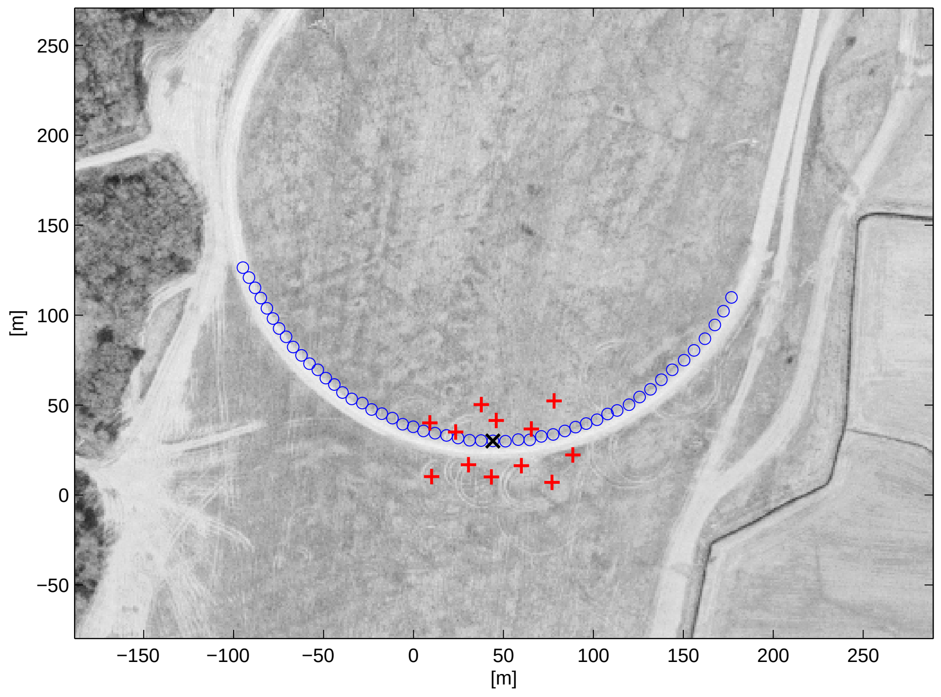
Sensor Fusion

Fredrik Gustafsson

Lecture	Content	Chapters
1	Course overview. Estimation theory for linear and nonlinear models.	1–3
2	Sensor networks and detection theory	4–5
3	Nonlinear filter theory. The Kalman filter, EKF, UKF, KFB.	6–8, 10
4	The point-mass filter and the particle filter.	9
5	The particle filter theory. The marginalized particle filter.	9

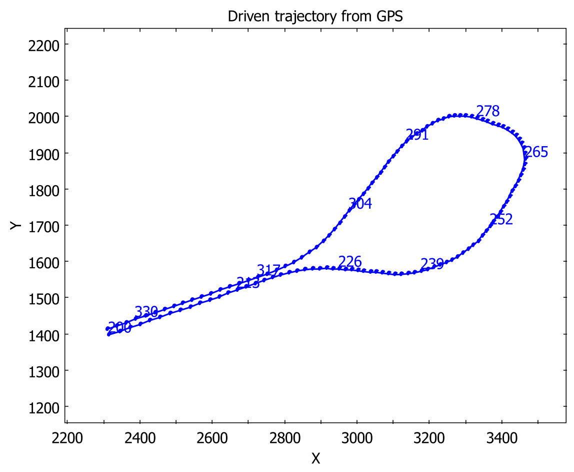
Literature: Statistical Sensor Fusion. Studentlitteratur, 2010.
Exercises: compendium.
Software: *Signals and Systems Lab* for Matlab.
Laboration: online

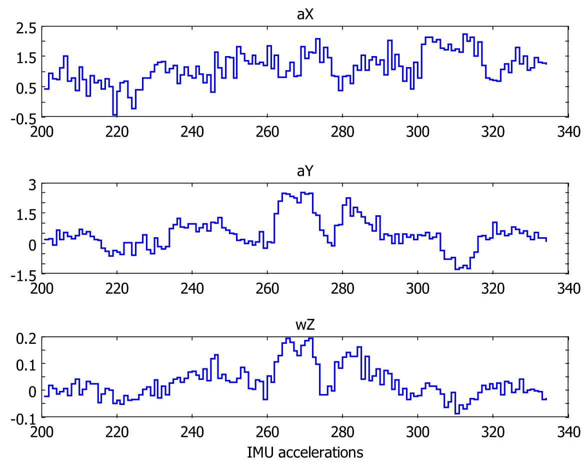
Example 1: sensor network



12 sensor nodes, each one with microphone, geophone and magnetometer.
One moving target.
Detect, localize and track/predict the target.

Example 2: fusion of GPS and IMU





GPS gives good position. IMU gives accurate accelerations.
Combine these to get even better position, velocity and acceleration.

Example 3: fusion of camera and radar images

Radar gives range and range rate with good horizontal angle resolution, but no vertical resolution.
Camera gives very good angular resolution, and color, but no range.
Combined, they have a great potential for situation awareness.

Chapter 2: Estimation theory in linear models

Whiteboard:

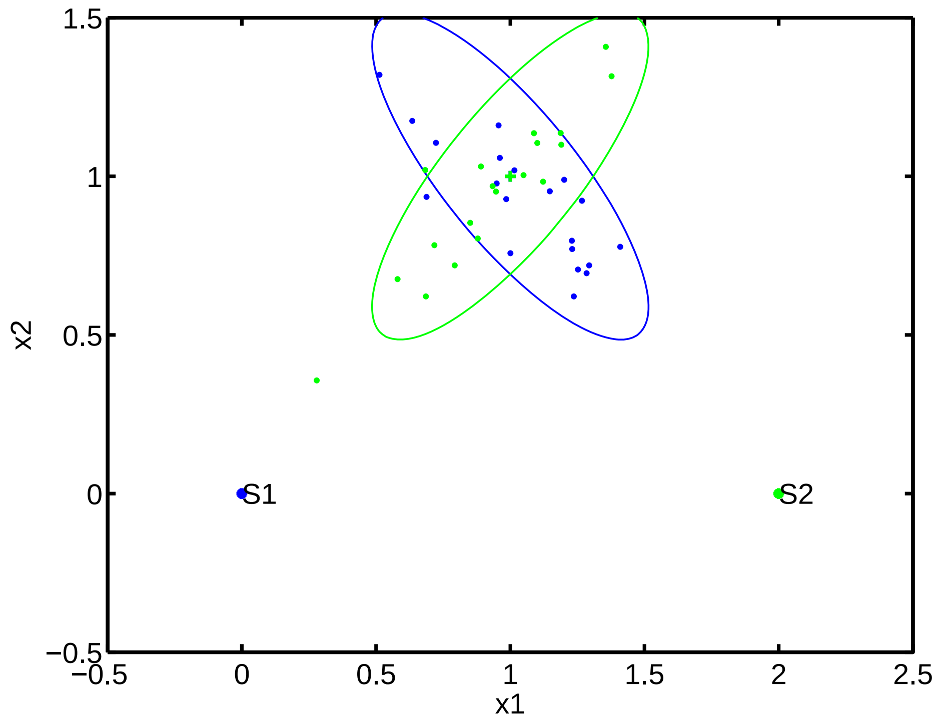
- The weighted least squares (WLS) method.
- The maximum likelihood (ML) method.
- The Cramér-Rao lower bound (CRLB).
- Fusion algorithms.

Slides:

- Examples
- Code examples
- Algorithms

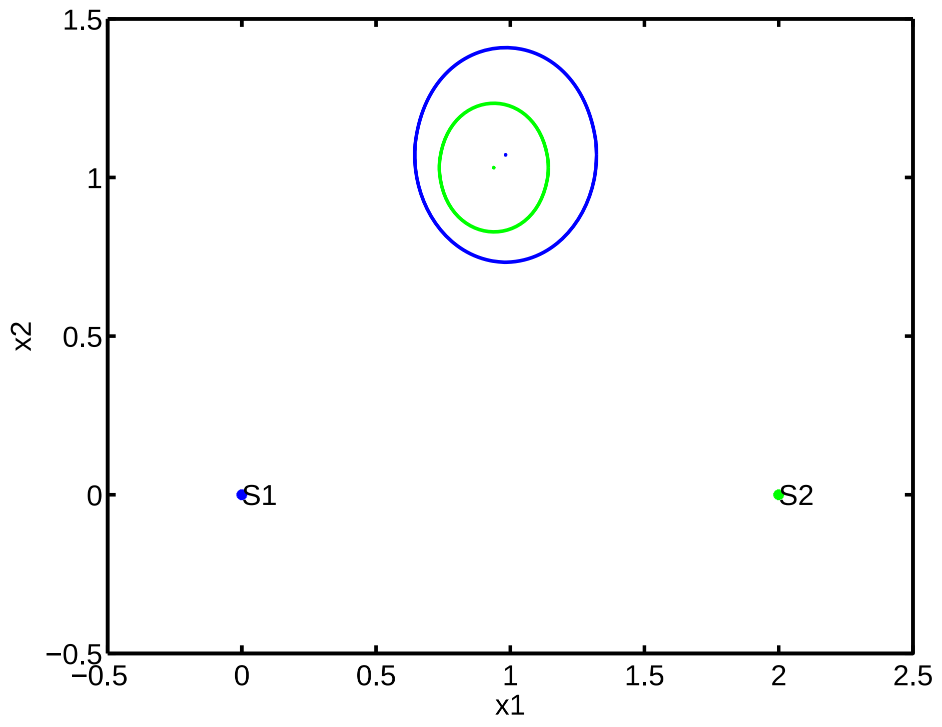
Code for *Signals and Systems Lab*:

```
p1=[0;0];
p2=[2;0];
x=[1;1];
X1=ndist(x,0.1*[1 -0.8;-0.8 1]);
X2=ndist(x,0.1*[1 0.8;0.8 1]);
plot2(X1,X2)
```



Sensor network example, cont'd

```
X3=fusion(X1,X2);    % WLS
X4=0.5*X1+0.5*X2;    % LS
plot2(X3,X4)
```



Information loops in sensor networks

- Information and sufficient statistics should be communicated in sensor networks.
- In sensor networks with untagged observations, our own observations may be included in the information we receive.
- Information loops (updating with the same sensor reading several times) give rise to optimistic covariances.
- Safe fusion algorithms (or *covariance intersection techniques*) give conservative covariances, using a worst case way of reasoning.

Safe fusion

Given two unbiased estimates $\hat{x}_1, \hat{x}_2$ with information $\mathcal{I}_1 = P_1^{-1}$ and $\mathcal{I}_2 = P_2^{-1}$ (pseudo-inverses if singular covariances), respectively. Compute the following:

