Lecture 6: Recursive Nonlinear Non-Gaussian Estimation, Part II

AER1513: State Estimation

Timothy D. Barfoot

University of Toronto

Outline

UNIVERSITY OF TORONTO

Motivation

- last lecture we looked at recursive estimation techniques for nonlinear non-Gaussian situations
- we derived the Bayes filter, generalized Gaussian filter, and finally the IEKF and EKF
- this lecture, we will continue to explore some other options in our filter taxonomy that avoid the use of linearization
- we'll introduce the sigmapoint transformation as a replacement to linearization, and use it to derive the sigmapoint Kalman filter
- we'll then return to the Bayes filter and branch off in a different way to derive the particle filter

UNIVERSITY OF TORONTO

System

- we define our system using the following nonlinear, time-varying models:

$$\begin{aligned} \text{motion model:} \quad \mathbf{x}_k &= \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k\right), \quad && k = 1 \ldots K \quad \text{(1a)} \\ \text{observation model:} \quad \mathbf{y}_k &= \mathbf{g}\left(\mathbf{x}_k, \mathbf{n}_k\right), \quad && k = 0 \ldots K \quad \text{(1b)} \end{aligned}$$

  where k is again the discrete-time index and K its maximum

- the variables have the following meanings:

$$\begin{aligned} \text{system state :} \quad & \mathbf{x}_k \in \mathbb{R}^N \\ \text{initial state :} \quad & \mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}\left(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0\right) \\ \text{input :} \quad & \mathbf{v}_k \in \mathbb{R}^N \\ \text{process noise :} \quad & \mathbf{w}_k \in \mathbb{R}^N \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_k\right) \\ \text{measurement :} \quad & \mathbf{y}_k \in \mathbb{R}^M \\ \text{measurement noise :} \quad & \mathbf{n}_k \in \mathbb{R}^M \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right) \end{aligned}$$

UNIVERSITY OF TORONTO

Markov property

$$\begin{tikzcd}
\check{\mathbf{x}}_0 \arrow[d] & & \mathbf{v}_1 \arrow[d] & & & \mathbf{v}_{k-1} \arrow[d] & & \mathbf{v}_k \arrow[d] & & \\
\boxed{+} \arrow[r] & \mathbf{x}_0 \arrow[r] \arrow[dd] & \boxed{\mathbf{f}} \arrow[r] & \mathbf{x}_1 \arrow[r] \arrow[dd] & \cdots \arrow[r] & \boxed{\mathbf{f}} \arrow[r] & \mathbf{x}_{k-1} \arrow[r] \arrow[dd] & \boxed{\mathbf{f}} \arrow[r] & \mathbf{x}_k \arrow[r] \arrow[dd] & \cdots \\
\mathbf{w}_0 \arrow[u] & & \mathbf{w}_1 \arrow[u] & & & \mathbf{w}_{k-1} \arrow[u] & & \mathbf{w}_k \arrow[u] & & \\
\mathbf{n}_0 \arrow[r] & \boxed{\mathbf{g}} \arrow[d] & \mathbf{n}_1 \arrow[r] & \boxed{\mathbf{g}} \arrow[d] & & \mathbf{n}_{k-1} \arrow[r] & \boxed{\mathbf{g}} \arrow[d] & \mathbf{n}_k \arrow[r] & \boxed{\mathbf{g}} \arrow[d] & \\
 & \mathbf{y}_0 & & \mathbf{y}_1 & & & \mathbf{y}_{k-1} & & \mathbf{y}_k &
\end{tikzcd}$$

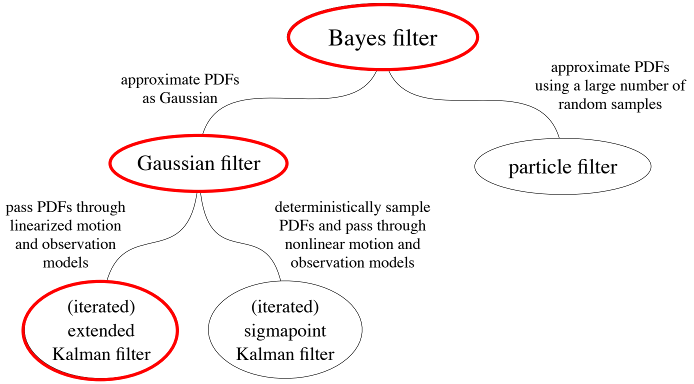
Problem statement

- our state estimation problem is still the following:

Definition

The problem of state estimation is to come up with an estimate, $\hat{\mathbf{x}}_k$, of the true *state* of a system, at one or more timesteps, k, given knowledge of the initial state, $\check{\mathbf{x}}_0$, a sequence of measurements, $\mathbf{y}_{0:K,\text{meas}}$, a sequence of inputs, $\mathbf{v}_{1:K}$, as well as knowledge of the system's motion and observation models.

