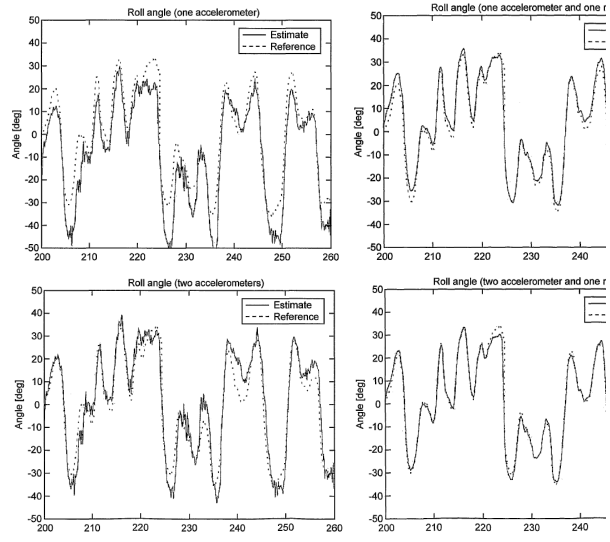
- Loose integration: GPS is seen as one position sensor.
- Tight integration: GPS is seen as many TDOA sensors.

Latter option gives better outlier detection possibilities.
http://youtu.be/zRHFXfZLQ64



Roll angle estimation in motor cycles

Applications: ABS, traction control, headlight control
Sensors: many possible configurations of gyroscopes and accelerometers. http://youtu.be/hT6S1FgHxOc



F