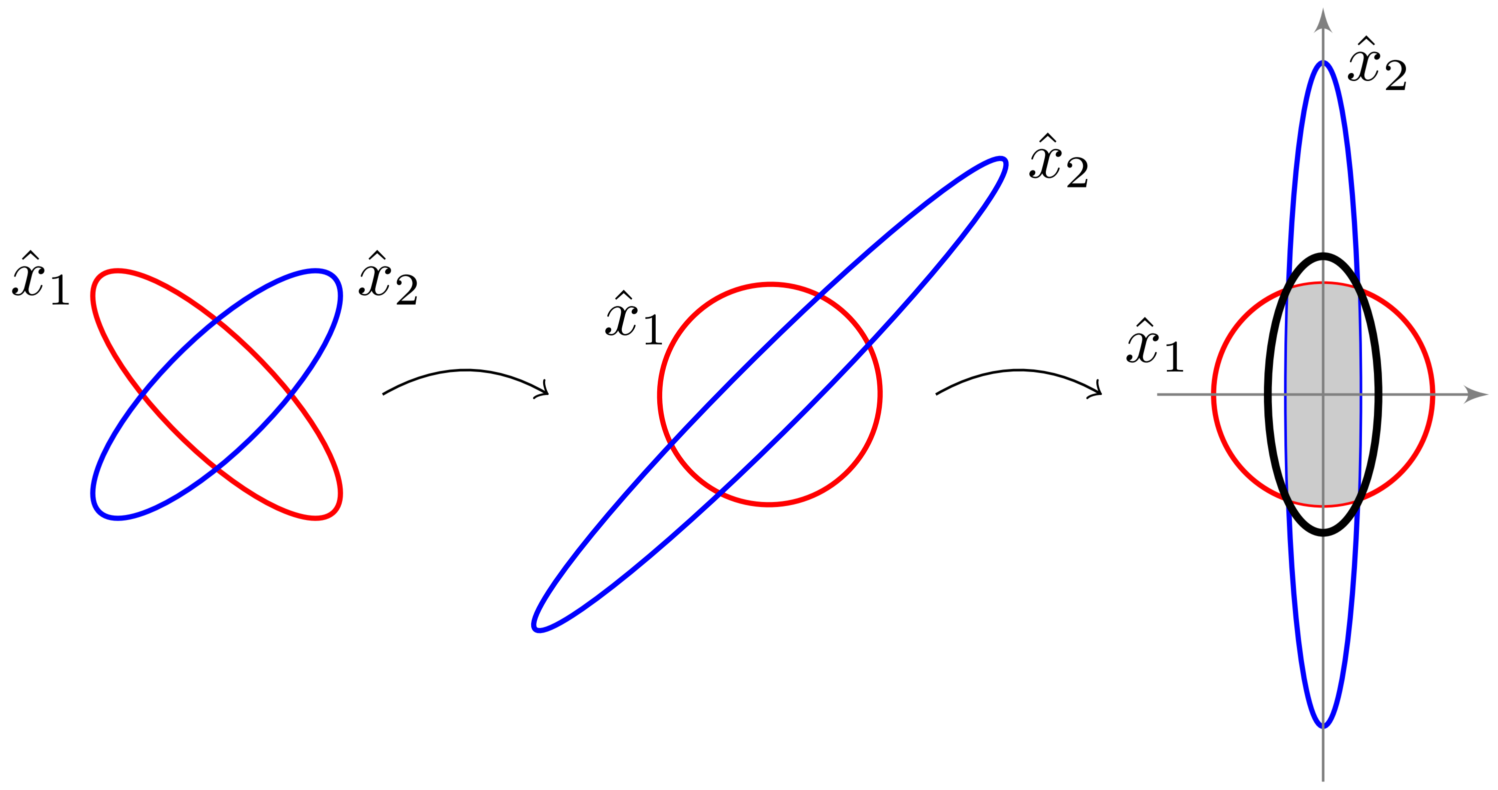
1. SVD: $\mathcal{I}_1 = U_1 D_1 U_1^T$.
2. SVD: $D_1^{-1/2} U_1^T \mathcal{I}_2 U_1 D_1^{-1/2} = U_2 D_2 U_2^T$.
3. Transformation matrix: $T = U_2^T D_1^{1/2} U_1$.
4. State transformation: $\hat{\hat{x}}_1 = T\hat{x}_1$ and $\hat{\hat{x}}_2 = T\hat{x}_2$. The covariances of these are $\text{Cov}(\hat{\hat{x}}_1) = I$ and $\text{Cov}(\hat{\hat{x}}_2) = D_2^{-1}$, respectively.
5. For each component $i = 1, 2, \ldots, n_x$, let

$$\hat{\hat{x}}^i = \hat{\hat{x}}_1^i, \quad D^{ii} = 1 \quad \text{if} \quad D_2^{ii} < 1,$$
$$\hat{\hat{x}}^i = \hat{\hat{x}}_2^i, \quad D^{ii} = D_2^{ii} \quad \text{if} \quad D_2^{ii} > 1.$$

6. Inverse state transformation:

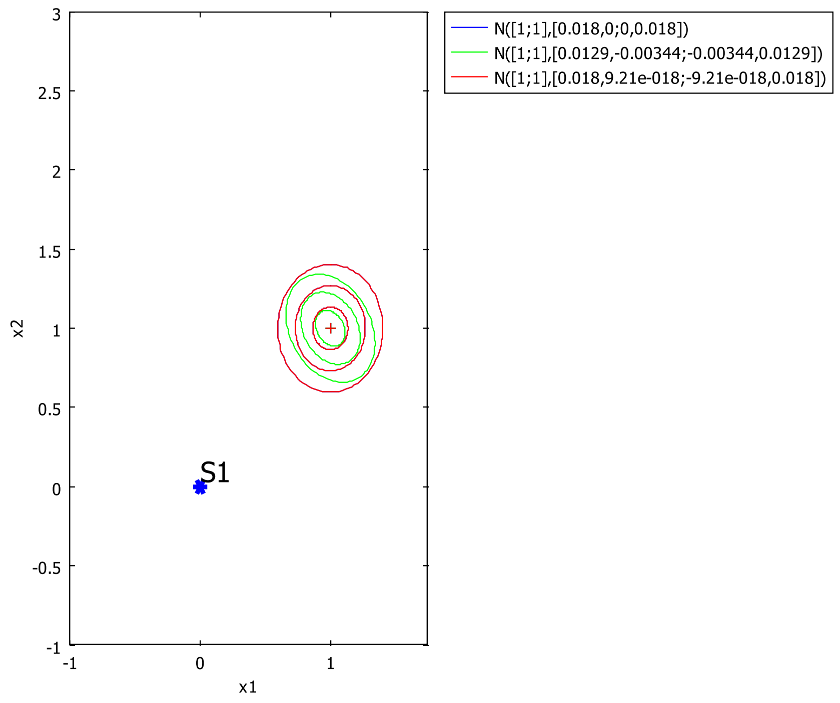
$$\hat{x} = T^{-1}\hat{\hat{x}}, \quad P = T^{-1} D^{-1} T^{-T}$$

Transformation steps



Sensor network example, cont'd

```
X3=fusion(X1,X2);       % WLS
X4=fusion(X1,X3);       % X1 used twice
X5=safefusion(X1,X3);
plot2(X3,X4,X5)
```



Sequential WLS

The WLS estimate can be computed recursively in the space/time sequence y_k. Suppose the estimate $\hat{x}_{k-1}$ with covariance P_k based on observations $y_{1:k-1}$. A new observation is fused using

$$\hat{x}_k = \hat{x}_{k-1} + P_{k-1}H_k^T \left(H_k P_{k-1} H_k^T + R_k\right)^{-1} (y_k - H_k \hat{x}_{k-1}),$$
$$P_k = P_{k-1} - P_{k-1}H_k^T \left(H_k P_{k-1} H_k^T + R_k\right)^{-1} H_k P_{k-1}.$$

Note that the fusion formula can be used alternatively. In fact, the derivation is based on the information fusion formula applying the matrix inversion lemma.

Batch vs sequential evaluation of loss function

The minimizing loss function can be computed in two ways using batch and sequential computations, respectively,

$$\begin{aligned} V^{WLS}(\hat{x}_N) &= \sum_{k=1}^{N} (y_k - H_k \hat{x}_N)^T R_k^{-1} (y_k - H_k \hat{x}_N) \\ &= \sum_{k=1}^{N} (y_k - H_k \hat{x}_{k-1})^T (H_k P_{k-1} H_k^T + R_k)^{-1} (y_k - H_k \hat{x}_{k-1}) \\ &- (\hat{x}_0 - \hat{x}_N)^T P_0^{-1} (\hat{x}_0 - \hat{x}_N) \end{aligned}$$

The second expression should be used in decentralized sensor network implementations and on-line algorithms.

The last correction term to de-fuse the influence of the initial values is needed only when this initialization is used.

Batch vs sequential evaluation of likelihood

The Gaussian likelihood of data is important in model validation, change detection and diagnosis. Generally, Bayes formula gives

$$p(y_{1:N}) = p(y_1) \prod_{k=2}^{N} p(y_k | y_{1:k-1}).$$

For Gaussian noise and using the sequential algorithm, this is

$$p(y_{1:N}) = \prod_{k=1}^{N} \gamma(y_k; H_k \hat{x}_{k-1}, H_k P_{k-1} H_k^T + R_k)$$

The off-line form is:

$$p(y_{1:N}) = \gamma(\hat{x}_N; x_0, P_0) \sqrt{\det(P_N)} \prod_{k=1}^{N} \gamma(y_k; H_k \hat{x}_N, R_k).$$

Prefered for de-centralized computation in sensor networks or on-line algorithms.

Chapter 3: Estimation theory in nonlinear models and sensor networks

Whiteboard:

- The grid method.
- The nonlinear least squares (NLS) method.
- Typical sensor models.
- Sensor fusion based on WLS and the fusion formula.

Slides:

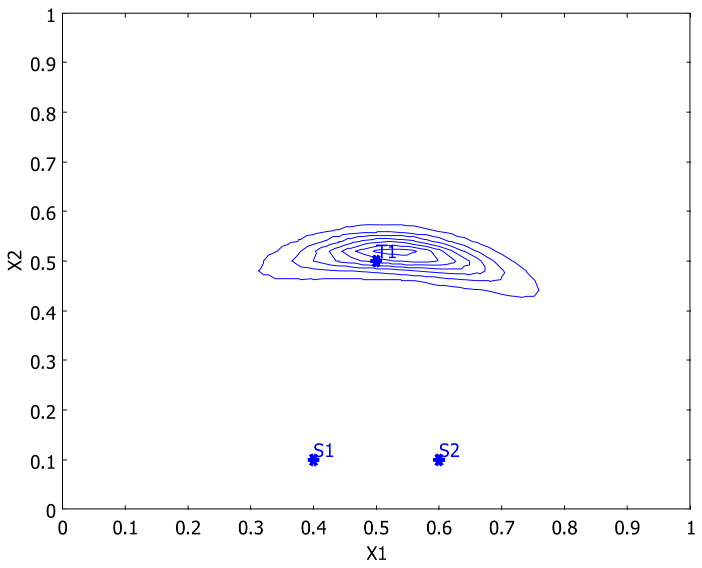
- Details on sensor models.
- Examples.
- Dedicated least squares methods.

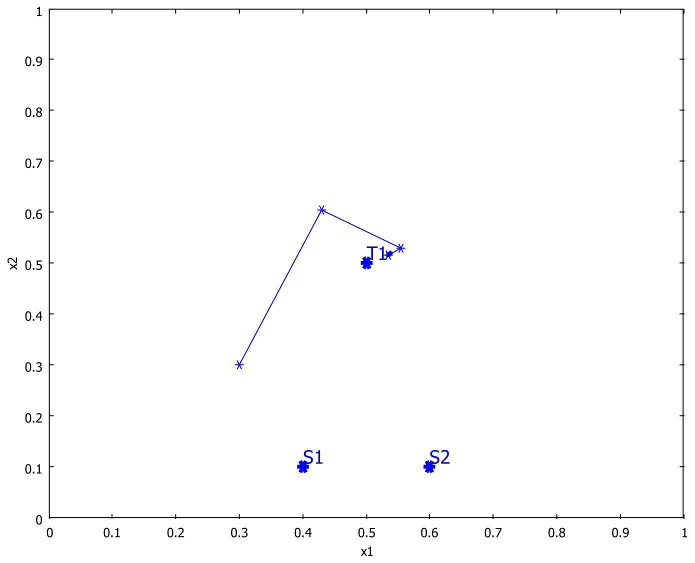
Chapter 3 overview

- Model $y_k = h_k(x) + e_k$
- Ranging sensor example.
- Fusion based on
  - inversion of $h(x)$.
  - gridding the parameter space
  - Taylor expansion of $h(x)$.
- Nonlinear least squares.
- Sub-linear models.
- Implicit models $h_k(y_k, x, e_k) = 0$

NLS for a pair of TOA sensors

```
th=[0.4 0.1 0.6 0.1]; x0=[0.5 0.5];   % Positions
s=exsensor('toa',2);                  % TOA sensor model
s.th=th; s.x0=x0;                     % Change defaults
s.pe=0.001*eye(2);                    % Noise variance
plot(s), hold on                      % Plot network
y=simulate(s,1);                      % Generate observations
lh2(s,y,[0:0.02:1],[0:0.02:1]);       % Likelihood function plot
```





The likelihood function and the iterations in the NLS estimate.