Filter taxonomy


Bayes filter
approximate PDFs as Gaussian
approximate PDFs using a large number of random samples
Gaussian filter
particle filter
pass PDFs through linearized motion and observation models
deterministically sample PDFs and pass through nonlinear motion and observation models
(iterated) extended Kalman filter
(iterated) sigmapoint Kalman filter

Bayes filter

- recall the very general Bayes filter:

$$\underbrace{p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\text{posterior belief}}$$
$$= \eta \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\substack{\text{observation}\\ \text{correction}\\ \text{using } \mathbf{g}(\cdot)}} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k)}_{\substack{\text{motion}\\ \text{prediction}\\ \text{using } \mathbf{f}(\cdot)}} \underbrace{p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})}_{\text{prior belief}} \, d\mathbf{x}_{k-1} \quad (2)$$

- we can see this takes on a predictor-corrector form, just like the KF

UNIVERSITY OF TORONTO

Generalized Gaussian filter

– assuming all the densities are Gaussian we have the generalized Gaussian filter:

$$\begin{aligned}
\text{predictor:} \quad \check{\mathbf{P}}_k &= \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}'_k && \text{(3a)}\\
\check{\mathbf{x}}_k &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}) && \text{(3b)}\\
\text{Kalman gain:} \quad \mathbf{K}_k &= \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1} && \text{(3c)}\\
\text{corrector:} \quad \hat{\mathbf{P}}_k &= \check{\mathbf{P}}_k - \mathbf{K}_k\boldsymbol{\Sigma}_{xy,k}^T && \text{(3d)}\\
\hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \boldsymbol{\mu}_{y,k}\right) && \text{(3e)}
\end{aligned}$$

– in this case, the moments come from constructing the joint Gaussian (state and measurement)...somehow

$$p(\mathbf{x}_k, \mathbf{y}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{x,k}\\ \boldsymbol{\mu}_{y,k}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{xx,k} & \boldsymbol{\Sigma}_{xy,k}\\ \boldsymbol{\Sigma}_{yx,k} & \boldsymbol{\Sigma}_{yy,k}\end{bmatrix}\right) \quad (4)$$

UNIVERSITY OF TORONTO

IEKF

- using linearization to compute the statistical moment in the joint Gaussian we have

predictor:

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_k' \quad (5a)$$

$$\check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}) \quad (5b)$$

Kalman gain:

$$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{G}_k^T\left(\mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^T + \mathbf{R}_k'\right)^{-1} \quad (5c)$$

corrector:

$$\hat{\mathbf{P}}_k = \left(\mathbf{1} - \mathbf{K}_k\mathbf{G}_k\right)\check{\mathbf{P}}_k \quad (5d)$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{y}_{\text{op},k} - \mathbf{G}_k(\check{\mathbf{x}}_k - \mathbf{x}_{\text{op},k})\right) \quad (5e)$$

- in this case, we iterate the last three equations to convergence
- the ‘mean’ of the IEKF actually converges to the mode of the posterior

UNIVERSITY OF TORONTO

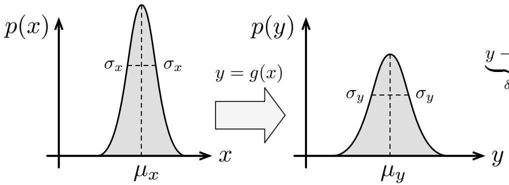
EKF

– finally, with just one iteration we have the EKF

$$\begin{aligned}
&\text{predictor:} & \check{\mathbf{P}}_k &= \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}'_k && \text{(6a)}\\
& & \check{\mathbf{x}}_k &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}) && \text{(6b)}\\
&\text{Kalman gain:} & \mathbf{K}_k &= \check{\mathbf{P}}_k\mathbf{G}_k^T\left(\mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^T + \mathbf{R}'_k\right)^{-1} && \text{(6c)}\\
&\text{corrector:} & \hat{\mathbf{P}}_k &= \left(\mathbf{1} - \mathbf{K}_k\mathbf{G}_k\right)\check{\mathbf{P}}_k && \text{(6d)}\\
& & \hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0})\right) && \text{(6e)}
\end{aligned}$$