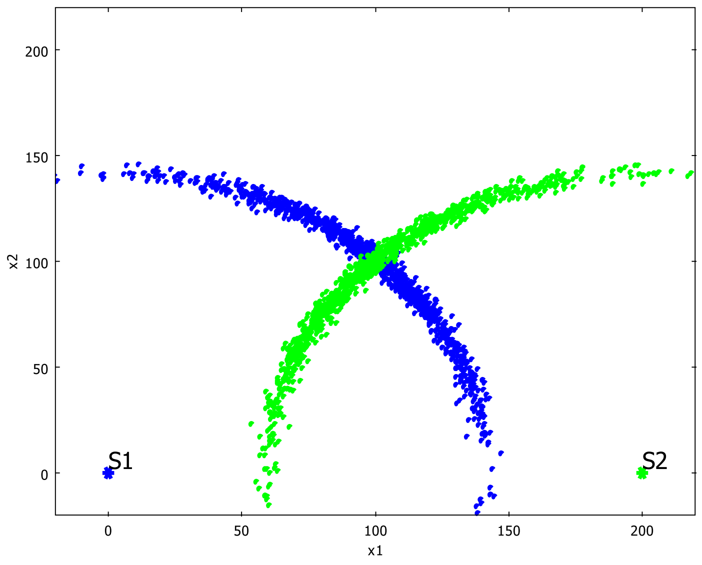
```
s0=s; s0.x0=[0.3;0.3];                   % Prior model for estimation
[xhat,shat,res]=ml(s0,y);                % ML calls NLS
shat                                     % Display estimated signal model
SIGMOD object: TOA (calibrated from data)
         / sqrt((x(1,:)-th(1)).^2+(x(2,:)-th(2)).^2) \
     y = \ sqrt((x(1,:)-th(3)).^2+(x(2,:)-th(4)).^2) / + e
   x0' = [0.54      0.52] + N(0,[3.2e-013,2.3e-013;2.3e-013,1.7e-013])
   th' = [0.4       0.1         0.6         0.1]

xplot2(xhat,'conf',90)                   % Estimate and covariance plot
plot(res.TH(1,:),res.TH(2,:),'*-')       % Estimate for each iteration
```

Simulation example 1

Generate measurements of range and bearing. Invert $x = h{-}1(y)$ for each sample. Banana shaped distribution of estimates.

```
R1=ndist(100*sqrt(2),5);
Phi1=ndist(pi/4,0.1);
p1=[0;0];
hinv=inline('[p(1)+R*cos(Phi);
p(2)+R*sin(Phi)]','R','Phi','p');
R2=ndist(100*sqrt(2),5);
Phi2=ndist(3*pi/4,0.1);
p2=[200;0];
xhat1=hinv(R1,Phi1,p1);
xhat2=hinv(R2,Phi2,p2);
figure(1)
plot2(xhat1,xhat2,'legend','')
```



Analytic approximation of Cov(x)

For the radar sensor, one can show

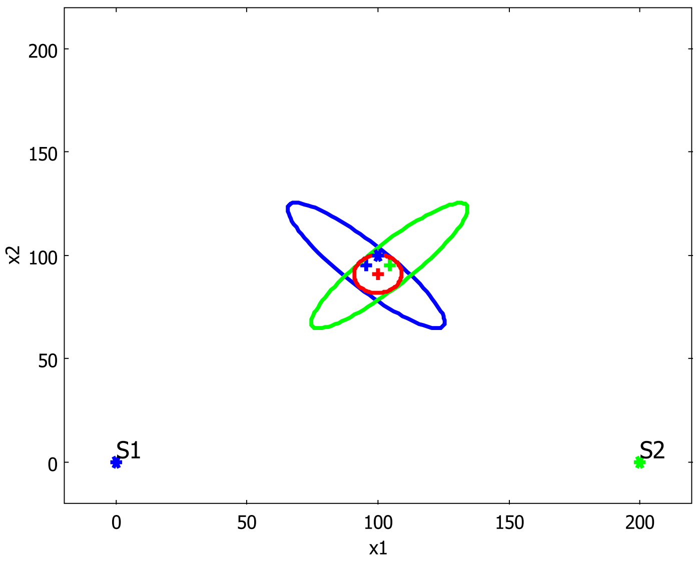
$$\begin{aligned} \text{Cov}(x) &= \frac{\sigma_r^2 - r^2\sigma_\varphi^2}{2} \begin{pmatrix} b + \cos(2\varphi) & \sin(2\varphi) \\ \sin(2\varphi) & b - \cos(2\varphi) \end{pmatrix} \\ b &= \frac{\sigma_r^2 + r^2\sigma_\varphi^2}{\sigma_r^2 - r^2\sigma_\varphi^2}. \end{aligned}$$

Approximation accurate if $r\sigma_\varphi^2/\sigma_r < 0.4$.
This is normally the case in radar applications.
It does not hold in the example where $r\sigma_\varphi^2/\sigma_r = 100\sqrt{2} \cdot 0.1/\sqrt{5} \approx 6.3$.

Simulation example 2

Fit a Gaussian to the Monte Carlo samples and apply the sensor fusion formula.

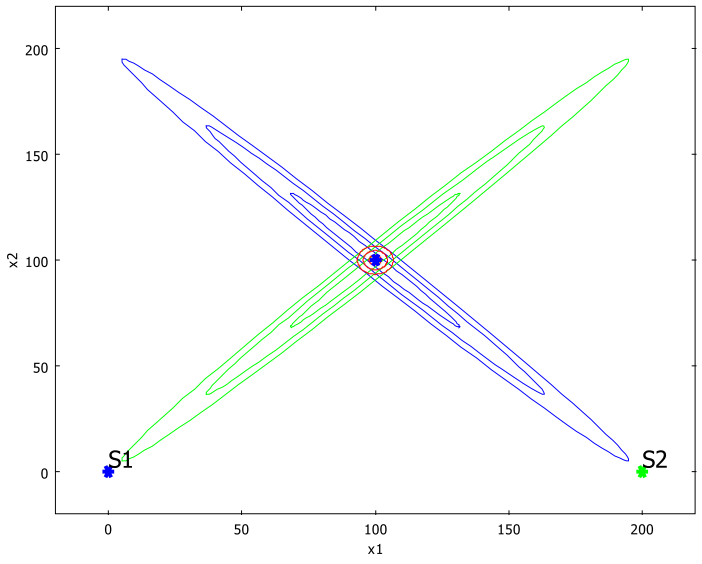
```
V1=[R1;Phi1]
N([141;0.785],[5,0;0,0.1])
Nhat1=estimate(ndist,xhat1)
N([96.2;93.7],[1e+003,-878;-878,1.02e+003])
Nhat2=estimate(ndist,xhat2)
N([104;93.9],[1e+003,878;878,1.04e+003])
xhat=fusion(Nhat1,Nhat2)
N([100;90.6],[98,1.01;1.01,98.6])
plot2(Nhat1,Nhat2,xhat,'legend','')
```



Simulation example 3

Gauss approximation formula applied to the banana transformation gives too optimistic result.

```
y1=[R1;Phi1];
y2=[R2;Phi2];
hinv=inline('[p(1)+x(1,:).*cos(x(2,:)); p(2)+x
Nhat1=tt1eval(y1,hinv,p1)
N([100;100],[1e+003,-1e+003;-1e+003,1e+003])
Nhat2=tt1eval(y2,hinv,p2)
N([100;100],[1e+003,1e+003;1e+003,1e+003])
xhat=fusion(Nhat1,Nhat2)
N([100;100],[4.99,-2.97e-011;-2.97e-011,4.99])
plot2(Nhat1,Nhat2,xhat,'legend','')
```



The unscented transformation

Method for transforming mean and covariance of Y to $X = g(Y)$:

1. The so called *sigma points* $y^{(i)}$ are computed. These are the mean and symmetric deviations around the mean computed from the covariance matrix of y.
2. The sigma points are mapped to $x^{(i)} = h^{-1}\left(y^{(i)}\right)$.
3. The mean and covariance are fitted to the mapped sigma points

$$\mu_x = \sum_{i=1}^{N} \omega_m^i x^{(i)},$$

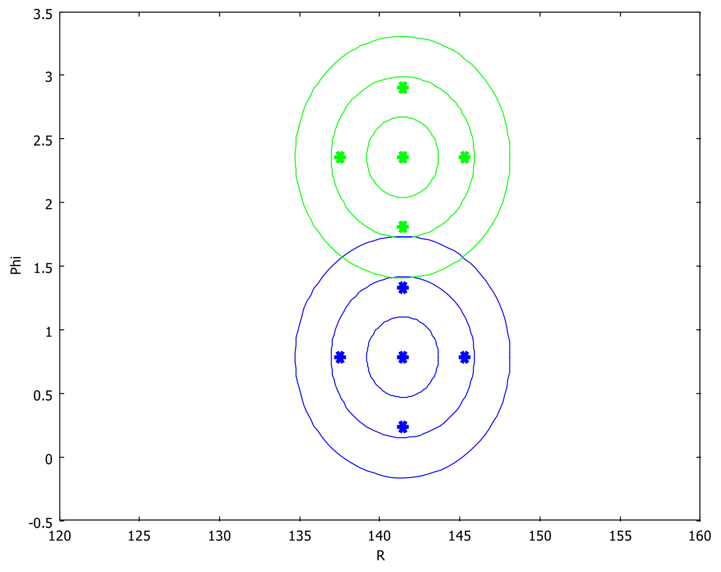
$$P_x = \sum_{i=1}^{N} \omega_c^i \left(x^{(i)} - \mu_x\right)\left(x^{(i)} - \mu_x\right)^T.$$

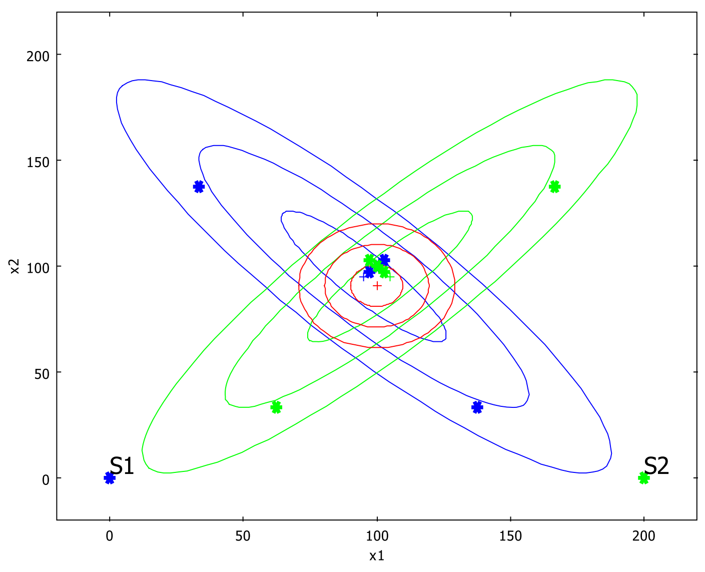
Tricks and rule of thumbs available to tune the weights.
Can be seen as a Monte Carlo method with deterministic sampling.

Simulation example 4

Left: Distribution and sigma points of y

Right: Transformed sigma points and fitted Gaussian distribution





```
[Nhat1,S1,fS1]=uteval(y1,hinv,'std',[],p1)
N([95.1;95.1],[1e+003,-854;-854,1e+003])
S1 =
   141.4214   145.2943   141.4214   137.5484   141.4214
     0.7854     0.7854     1.3331     0.7854     0.2377
fS1 =
   100.0000   102.7386    33.2968    97.2614   137.4457
        100   102.7386   137.4457    97.2614    33.2968
[Nhat2,S2,fS2]=uteval(y2,hinv,'std',[],p2)
N([105;95.1],[1e+003,854;854,1e+003])
S2 =
   141.4214   145.2943   141.4214   137.5484   141.4214
     2.3562     2.3562     2.9039     2.3562     1.8085
fS2 =
        100    97.2614    62.5543   102.7386   166.7032
   100.0000   102.7386    33.2968    97.2614   137.4457
xhat=fusion(Nhat1,Nhat2)
N([100;90.8],[94.9,1.48e-013;-1.48e-013,94.9])
plot2(y1,y2,'legend','')
plot2(Nhat1,Nhat2,xhat,'legend','')
```

Conditionally linear models

$$y_k = h_k(x_n)x_l + e_k, \quad \text{Cov}(e_k) = R_k(x_n),$$

Separable least squares: The WLS solution for x_l is explicitly given by

$$\hat{x}_l^{WLS}(x_n) = \left(\sum_{k=1}^{N} h_k^T(x_n) R_k^{-1}(x_n) h_k(x_n) \right)^{-1} \sum_{k=1}^{N} h_k^T(x_n) R_k^{-1}(x_n) y_k.$$

for each value of x_n. Which one to choose?

- Almost always utilize the separable least squares principle
- In some cases, the loss function $\arg\min_{x_n} V^{WLS}\big(x_n, \hat{x}_l(x_n)\big)$ might have more local minima than the original formulation.

Chapter 4: Sensor networks

Whiteboard:

- The most common sensor models are already presented.

Slides:

- Sensor networks: dedicated measurements and LS solutions.

TOA and RSS

Received signal $y_k(t)$ as a delayed and attenuated version of the transmitted signal $s(t)$

$$y_k(t) = a_k s(t - \tau_k) + e_k(t), \quad k = 1, 2, \dots N,$$

Time-delay estimation using known training signal (pilot symbols) gives

$$r_k = \tau_k v = \|x - p_k\| = \sqrt{(x_1 - p_{k,1})^2 + (x_2 - p_{k,2})^2}.$$

Time-of-arrival (TOA) = transport delay.

Estimation of a_k gives Received Signal Strength (RSS), which does not require known training signal, just transmitter power P_0 and path propagation constant α.

$$P_k = P_0 - \alpha \log\left(\|x - p_k\|\right).$$

TDOA

Common offset r_0

$$r_k = \|x - p_k\| + r_0, \quad k = 1, 2, \ldots, N.$$

Study range differences

$$r_{i,j} = r_i - r_j, \quad 1 \leq i < j \leq N.$$

TDOA maths

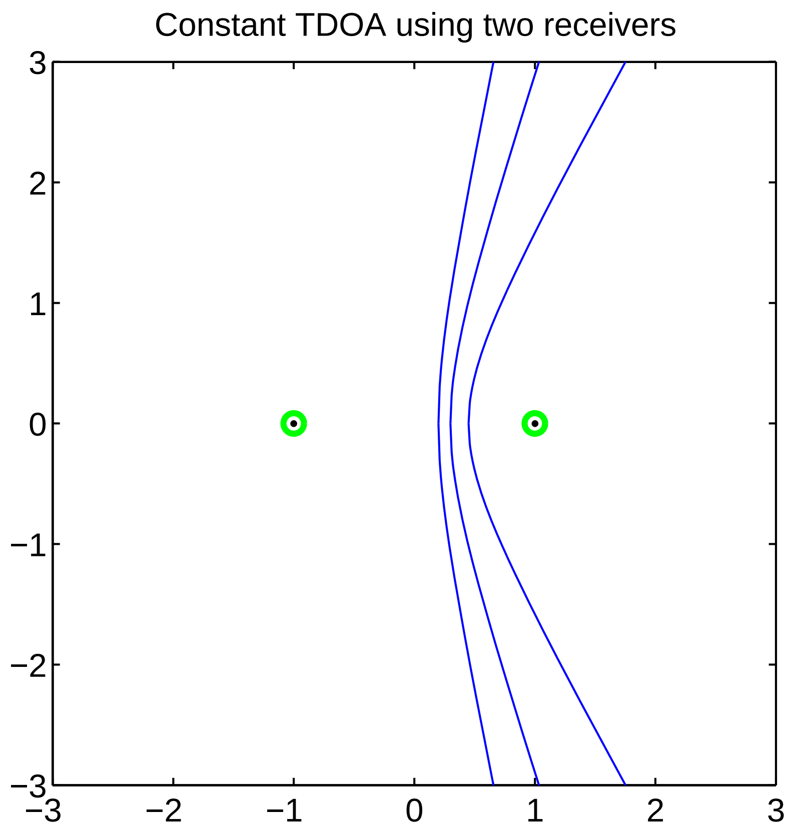
Assume $p_1 = (D/2, 0)^T$ and $p_2 = (-D/2, 0)^T$, respectively. Then

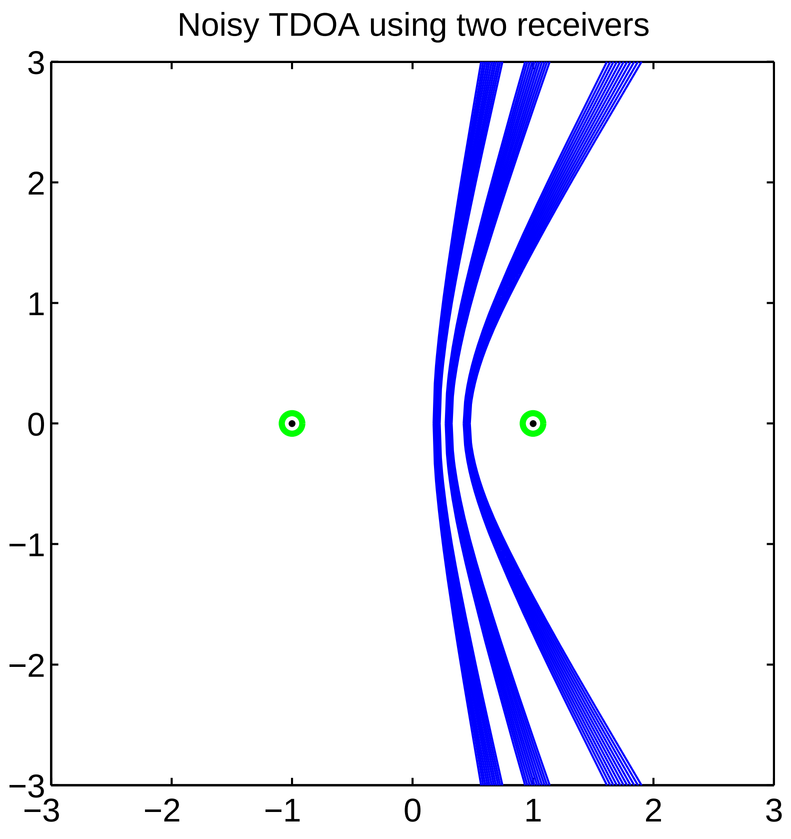
$$\begin{aligned} r_2 &= \sqrt{x_2^2 + (x_1 + D/2)^2}, \\ r_1 &= -\sqrt{x_2^2 + (x_1 - D/2)^2}, \\ r_{12} &= r_2 - r_1 = h(x, D) \\ &= \sqrt{x_2^2 + (x_1 + D/2)^2} - \sqrt{x_2^2 + (x_1 - D)/2)^2}. \end{aligned}$$

Simplify

$$\frac{x_1^2}{a} - \frac{x_2^2}{b} = \frac{x_1^2}{r_{12}^2/4} - \frac{x_2^2}{D^2/4 - r_{12}^2/4} = 1.$$

TDOA illustration





DOA/AOA

The solution to this hyperbolic equation has asymptotes along the lines

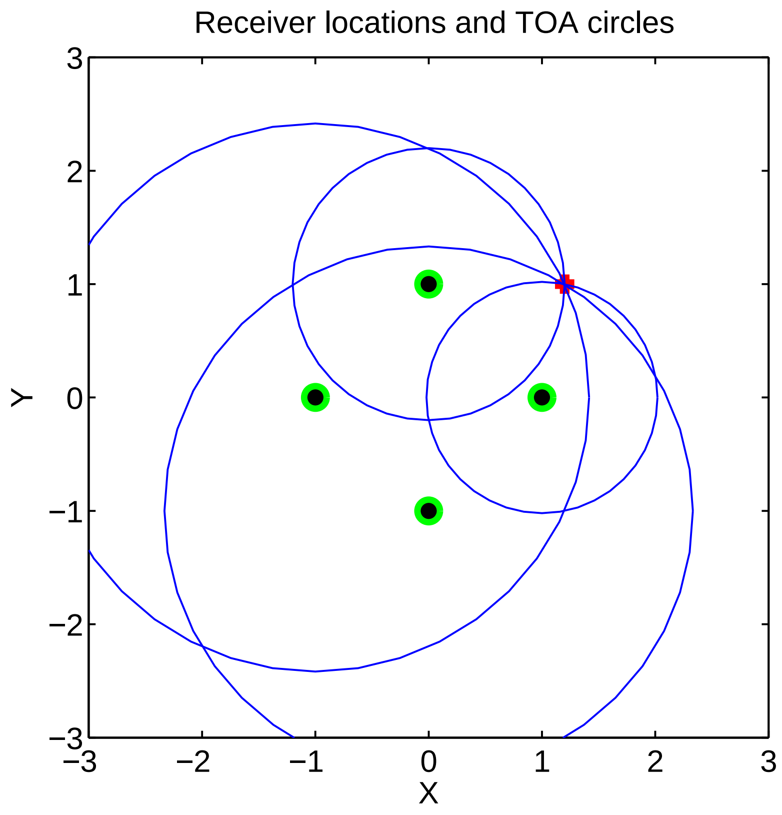
$$x_2 = \pm \frac{b}{a} x_1 = \pm \sqrt{\frac{D^2/4 - r_{12}^2/4}{r_{12}^2/4}} x_1$$
$$= \pm x_1 \sqrt{\left(\frac{D}{r_{12}}\right)^2 - 1}.$$

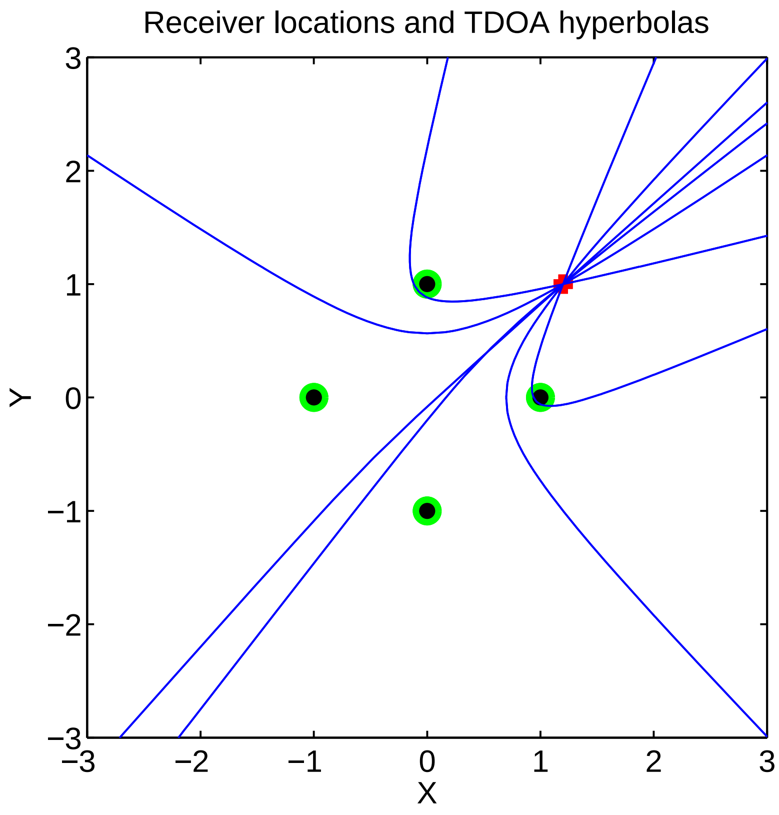
AOA φ for far-away transmitters (the far-field assumptions of planar waves):

$$\varphi = \arctan\left(\sqrt{\left(\frac{D}{r_{12}}\right)^2 - 1}\right)$$

THE example

Noise-free nonlinear relations for TOA and TDOA.