UNIVERSITY OF TORONTO

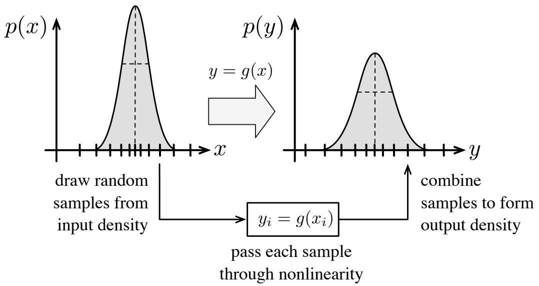
Transforming PDFs through nonlinearities



$$\begin{aligned}
\mu_y &= g(\mu_x) \\
\underbrace{y - \mu_y}_{\delta y} &\approx \underbrace{\left.\frac{\partial g(x)}{\partial x}\right|_{x=\mu_x}}_{a} \underbrace{x - \mu_x}_{\delta x} \\
\sigma_y^2 &= E[\delta y^2] \\
&= a^2 E[\delta x^2] \\
&= a^2 \sigma_x^2
\end{aligned}$$

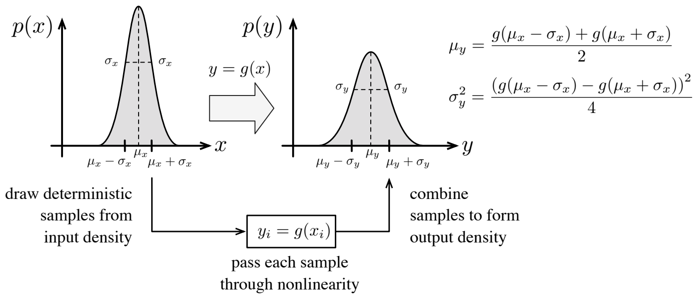
- we've seen how to use linearization to transform a Gaussian through a nonlinearity
- there are some other techniques we can use:
  - Monte Carlo sampling can be used to transform any PDF through a nonlinearity
  - sigmapoint transformation can be used to transform a Gaussian through a nonlinearity

Monte Carlo sampling


$p(x)$
x
$y = g(x)$
$p(y)$
y
draw random samples from input density
$y_i = g(x_i)$
pass each sample through nonlinearity
combine samples to form output density

- Monte Carlo just draws many random samples from the input, transforms each sample, then recombines the samples (e.g., compute moments)
- it's very inefficient, works with any type of PDF, any type of nonlinearity

Sigmapoint transformation



$p(x)$
σ_x
σ_x
x
$\mu_x - \sigma_x$
μ_x
$\mu_x + \sigma_x$
$y = g(x)$
$p(y)$
σ_y
σ_y
y
$\mu_y - \sigma_y$
μ_y
$\mu_y + \sigma_y$
$\mu_y = \frac{g(\mu_x - \sigma_x) + g(\mu_x + \sigma_x)}{2}$
$\sigma_y^2 = \frac{(g(\mu_x - \sigma_x) - g(\mu_x + \sigma_x))^2}{4}$
draw deterministic samples from input density
$y_i = g(x_i)$
pass each sample through nonlinearity
combine samples to form output density

- the sigmapoint transformation draws a small number of deterministic samples, transforms each sample, then recombines using special moment formulas
- it's quite efficient, works with Gaussian PDFs

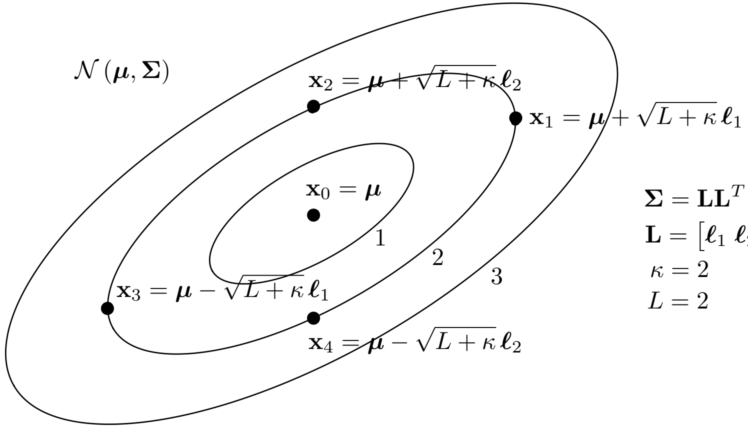
SP: step 1

- a set of $2L+1$ sigmapoints is computed from the input density, $\mathcal{N}\left(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}\right)$, according to