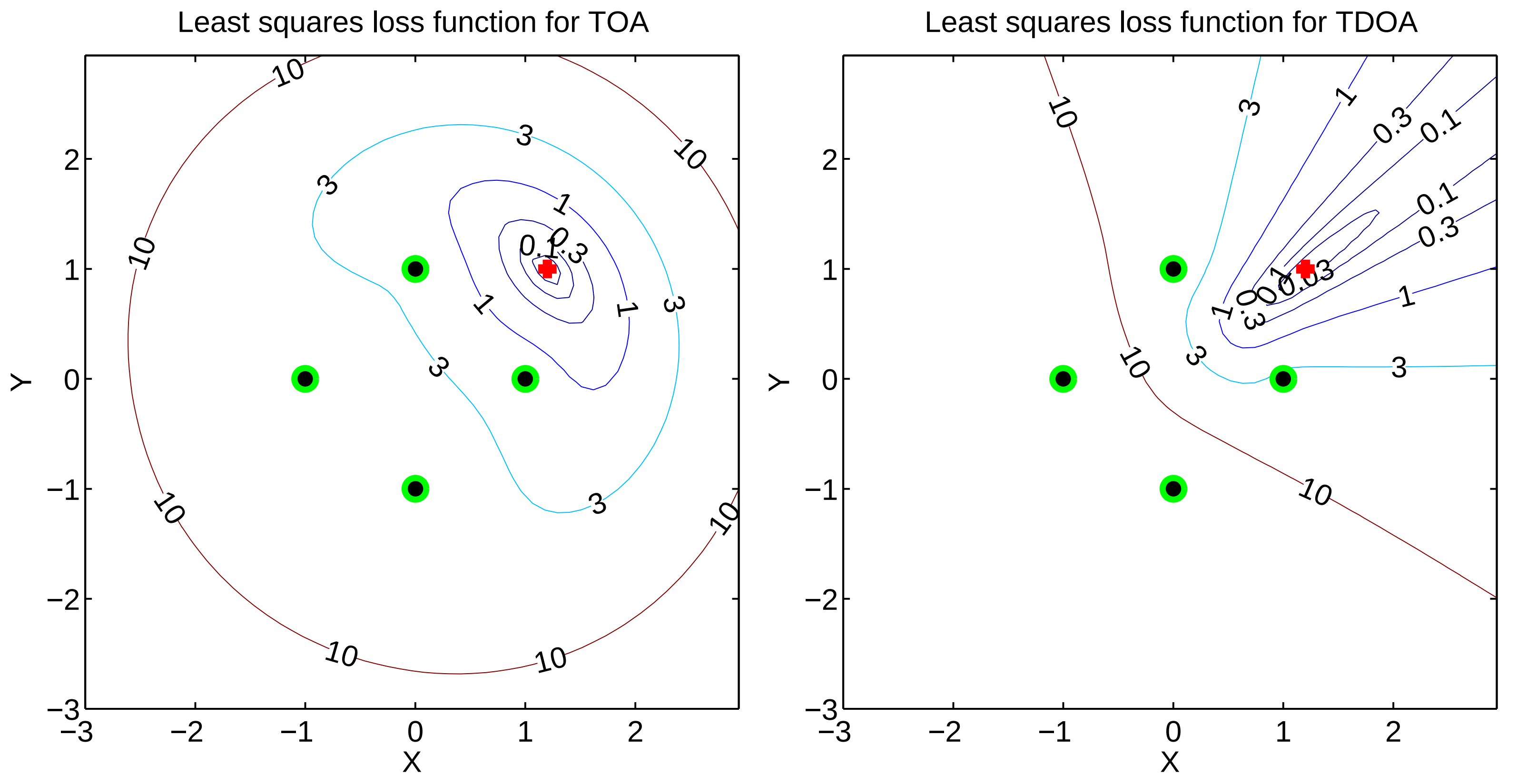


Estimation criteria

NLS	$V^{NLS}(x) = \|\mathbf{y} - \mathbf{h}(x)\|^2 = (\mathbf{y} - \mathbf{h}(x))^T(\mathbf{y} - \mathbf{h}(x))$
WNLS	$V^{WNLS}(x) = (\mathbf{y} - \mathbf{h}(x))^T \mathbf{R}^{-1}(x)(\mathbf{y} - \mathbf{h}(x))$
ML	$V^{ML}(x) = \log\ p_{\mathbf{e}}\big(\mathbf{y} - \mathbf{h}(x)\big)$
GML	$V^{GML}(x) = (\mathbf{y} - \mathbf{h}(x))^T \mathbf{R}^{-1}(x)(\mathbf{y} - \mathbf{h}(x)) + \log\det \mathbf{R}(x)$

THE example

Level curves $V(x)$ for TOA and TDOA.



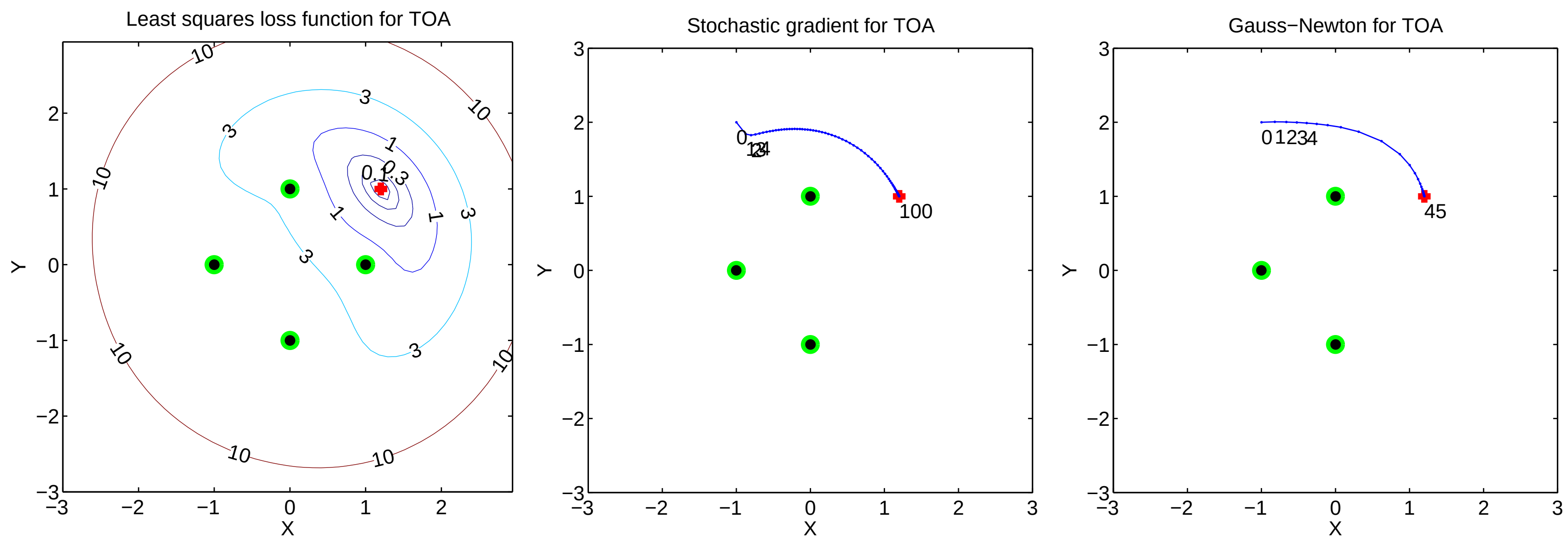
Estimation methods

Steepest descent	$\hat{x}_k = \hat{x}_{k-1} + \mu_k \mathbf{H}^T(\hat{x}_{k-1})R^{-1}(y - \mathbf{h}(\hat{x}_{k-1}))$
Gauss-Newton	$\hat{x}_k = \hat{x}_{k-1} + \mu_k \left(\mathbf{H}^T(\hat{x}_{k-1})R^{-1}\mathbf{H}(\hat{x}_{k-1})\right)^{-1}$ $\mathbf{H}^T(\hat{x}_{k-1})R^{-1}(y - \mathbf{h}(\hat{x}_{k-1}))$

Method	$h(x, p_i)$	$\partial h / dx_1$	$\partial h / \partial x_2$
RSS:	$K + 10\alpha \log_{10} r_i$	$\frac{10\alpha}{\log 10} \frac{x_1 - p_{i,1}}{r_i^2}$	$\frac{10\alpha}{\log 10} \frac{x_2 - p_{i,2}}{r_i^2}$
TOA:	r_i	$\frac{x_1 - p_{i,1}}{r_i}$	$\frac{x_2 - p_{i,2}}{r_i}$
TDOA:	$r_i - r_j$	$\frac{x_1 - p_{i,1}}{D_i} - \frac{x_1 - p_{j,1}}{D_j}$	$\frac{x_2 - p_{i,2}}{D_i} - \frac{x_2 - p_{j,2}}{D_j}$
AOA:	$\alpha_i + \frac{180}{\pi} \arctan \frac{x_2 - p_{i,2}}{x_1 - p_{i,1}}$	$\frac{-(x_1 - p_{i,1})}{r_i^2} \frac{180}{\pi}$	$\frac{x_2 - p_{i,2}}{r_i^2} \frac{180}{\pi}$

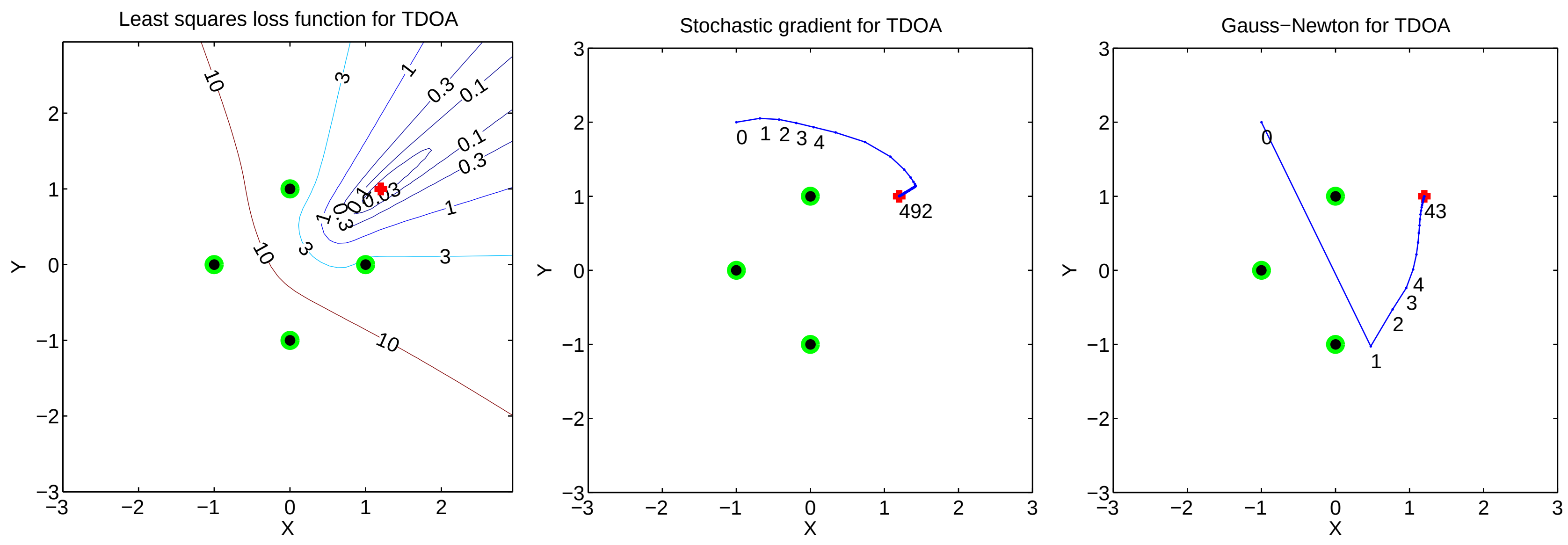
THE example

The steepest descent and Gauss-Newton algorithms for TOA.



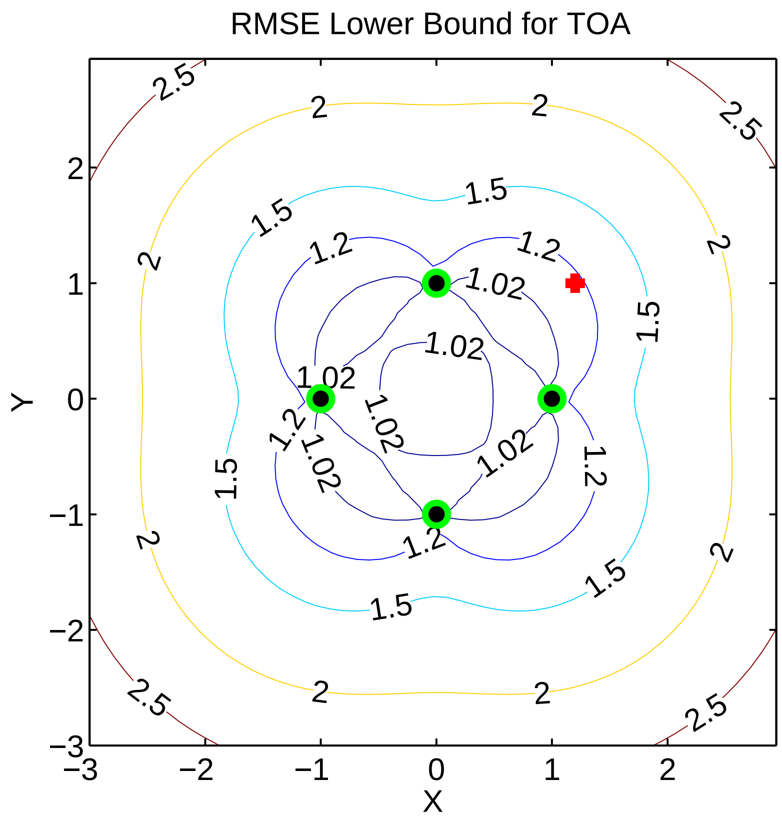
THE example

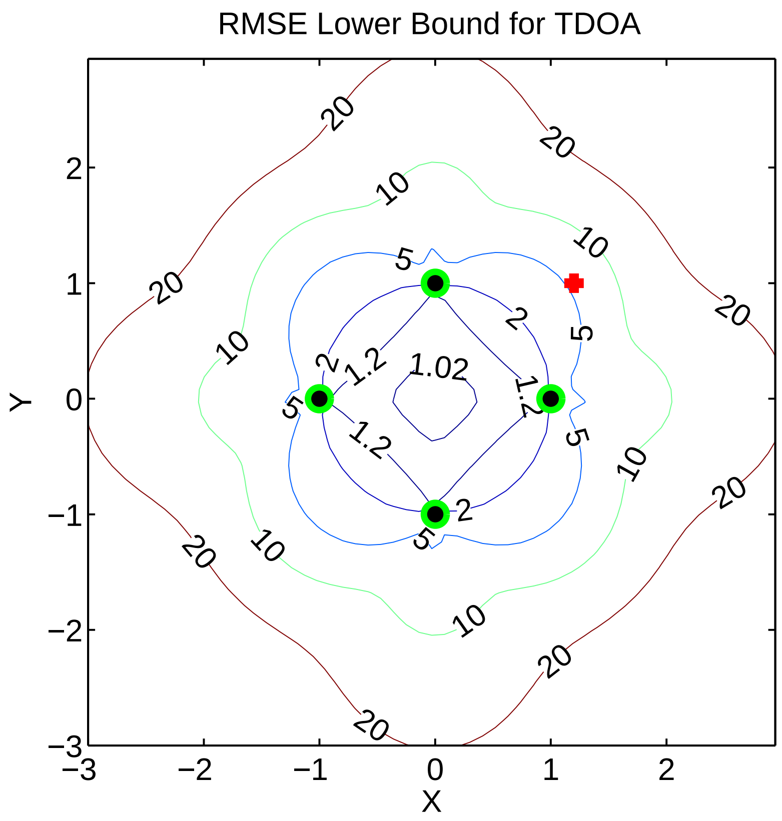
The steepest descent and Gauss-Newton algorithms for TDOA.



THE example

CRLB for TOA and TDOA: RMSE $(\hat{x}) \geq \sqrt{\operatorname{tr}\left(\mathcal{I}^{-1}\right)}$





Dedicated explicit LS solutions

Basic trick: study NLS of **squared** distance measurements:

$$\hat{x} = \arg\min_x \sum_{k=1}^{N} \left(r_k^2 - \|x - p_k\|^2\right)^2 .$$

Note: what does this imply for the measurement noise?

TOA approach 1: range parameter

Squared range model, assuming additive noise on the squares

$$\begin{aligned} r_k^2 &= \|x - p_k\|^2 = (x_1 - p_{k,1})^2 + (x_2 - p_{k,2})^2 + e_k \\ &= -2x_1 p_{2,1} - 2x_2 p_{2,2} + \|p_k\|^2 + \|x\|^2 + e_k. \end{aligned}$$

This can be seen as a linear model

$$y_k = \varphi_k \theta + e_k,$$

where

$$\begin{aligned} y_k &= r_k^2 - \|p_k\|^2, \\ \varphi_k &= (-2p_{k,1}, -2p_{k,e}, 1)^T, \\ \theta &= (x_1, x_2, \|x\|^2)^T. \end{aligned}$$

Using the standard least squares method, this gives

$$\hat{\theta} = \left(\sum_{k=1}^{M} \varphi_k \varphi_k^T \right)^{-1} \sum_{k=1}^{M} \varphi_k y_k,$$

$$P = \sigma_e^2 \left(\sum_{k=1}^{M} \varphi_k \varphi_k^T \right)^{-1},$$

$$\hat{\theta} \sim \mathcal{N}(\theta, P).$$

Either ignore the range estimate $\hat{\theta}_3$, or apply one of the methods using $x = h^{-1}(y)$, where $y = \hat{\theta}$.

TOA approach 2: reference sensor

Assume first sensor at the origin:

$$r_1^2 = \|x\|^2 = x_1^2 + x_2^2 + e_1,$$
$$r_2^2 = \|x - p_2\|^2 = (x_1 - p_{2,1})^2 + (x_2 - p_{2,2})^2 + e_2.$$

Complete the squares and subtract the first equation from the second

$$r_2^2 - r_1^2 = -2p_{2,1}x_1 - 2p_{2,2}x_2 + p_{2,1}^2 + p_{2,2}^2.$$

This is one linear relation.

Continuing with more than two sensors give a linear model

$$\mathbf{y} = \mathbf{H}x + \mathbf{e},$$

$$\mathbf{y} = \begin{pmatrix} r_2^2 - r_1^2 - \|p_2\|^2 \\ r_3^2 - r_1^2 - \|p_3\|^2 \\ \vdots \\ r_N^2 - r_1^2 - \|p_N\|^2 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} -2p_{2,1} & -2p_{2,2} \\ -2p_{3,1} & -2p_{3,2} \\ \vdots & \\ -2p_{N,1} & -2p_{N,2} \end{pmatrix}, \quad \mathbf{e} = \begin{pmatrix} e_2 - e_1 \\ e_3 - e_1 \\ \vdots \\ e_N - e_1 \end{pmatrix}.$$

TDOA: reference sensor with range parameter

Study range differences

$$r_{i,1} = r_i - r_1,\ i > 1,$$

$$(r_{i,2} + r_1)^2 = -2p_{2,1}x_1 - 2p_{2,2}x_2 + p_{2,1}^2 + p_{2,2}^2 + r_1^2.$$

Use r_1 as a scalar parameter

$$\mathbf{y} = \mathbf{H}x + \mathbf{G}r_1 + \mathbf{e},$$

$$\mathbf{y} = \begin{pmatrix} r_{i,2}^2 - \|p_2\|^2 \\ r_{i,3}^2 - \|p_3\|^2 \\ \vdots \\ r_{i,N}^2 - \|p_N\|^2 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} -2p_{2,1} & -2p_{2,2} \\ -2p_{3,1} & -2p_{3,2} \\ \vdots & \\ -2p_{N,1} & -2p_{N,2} \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} r_{2,1} \\ r_{3,1} \\ \vdots \\ r_{N,1} \end{pmatrix}.$$

Separable least squares gives the solution as a function of r_1:

$$\hat{x}(r_1) = \left(\mathbf{H}^T \mathbf{H}\right)^{-1} \mathbf{H}^T \left(\mathbf{y} - \mathbf{G} r_1\right).$$

We now just have to tune the scalar r_1 such that

$$r_1^2 = \|\hat{x}(r_1)\|^2 = \hat{x}^T(r_1)\hat{x}(r_1).$$

There is an explicit solution!

AOA triangulation

Linear model immediate

$$
\begin{aligned}
\varphi_k &= \arctan\left(\frac{x_2 - p_{k,2}}{x_1 - p_{k,1}}\right), \\
(x_1 - p_{k,1})\tan(\varphi_k) &= x_2 - p_{k,2} \\
\mathbf{y} &= \mathbf{H}x + \mathbf{e}, \\
\mathbf{y} &= \begin{pmatrix} p_{1,1}\tan(\varphi_1) - p_{1,2} \\ p_{2,1}\tan(\varphi_2) - p_{2,2} \\ \vdots \\ p_{N,1}\tan(\varphi_N) - p_{N,2} \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} \tan(\varphi_1) & -1 \\ \tan(\varphi_2) & -1 \\ \vdots & \\ \tan(\varphi_N) & -1 \end{pmatrix}.
\end{aligned}
$$

RSS

Received signal strength (RSS) observations:

- All waves (radio, radar, IR, seismic, acoustic, magnetic) decay exponentially in range:
- Receiver k measures energy/power/signal strength for wave i.

$$P_{k,i} = P_{0,i} \|x - p_k\|^{n_{p,i}}.$$

- Transmitted signal strength and path loss constant unknown.
- Communication constraints make coherent detection from the signal waveform impossible.
- Compare $P_{k,i}$ for different receivers

Log model:

$$\bar{P}_{k,i} = \bar{P}_{0,i} + n_{p,i} \underbrace{\log\big(\|x - p_k\|\big)}_{\triangleq c_k(x)},$$

$$y_{k,i} = \bar{P}_{k,i} + e_{k,i}.$$

Use separable least squares to eliminate path loss constant and transmitted power for wave i.

$$\begin{aligned}
\widehat{(x,\theta)} &= \arg\min_{x,\theta} V(x,\theta), \\
V(x,\theta) &= \sum_{i=1}^{M} \sum_{k=1}^{N} \frac{\big(y_{k,i} - h(c_k(x),\theta_i)\big)^2}{\sigma_{P,i}^2}, \\
h(c_k(x),\theta_i) &= \theta_{i,1} + \theta_{i,2} c_k(x), \\
c_k(x) &= \log\big(\|x - p_k\|\big).
\end{aligned}$$

Finally, use NLS to optimize over 2D target position x.

Chapter 5: Detection problems

Whiteboard:

- Detection notation overview
- Neyman-Pearson’s lemma
- Detection tests for no model, linear model and nonlinear model

Slides:

- Sensor network examples

Tests

Hypothesis test in statistics:

$$\begin{aligned} H_0 &: \mathbf{y} = \mathbf{e}, \\ H_1 &: \mathbf{y} = x + \mathbf{e}, \\ &\mathbf{e} \sim p(\mathbf{e}). \end{aligned}$$

General model-based test (sensor clutter versus target present):

$$\begin{aligned} H_0 &: \mathbf{y} = \mathbf{e}^0, & \mathbf{e}^0 &\sim p^0(\mathbf{e}^0), \\ H_1 &: \mathbf{y} = \mathbf{h}(x) + \mathbf{e}^1, & \mathbf{e}^1 &\sim p^1(\mathbf{e}^1). \end{aligned}$$

Special case: Linear model $\mathbf{h}(x) = \mathbf{H}x$.

First example, revisited

Detect if a target is present.

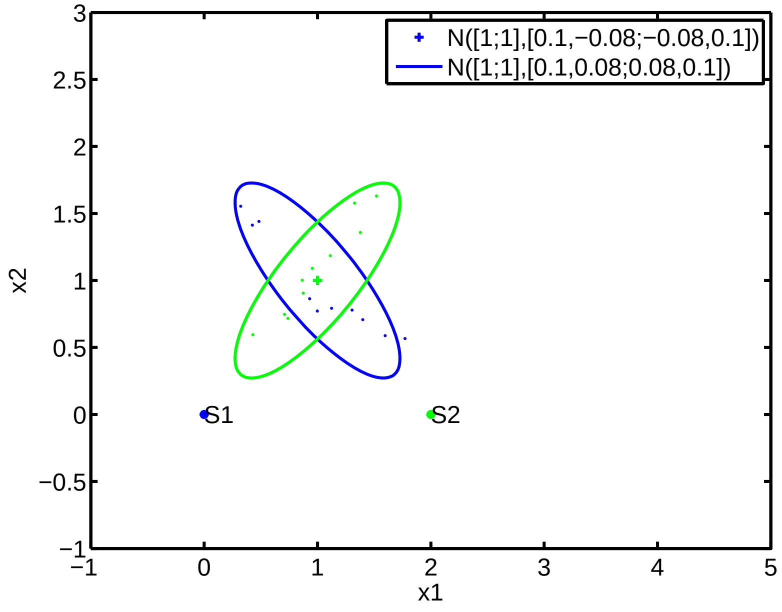
$$y_1 = x + e_1, \quad \text{Cov}(e_1) = R_1$$
$$y_2 = x + e_2, \quad \text{Cov}(e_2) = R_2$$
$$\mathbf{y} = \mathbf{H}x + \mathbf{e}, \quad \text{Cov}(\mathbf{e}) = \mathbf{R} \quad \mathbf{H} = \begin{pmatrix} I \\ I \end{pmatrix}$$
$$\mathbf{R} = \begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix}$$
$$T(\mathbf{y}) = \mathbf{y}^T \mathbf{R}^{-T/2} \Pi \mathbf{R}^{-1/2} \mathbf{y},$$
$$\Pi = \mathbf{R}^{-T/2} \mathbf{H} (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H} \mathbf{R}^{-1/2} = \begin{bmatrix} 22.8 & 22.2 & 5.0 & 0.0 \\ 22.2 & 22.8 & 0.0 & 5.0 \\ 5.0 & 0.0 & 22.8 & -22.2 \\ -0.0 & 5.0 & -22.2 & 22.8 \end{bmatrix}$$