$$\begin{aligned}
\mathbf{L}\mathbf{L}^T &= \boldsymbol{\Sigma}_x \quad \text{(Cholesky decomposition, } \mathbf{L} \text{ lower-triangular)} && \text{(7a)} \\
\mathbf{x}_0 &= \boldsymbol{\mu}_x && \text{(7b)} \\
\mathbf{x}_i &= \boldsymbol{\mu}_x + \sqrt{L+\kappa}\,\text{col}_i\mathbf{L} \qquad i = 1 \ldots L && \text{(7c)} \\
\mathbf{x}_{i+L} &= \boldsymbol{\mu}_x - \sqrt{L+\kappa}\,\text{col}_i\mathbf{L} && \text{(7d)}
\end{aligned}$$

where $L = \text{dim}(\boldsymbol{\mu}_x)$

SP: step 1



$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
$\mathbf{x}_2 = \boldsymbol{\mu} + \sqrt{L+\kappa}\,\boldsymbol{\ell}_2$
$\mathbf{x}_1 = \boldsymbol{\mu} + \sqrt{L+\kappa}\,\boldsymbol{\ell}_1$
$\mathbf{x}_0 = \boldsymbol{\mu}$
1
2
3
$\mathbf{x}_3 = \boldsymbol{\mu} - \sqrt{L+\kappa}\,\boldsymbol{\ell}_1$
$\mathbf{x}_4 = \boldsymbol{\mu} - \sqrt{L+\kappa}\,\boldsymbol{\ell}_2$

$$\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$$
$$\mathbf{L} = \left[\boldsymbol{\ell}_1 \; \boldsymbol{\ell}_2\right]$$
$$\kappa = 2$$
$$L = 2$$

SP: step 1

- the original moments can easily be rebuilt from the sigmapoints:

$$\boldsymbol{\mu}_x = \sum_{i=0}^{2L} \alpha_i \, \mathbf{x}_i \quad (8a)$$

$$\boldsymbol{\Sigma}_{xx} = \sum_{i=0}^{2L} \alpha_i \left(\mathbf{x}_i - \boldsymbol{\mu}_x\right)\left(\mathbf{x}_i - \boldsymbol{\mu}_x\right)^T \quad (8b)$$

  where

$$\alpha_i = \begin{cases} \frac{\kappa}{L+\kappa} & i = 0 \\ \frac{1}{2}\frac{1}{L+\kappa} & \text{otherwise} \end{cases} \quad (9)$$

  which we note sums to 1

- the user-definable parameter, κ, will be explained later

UNIVERSITY OF TORONTO

SP: steps 2-5

- each of the sigmapoints is individually passed through the nonlinearity, $\mathbf{g}(\cdot)$:

$$\mathbf{y}_i = \mathbf{g}\left(\mathbf{x}_i\right), \qquad i = 0 \ldots 2L \tag{10}$$

- the mean of the output density, $\boldsymbol{\mu}_y$, is computed as

$$\boldsymbol{\mu}_y = \sum_{i=0}^{2L} \alpha_i \, \mathbf{y}_i \tag{11}$$

- the covariance of the output density, $\boldsymbol{\Sigma}_y$, is computed as

$$\boldsymbol{\Sigma}_{yy} = \sum_{i=0}^{2L} \alpha_i \left(\mathbf{y}_i - \boldsymbol{\mu}_y\right) \left(\mathbf{y}_i - \boldsymbol{\mu}_y\right)^T \tag{12}$$

- the output density, $\mathcal{N}\left(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy}\right)$, is returned

UNIVERSITY OF TORONTO

SP commentary

- by approximating the input density instead of linearizing, we avoid the need to compute the Jacobian of the nonlinearity (either in closed form or numerically)
- we employ only standard linear algebra operations (Cholesky decomposition, outer products, matrix summations)
- the computation cost is similar to linearization (when a numerical Jacobian is used)
- there is no requirement that the nonlinearity be smooth and differentiable

Comparison example

- we'll use a simple example to compare the various transformation methods:
  - Monte Carlo
  - linearization
  - sigmapoint
- consider the simple one-dimensional nonlinearity

$$f(x) = x^2 \tag{13}$$

- let the prior density be

$$\mathcal{N}(\mu_x, \sigma_x^2) \tag{14}$$

Example: Monte Carlo

- for this example, we can work out the Monte Carlo method analytically
- an arbitrary sample (a.k.a., realization) of the input density is given by

$$x_i = \mu_x + \delta x_i, \qquad \delta x_i \leftarrow \mathcal{N}(0, \sigma_x^2) \tag{15}$$

- transforming this sample through the nonlinearity we get

$$y_i = f(x_i) = f(\mu_x + \delta x_i) = (\mu_x + \delta x_i)^2 = \mu_x^2 + 2\mu_x \delta x_i + \delta x_i^2 \tag{16}$$