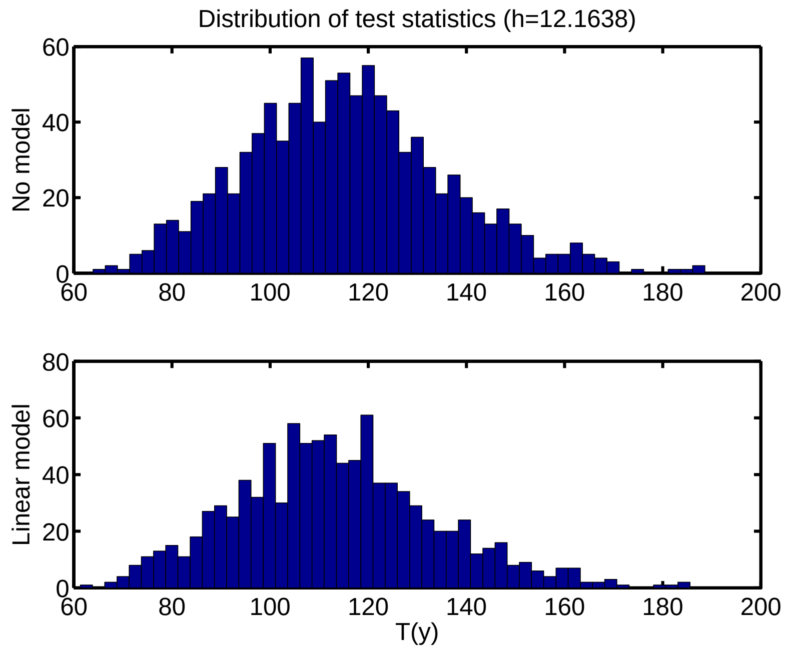
Numerical simulation

Threshold for the test:

```
h=erfinv(chi2dist(2),0.999)
h =
10.1405
```

Note: for no model (standard statistical test), $\Pi = I$. Both methods perform perfect $P_D = 1$.





Chapter 6

Whiteboard:

- A general algorithm based on estimation and fusion.
- Application to a linear model $\Rightarrow$ The Kalman filter.
- Bayes' optimal solution.

Slides:

- Summary of model and Bayes recursions for optimal filtering
- Particular nonlinear filters with explicit solutions.
- CRLB.

State space models

Nonlinear model:

$$\begin{aligned} x_{k+1} &= f(x_k, v_k) & \quad &\text{or} \quad p(x_{k+1}|x_k), \\ y_k &= h(x_k, e_k) & \quad &\text{or} \quad p(y_k|x_k). \end{aligned}$$

Nonlinear model with additive noise:

$$\begin{aligned} x_{k+1} &= f(x_k) + v_k, & \quad &\text{or} \quad p(x_{k+1}|x_k) = p_{v_k}\big(x_{k+1} - f(x_k)\big), \\ y_k &= h(x_k) + e_k, & \quad &\text{or} \quad p(y_k|x_k) = p_{e_k}\big(y_k - h(x_k)\big), \end{aligned}$$

Linear model:

$$\begin{aligned} x_{k+1} &= F_k x_k + v_k, \\ y_k &= H_k x_k + e_k. \end{aligned}$$

Gaussian model: $v_k \sim \mathcal{N}(0, Q_k)$, $e_k \sim \mathcal{N}(0, R_k)$ and $x_0 \sim \mathcal{N}(0, P_0)$

Bayes solution for nonlinear model with additive noise

$$\alpha = \int_{\mathcal{R}^{n_y}} p_{e_k}(y_k - h(x_k)|x_k) p(x_k|y_{1:k-1})\, de_k,$$

$$p(x_k|y_{1:k}) = \frac{1}{\alpha} p_{e_k}(y_k - h(x_k)) p(x_k|y_{1:k-1})$$

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

Need a model that keeps the same form of the posterior during

- The nonlinear transformation $f(x_k)$.
- The addition of $f(x_k)$ and v_k.
- The inference of x_k from y_k done in the measurement update.

Conjugate priors

The posterior gets the same form after the measurement update $p(x_k|y_{1:k}) \propto p_{e_k}(y_k - h(x_k))p(x_k|y_{1:k-1})$ in the following cases:

Likelihood	Conjugate prior
Normal unknown mean and known covariance	Normal
Normal, known mean unknown variance	inverse Wishart
Uniform	Pareto
Binomial	Beta
Exponential	Gamma

The normal distribution is the only multivariate distribution that works here.

Toy example with analytic solution

Let the model be

$$x_{k+1} = Fx_k,$$
$$y_k \sim \exp(x_k).$$

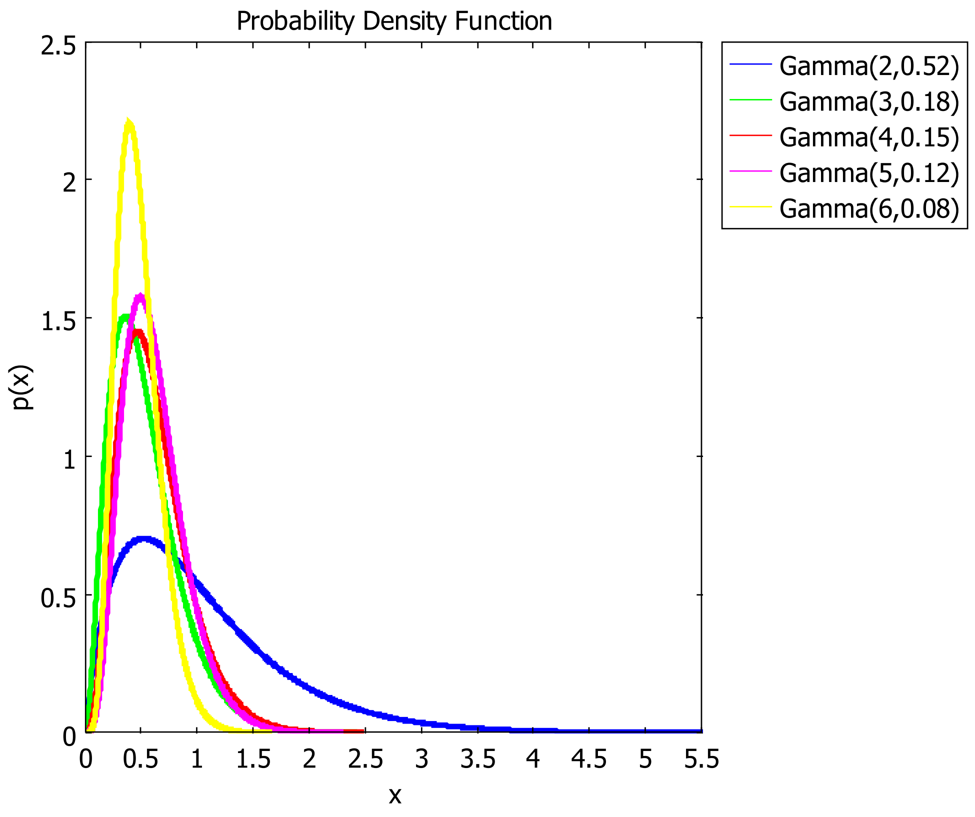
The model is setup and simulated below.

```
lh=expdist;
x0=0.4;
x(1)=x0;
F=0.9;
for k=1:5
   y(k)=rand(expdist(1/x(k)),1);
   x(k+1)=F*x(k);
end
```

y_k is taken at random from an exponential distrubution $\exp(0.4^k x_0)$, where only x_0 is unknown.

The conjugate prior to the exponential likelihood is the Gamma distribution. The gamma distribution is also closed under multiplication.

```
p{1}=gammadist(1,1);
for k=1:5
   p{k}=posterior(p{k},lh,y(k));  % Measurement update
   p{k+1}=F*p{k};                 % Time update
end
plot(p{1:5})
```


Probability Density Function
p(x)
x
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
Gamma(2,0.52)
Gamma(3,0.18)
Gamma(4,0.15)
Gamma(5,0.12)
Gamma(6,0.08)

Practical cases with analytic solution

Bayes solution can be represented with finite dimensional statistics analytically in the following cases:

- The Kalman filter
- HMM
- Gaussian mixture (but with exponential complexity in time)

The Kalman filter

Linear model, Gaussian prior, process noise and measurement noise $\Rightarrow$ Gaussian posterior in all steps.

Recursion for the mean and covariance already given.

Usual form without information matrix:

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

The HMM model

ξ is the unknown stochastic variable, the *mode*, which takes on integer values $1, 2, \dots, m$.
The observation y_k is an integer $1, 2, \dots, m$ related to ξ with probability $H^{(y_k, \xi)}$.
The model is specified by transition probability matrix F and observation probability matrix H.

$$\begin{aligned} x_{k+1} &= F x_k, \\ x_k^{(m)} &= \mathrm{P}(\xi_k = m), \quad m = 1, 2, \dots, n_x, \\ \mathrm{P}(y_k = i | \xi_k = j) &= H^{(i,j)}, \quad i = 1, 2, \dots, n_y, \end{aligned}$$

$$\mathrm{P}(y_k = i) = \sum_{j=1}^{n_x} H^{(i,j)} \, \mathrm{P}(\xi_k = j) = \sum_{j=1}^{n_x} H^{(i,j)} x_k^{(j)}, \quad i = 1, 2, \dots, n_y.$$

The HMM filter

The time and measurement update of the optimal filter are

$$\pi^i_{k|k-1} = \sum_{j=1}^{n_x} p(\xi_k = i|\xi_{k-1} = j)p(\xi_{k-1} = j|y_{k-1}) = \sum_{j=1}^{n_x} \pi^j_{k-1|k-1} F^{(i,j)},$$

$$\pi^i_{k|k} = \frac{p(\xi_k = i|y_{k-1})p(y_k|\xi_k = i)}{p(y_k|y_{k-1})} = \frac{\pi^i_{k|k-1} H^{(y_k,i)}}{\sum_{j=1}^{n_x} \pi^j_{k|k-1} H^{(y_k,j)}}.$$

The Gaussian mixture model (Jump Markov Linear Model)

General linear model with discrete mode parameter ξ, which takes on values $1, 2, \dots m$:

$$\begin{aligned}
x_{k+1} &= F(\xi_k)x_k + v_k(\xi_k),\\
y_k &= H(\xi_k)x_k + e_k(\xi_k),\\
v_k(\xi_k) &\in \mathcal{N}(\mu_v(\xi_k), Q(\xi_k)),\\
e_k(\xi_k) &\in \mathcal{N}(\mu_e(\xi_k), R(\xi_k)),\\
x_0(\xi_k) &\in \mathcal{N}(\mu_{x_0}(\xi_k), P_{x_0}(\xi_k)).
\end{aligned}$$

Note, a Gaussian mixture can approximate any PDF arbitrarily well.

The Gaussian mixture filter

Solution

$$
\begin{aligned}
p(x_k|y_{1:k}) &= \frac{\sum_{\xi_{1:k}} p(\xi_{1:k}|y_{1:k})p(x_k|y_{1:k},\xi_{1:k})}{\sum_{\xi_{1:k}} p(\xi_{1:k}|y_{1:k})} \\
&= \frac{\sum_{\xi_{1:k}} p(\xi_{1:k}|y_{1:k})N\left(\hat{x}_{k|k}(\xi_{1:k}), P_{k|k}(\xi_{1:k})\right)}{\sum_{\xi_{1:k}} p(\xi_{1:k}|y_{1:k})}, \\
p(\xi_{1:k}|y_{1:k}) &= \frac{p(y_{1:k}|\xi_{1:k})p(\xi_{1:k})}{p(y_{1:k})}.
\end{aligned}
$$

The number of mixture components $\xi_{1:k}$ increases exponentially.

General approximation approaches

1. Approximate the model to a case where an optimal algorithm exists.
   (a) Extended KF (EKF) which approximates the model with a linear one.
   (b) Unscented KF and EKF2 that apply higher order approximations.
2. Approximate the optimal nonlinear filter for the original model.
   (a) Point-mass filter (PMF) which uses a *regular* grid of the state space and applies the Bayesian recursion.
   (b) Particle filter (PF) which uses a *random* grid of the state space and applies the Bayesian recursion.