- taking the expectation of both sides we arrive at the mean of the output:

$$\mu_y = E\left[y_i\right] = \mu_x^2 + 2\mu_x \underbrace{E\left[\delta x_i\right]}_{0} + \underbrace{E\left[\delta x_i^2\right]}_{\sigma_x^2} = \mu_x^2 + \sigma_x^2 \tag{17}$$

Example: Monte Carlo

– we do a similar thing for the variance of the output:

$$\begin{aligned}
\sigma_y^2 &= E\left[(y_i - \mu_y)^2\right] && (18a)\\
&= E\left[(2\mu_x \delta x_i + \delta x_i^2 - \sigma_x^2)^2\right] && (18b)\\
&= \underbrace{E\left[\delta x_i^4\right]}_{3\sigma_x^4} + 4\mu_x \underbrace{E\left[\delta x_i^3\right]}_{0} + (4\mu_x^2 - 2\sigma_x^2)\underbrace{E\left[\delta x_i^2\right]}_{\sigma_x^2} \\
&\qquad - 4\mu_x \sigma_x^2 \underbrace{E\left[\delta x_i\right]}_{0} + \sigma_x^4 && (18c)\\
&= 4\mu_x^2\sigma_x^2 + 2\sigma_x^4 && (18d)
\end{aligned}$$

where $E\left[\delta x_i^3\right] = 0$ and $E\left[\delta x_i^4\right] = 3\sigma_x^4$ are the well-known third and fourth moments for a Gaussian PDF

Example: linearization

– linearizing about the mean of the input density we have

$$y_i = f(\mu_x + \delta x_i) \approx \underbrace{f(\mu_x)}_{\mu_x^2} + \underbrace{\left.\frac{\partial f}{\partial x}\right|_{\mu_x}}_{2\mu_x} \delta x_i = \mu_x^2 + 2\mu_x \delta x_i \qquad (19)$$

– taking the expectation we arrive at the mean of the output:

$$\mu_y = E[y_i] = \mu_x^2 + 2\mu_x \underbrace{E[\delta x_i]}_{0} = \mu_x^2 \qquad (20)$$

– for the variance of the output we have

$$\sigma_y^2 = E\left[(y_i - \mu_y)^2\right] = E\left[(2\mu_x \delta x_i)^2\right] = 4\mu_x^2 \sigma_x^2 \qquad (21)$$

– this estimate is biased and overconfident compared to Monte Carlo

Example: sigmapoint

- there are $2L+1=3$ sigmapoints in dimension $L=1$:

$$x_0=\mu_x, \quad x_1=\mu_x+\sqrt{1+\kappa}\,\sigma_x, \quad x_2=\mu_x-\sqrt{1+\kappa}\,\sigma_x \tag{22}$$

  where κ is a user-definable parameter that we discuss below
- we pass each sigmapoint through the nonlinearity:

$$\begin{aligned} y_0 &= f\left(x_0\right)=\mu_x^2 && \text{(23a)} \\ y_1 &= f\left(x_1\right)=\left(\mu_x+\sqrt{1+\kappa}\,\sigma_x\right)^2 \\ &\qquad =\mu_x^2+2\mu_x\sqrt{1+\kappa}\,\sigma_x+(1+\kappa)\sigma_x^2 && \text{(23b)} \\ y_2 &= f\left(x_2\right)=\left(\mu_x-\sqrt{1+\kappa}\,\sigma_x\right)^2 \\ &\qquad =\mu_x^2-2\mu_x\sqrt{1+\kappa}\,\sigma_x+(1+\kappa)\sigma_x^2 && \text{(23c)} \end{aligned}$$

UNIVERSITY OF TORONTO

Example: sigmapoint

- the mean of the output is given by

$$\begin{aligned}
\mu_y &= \frac{1}{1+\kappa}\left(\kappa y_0 + \frac{1}{2}\sum_{i=1}^{2} y_i\right) && \text{(24a)}\\
&= \frac{1}{1+\kappa}\Big(\kappa\mu_x^2 + \frac{1}{2}\big(\mu_x^2 + 2\mu_x\sqrt{1+\kappa}\,\sigma_x + (1+\kappa)\sigma_x^2 + \mu_x^2 \\
&\qquad -2\mu_x\sqrt{1+\kappa}\,\sigma_x + (1+\kappa)\sigma_x^2\big)\Big) && \text{(24b)}\\
&= \frac{1}{1+\kappa}\left(\kappa\mu_x^2 + \mu_x^2 + (1+\kappa)\sigma_x^2\right) && \text{(24c)}\\
&= \mu_x^2 + \sigma_x^2 && \text{(24d)}
\end{aligned}$$

which is independent of κ and exactly the same as Monte Carlo

UNIVERSITY OF TORONTO

Example: sigmapoint

- for the variance we have

$$\sigma_y^2 = \frac{1}{1+\kappa}\left(\kappa\left(y_0-\mu_y\right)^2 + \frac{1}{2}\sum_{i=1}^{2}\left(y_i-\mu_y\right)^2\right) \tag{25a}$$

$$= \frac{1}{1+\kappa}\Bigg(\kappa\sigma_x^4 + \frac{1}{2}\Big(\left(2\mu_x\sqrt{1+\kappa}\,\sigma_x+\kappa\sigma_x^2\right)^2 + \left(-2\mu_x\sqrt{1+\kappa}\,\sigma_x+\kappa\sigma_x^2\right)^2\Big)\Bigg) \tag{25b}$$

$$= \frac{1}{1+\kappa}\left(\kappa\sigma_x^4 + 4(1+\kappa)\mu_x^2\sigma_x^2 + \kappa^2\sigma_x^4\right) \tag{25c}$$

$$= 4\mu_x^2\sigma_x^2 + \kappa\sigma_x^4 \tag{25d}$$

which can be made to be identical to Monte Carlo by selecting the user-definable parameter, κ, to be 2

- thus, for this nonlinearity, the unscented transformation can exactly capture the correct mean and variance of the output

UNIVERSITY OF TORONTO

Example: sigmapoint

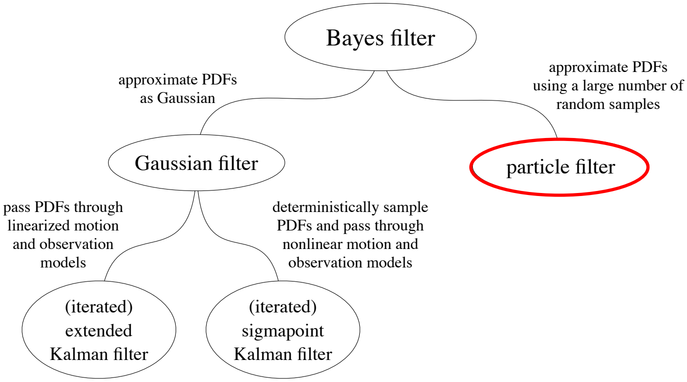
- we choose κ to make the fourth moment of the sigmapoints match the true kurtosis of the Gaussian input density:

$$
\begin{aligned}
3\sigma_x^4 &= \frac{1}{1+\kappa}\left(\kappa\underbrace{(x_0-\mu_x)^4}_{0}+\frac{1}{2}\sum_{i=1}^{2}(x_i-\mu_x)^4\right) && \text{(26a)}\\
&= \frac{1}{2(1+\kappa)}\left(\left(\sqrt{1+\kappa}\sigma_x\right)^4+\left(-\sqrt{1+\kappa}\sigma_x\right)^4\right) && \text{(26b)}\\
&= (1+\kappa)\sigma_x^4 && \text{(26c)}
\end{aligned}
$$

- comparing the desired and actual kurtosis, we should pick $\kappa = 2$ to make them match exactly
- not surprisingly, this has a positive effect on accuracy of the transformation

UNIVERSITY OF TORONTO

Particle filter


Bayes filter
approximate PDFs as Gaussian
approximate PDFs using a large number of random samples
Gaussian filter
particle filter
pass PDFs through linearized motion and observation models
deterministically sample PDFs and pass through nonlinear motion and observation models
(iterated) extended Kalman filter
(iterated) sigmapoint Kalman filter

UNIVERSITY OF TORONTO

Particle filter

- let's try to use the Monte Carlo transformation method to implement the Bayes filter
- the particle filter is one of the only practical techniques able to handle non-Gaussian noise and nonlinear observation and motion models
- it is practical in that it is very easy to implement; we do not even need to have analytical expressions for $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$, nor their derivatives
- can be inefficient due to the need for many samples
- sometimes called the bootstrap algorithm, the condensation algorithm, or the survival-of-the-fittest algorithm

PF: prediction

- draw M samples from the joint density comprising the prior and the motion noise:

$$\begin{bmatrix} \hat{\mathbf{x}}_{k-1,m} \\ \mathbf{w}_{k,m} \end{bmatrix} \leftarrow p\left(\mathbf{x}_{k-1} \middle| \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{1:k-1}\right) p(\mathbf{w}_k) \tag{27}$$

  where m is the unique particle index
- generate a prediction of the posterior PDF by using $\mathbf{v}_k$
- this is done by passing each prior particle/noise sample through the nonlinear motion model:

$$\check{\mathbf{x}}_{k,m} = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1,m}, \mathbf{v}_k, \mathbf{w}_{k,m}\right) \tag{28}$$