Parametric CRLB

- The parametric CRLB gives a lower bound on estimation error for a fixed trajectory $x_{1:k}$. That is, $\operatorname{Cov}(\hat{x}_{k|k} \leq P^{\text{CRLB}}_{k|k}$.
- Algorithm identical to Riccati equation in KF, where the gradients are evaluated along the trajectory $x_{1:k}$:

$$\begin{aligned}
P_{k+1|k} &= F_k P_{k|k} F_k^T + G_k Q_k G_k,\\
P_{k+1|k+1} &= P_{k+1|k} - P_{k+1|k} H_k^T (H_k P_{k+1|k} H_k^T + R_k)^{-1} H_k P_{k+1|k},\\
F_k^T &= \frac{\partial}{\partial x_k} f^T(x_k, v_k),\\
G_k^T &= \frac{\partial}{\partial v_k} f^T(x_k, v_k),\\
H_k^T &= \frac{\partial}{\partial x_k} h^T(x_k, v_k).
\end{aligned}$$

Posterior CRLB

- Average over all possible trajectories $x_{1:k}$ with respect to v_k.
- Much more complicated expressions.
- For linear system, the parametric and posterior CRLB coincide.

Chapter 7: The Kalman filter

Whiteboard:

- Derivation of KF using Lemma 7.1
- Derivation of EKF using Taylor expansion of the model
- Derivation of EKF, UKF using Lemma 7.1.

Slides:

- KF properties and practical aspects
- Distributed implementations of KF
- KF code example

The Kalman filter

Time-varying state space model:

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

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, innovation covariance and Kalman gain

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

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

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

Optimality properties

- If x_0, v_k, e_k are Gaussian variables, then

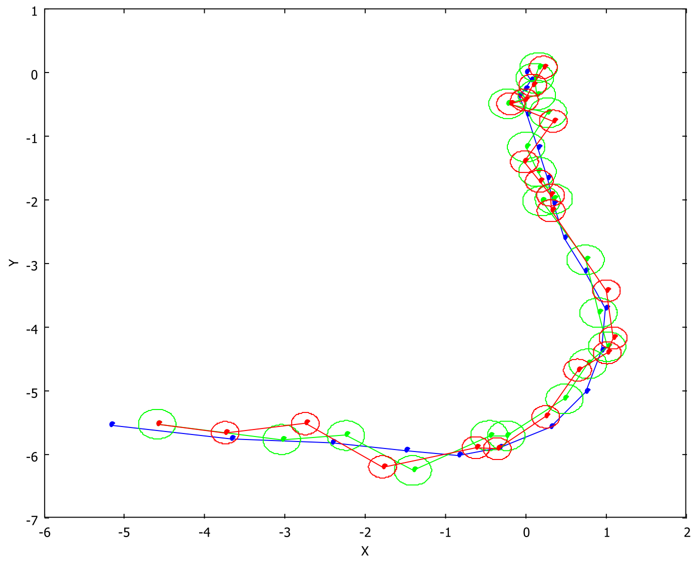
$$\begin{aligned} x_{k+1}|y_{1:k} &\in \mathcal{N}(\hat{x}_{k+1|k}, P_{k+1|k}) \\ x_k|y_{1:k} &\in \mathcal{N}(\hat{x}_{k|k}, P_{k|k}) \\ \varepsilon_k &\in \mathcal{N}(0, S_k) \end{aligned}$$

- If x_0, v_k, e_k are Gaussian variables, then the Kalman filter is MV. That is, the best possible estimator among all *linear and nonlinear* ones.
- Independently of the distribution of x_0, v_k, e_k, the Kalman filter is BLUE. That is, the best possible *linear* filter in the unconditional meaning.

Simulation example

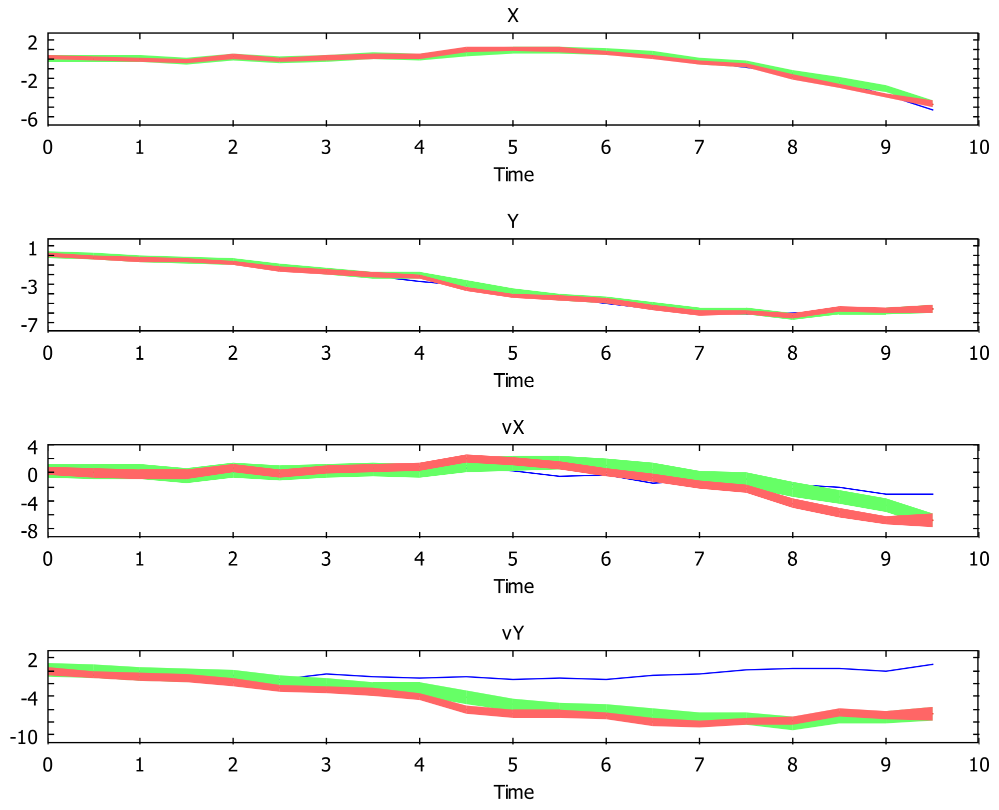
Create a constant velocity model, simulate and Kalman filter

```
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])
```



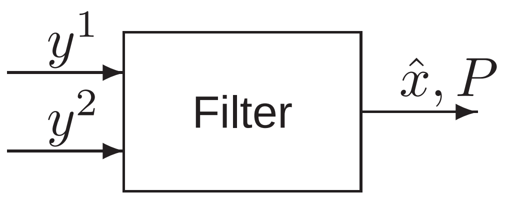
Covariance illustrated as confidence ellipsoids in 2D plots or confidence bands in 1D plots.

```
xplot(z,xhat1,xhat4,'conf',99)
```

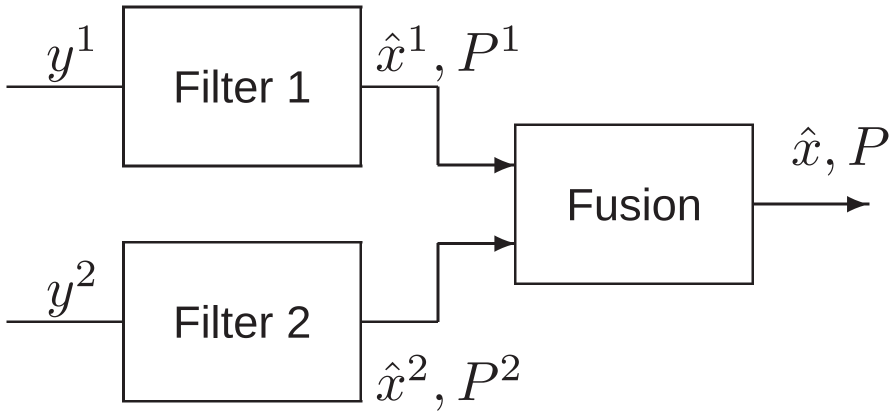


Distributed Filtering

Centralized filtering



Decentralized filtering



Decentralized filtering often required in distributed sensor networks.
Advantage: flexible solution.
Disadvantage: heavy signaling. Unpredictable communication delays.
Practical constraint: built-in KF in sensors.

Centralized Filtering: Simple concept. Just concatenate the measurements.

$$y_k = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_m \end{pmatrix} x_k + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{pmatrix}$$

Decentralized Filtering: Problem with multiple time updates in the network, only one is done in the centralized solution.
SF formula gives too small covariance.
The KF on information form solves the data handling by recovering the individual sensor information.
Risk for information loops, safe fusion is needed.

Out-of-sequence measurements

Three cases of increasing difficulty:

1. At time n it is known that a measurement is taken somewhere, H_n is known, but the numeric value y_n comes later. Typical case: remote sensor.
2. At time n it is known that a measurement is taken somewhere, but both H_n and the numeric value y_n come later. Typical case: computer vision algorithms need time for finding features and convert these to a measurement relation.
3. All of n, H_n and y_n arrive late. Typical case: asynchronous sensor networks.

Note: n might denote non-uniform sampling times t_n here.

Out-of-sequence measurements: case 1

$$\begin{aligned}\hat{x}_{k|k} &= F_k \hat{x}_{k-1|k-1} + K_k(y_k - H_k F_k \hat{x}_{k-1|k-1}) \\ &= (I - K_k H_k) F_k \hat{x}_{k-1|k-1} + K_k y_k \\ &= \sum_{m=0}^{k} \prod_{l=m+1}^{k} (I - K_l H_l) F_l K_m y_m + \prod_{l=0}^{k} (I - K_l H_l) F_l \hat{x}_{0|0}.\end{aligned}$$

Algorithm outline:

1. Replace y_n with its expected value $H_n \hat{x}_{n|n-1}$. This corresponds to doing nothing in the Kalman filter.
2. Compute the covariance update as if y_n was available.
3. When y_n arrives, compensate with the term

$$\prod_{l=n+1}^{k} (I - K_l H_l) F_l K_n (y_n - H_n \hat{x}_{n|n-1})$$

Independent sensors

Assume M independent sensors, so R is block-diagonal.
The complexity of the KF can be reduced.
Two cases: information filter and standard KF.

The information form gives

$$P_{k|k}^{-1} = P_{k|k-1}^{-1} + H_k^T R_k^{-1} H_k = P_{k|k-1}^{-1} + \sum_{i=1}^{M} (H_k^i)^T (R_k^{ii})^{-1} H_k^i.$$

Thus, only small matrices R_k^{ii} need to be inverted.

Independent sensors: the iterated Kalman filter

Similarly, the block form gives that the Kalman filter can be written for sensor $i = 1, 2, \ldots, M$

$$
\begin{aligned}
\hat{\mathbf{x}}_k^0 &= \hat{\mathbf{x}}_{k|k-1}, \\
P_k^0 &= P_{k|k-1}, \\
K_k^i &= P_k^{i-1}(H_k^i)^T (H_k^i P_k^{i-1} (H_k^i)^T + R_k^{ii})^{-1}, \\
P_{k|k}^i &= P_k^{i-1} - K_k^i H_k^i P_k^{i-1}, \\
\hat{\mathbf{x}}_{k|k}^i &= \hat{\mathbf{x}}_k^{i-1} + K_k^i (y_k^i - H_k^i \hat{\mathbf{x}}_k^{i-1}), \\
\hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_k^M, \\
P_{k|k} &= P_k^M.
\end{aligned}
$$

That is, the measurement update is iterated for each sensor.

Robustness and sensitivity

- 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.

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,

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

3. The covariance matrix P_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 $S_k = \mathsf{E}(\varepsilon_k \varepsilon_k^T)$ (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

If $\varepsilon_k \in \mathcal{N}(0, S_k)$, 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 $T(y_k) > h_\alpha$, 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

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

To get a more confident test (P_{FA} vs P_D), a sliding window can be used

$$T(y_{k-L+1:k}^i) = \sum_{k=k-L+1}^{t} \varepsilon_k^{i,T} S_k^{-1} \varepsilon_k^i \sim \chi^2(L \dim(y_k^i)).$$

The measurement update can be postponed according to the principle of out of sequence measurements.

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

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

filter divergence can be concluded, and the filter 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 $\mathsf{E}(\theta) = \hat{\theta}$ and $\mathsf{Cov}(\theta) = P_\theta$.

The state estimate $\hat{x}_k$ is as a stochastic variable a function of four stochastic sources. A Taylor expansion gives

$$\mathsf{Cov}(\hat{x}_k) = P_k + \frac{d\hat{x}_k}{d\theta} P_\theta \left(\frac{d\hat{x}_k}{d\theta} \right)^T .$$

The gradient $d\hat{x}_k/d\theta$ can be computed numerically by simulations.