- these new predicted particles together approximate the density, $p\left(\mathbf{x}_k \middle| \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k-1}\right)$

PF: correction

- correct the posterior PDF by incorporating $\mathbf{y}_k$
- first, assign a scalar weight, $w_{k,m}$, to each predicted particle based on the divergence between the desired posterior and the predicted posterior for each particle:

$$w_{k,m} = \frac{p\left(\check{\mathbf{x}}_{k,m} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k}\right)}{p\left(\check{\mathbf{x}}_{k,m} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k-1}\right)} = \eta\, p\left(\mathbf{y}_k | \check{\mathbf{x}}_{k,m}\right) \tag{29}$$

  where η is a normalization constant
- this is typically accomplished in practice by simulating an expected sensor reading, $\check{\mathbf{y}}_{k,m}$, using the nonlinear observation model:

$$\check{\mathbf{y}}_{k,m} = \mathbf{g}\left(\check{\mathbf{x}}_{k,m}, \mathbf{0}\right) \tag{30}$$

- we then assume $p\left(\mathbf{y}_k | \check{\mathbf{x}}_{k,m}\right) = p\left(\mathbf{y}_k | \check{\mathbf{y}}_{k,m}\right)$, where the right-hand side is a known density (e.g., Gaussian)

UNIVERSITY OF TORONTO

PF: correction

- second, resample the posterior based on the weight assigned to each predicted posterior particle:

$$\hat{\mathbf{x}}_{k,m} \overset{\text{resample}}{\longleftarrow} \{\check{\mathbf{x}}_{k,m}, w_{k,m}\} \tag{31}$$

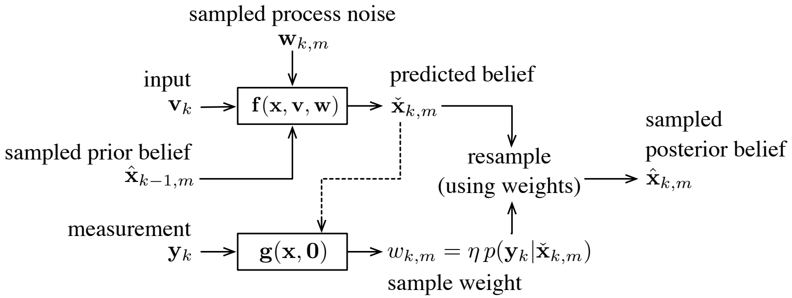
- this can be done in several different ways (e.g., Madow systematic sampling)
- create M bins based on weights

$$\beta_m = \frac{\sum_{n=1}^{m} w_n}{\sum_{\ell=1}^{M} w_\ell} \tag{32}$$

$$0 \le \beta_1 \le \beta_2 \le \ldots \le \beta_{M-1} \le 1$$

- march along in steps of $1/M$, adding a sample from each bin that is visited

PF in pictures



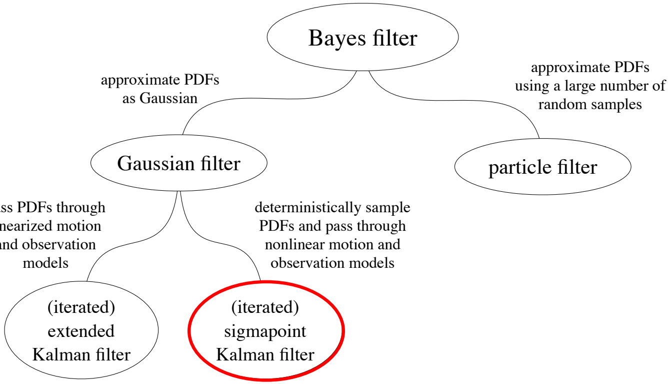
PF in action

PF commentary

- we do not necessarily need to resample every time we go through the algorithm
- to be on the safe side, it is wise to add a small percentage of samples in the prediction step that are uniformly drawn from the entire state sample space
- for high-dimensional state estimation problems, the PF can become computationally intractable
- the PF is an anytime algorithm – we can just keep adding particles until we run out of time, then resample and give an answer
- we can dynamically pick the number of particles online using a heuristic such as $\sum w_{k,m} \geq w_{\text{thresh}}$, a threshold
- the CRLB is set by the uncertainty in the measurements that we have available – using more samples does not allow us to do better

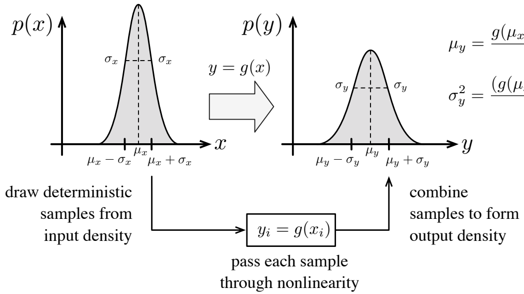
UNIVERSITY OF TORONTO

Sigmapoint Kalman filter


Bayes filter
approximate PDFs as Gaussian
approximate PDFs using a large number of random samples
Gaussian filter
particle filter
pass PDFs through linearized motion and observation models
deterministically sample PDFs and pass through nonlinear motion and observation models
(iterated) extended Kalman filter
(iterated) sigmapoint Kalman filter

SPKF

– we can also derive a filtering scheme that uses the sigmapoint transformation to implement the generalized Gaussian filter, rather than linearization



$$\mu_y = \frac{g(\mu_x - \sigma_x) + g(\mu_x + \sigma_x)}{2}$$

$$\sigma_y^2 = \frac{(g(\mu_x - \sigma_x) - g(\mu_x + \sigma_x))^2}{4}$$

SPKF: prediction

- both the prior belief and the motion noise have uncertainty so these are stacked together in the following way:

$$\boldsymbol{\mu}_z = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} \\ \mathbf{0} \end{bmatrix}, \qquad \boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \hat{\mathbf{P}}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k \end{bmatrix} \tag{33}$$

  where we see that $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation
- we let $L = \dim \boldsymbol{\mu}_z$
- convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation:

$$\begin{aligned}
\mathbf{L}\mathbf{L}^T &= \boldsymbol{\Sigma}_{zz} \quad \text{(Cholesky decomposition, } \mathbf{L} \text{ lower-triangular)} && \text{(34a)} \\
\mathbf{z}_0 &= \boldsymbol{\mu}_z && \text{(34b)} \\
\mathbf{z}_i &= \boldsymbol{\mu}_z + \sqrt{L+\kappa}\,\mathrm{col}_i \mathbf{L} \qquad i = 1 \ldots L && \text{(34c)} \\
\mathbf{z}_{i+L} &= \boldsymbol{\mu}_z - \sqrt{L+\kappa}\,\mathrm{col}_i \mathbf{L} && \text{(34d)}
\end{aligned}$$

UNIVERSITY OF TORONTO

SPKF: prediction

- unstack each sigmapoint into state and motion noise,

$$\mathbf{z}_i = \begin{bmatrix} \hat{\mathbf{x}}_{k-1,i} \\ \mathbf{w}_{k,i} \end{bmatrix} \tag{35}$$

- pass each sigmapoint through the nonlinear motion model exactly:

$$\check{\mathbf{x}}_{k,i} = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1,i}, \mathbf{v}_k, \mathbf{w}_{k,i}\right) \qquad i = 0 \ldots 2L \tag{36}$$

- the latest input, $\mathbf{v}_k$, is required

UNIVERSITY OF TORONTO

SPKF: prediction

– recombine the transformed sigmapoints into the predicted belief, $\left\{\check{\mathbf{x}}_k, \check{\mathbf{P}}_k\right\}$, according to

$$\check{\mathbf{x}}_k = \sum_{i=0}^{2L} \alpha_i \, \check{\mathbf{x}}_{k,i} \qquad (37a)$$

$$\check{\mathbf{P}}_k = \sum_{i=0}^{2L} \alpha_i \left(\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k\right)\left(\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k\right)^T \qquad (37b)$$

where

$$\alpha_i = \begin{cases} \frac{\kappa}{L+\kappa} & i = 0 \\ \frac{1}{2}\frac{1}{L+\kappa} & \text{otherwise} \end{cases} \qquad (38)$$

UNIVERSITY OF TORONTO

SPKF: correction

- this step is a little more subtle
- we'll begin by recalling the generalized Gaussian correction step:

$$
\begin{aligned}
\mathbf{K}_k &= \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1} && \text{(39a)}\\
\hat{\mathbf{P}}_k &= \check{\mathbf{P}}_k - \mathbf{K}_k\boldsymbol{\Sigma}_{xy,k}^T && \text{(39b)}\\
\hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \boldsymbol{\mu}_{y,k}\right) && \text{(39c)}
\end{aligned}
$$

- we'll use the sigmapoint transformation to come up with better versions of $\boldsymbol{\mu}_{y,k}$, $\boldsymbol{\Sigma}_{yy,k}$, and $\boldsymbol{\Sigma}_{xy,k}$ than we found through linearization

SPKF: correction

- both the predicted belief and the observation noise have uncertainty so these are stacked together in the following way:

$$\boldsymbol{\mu}_z = \begin{bmatrix} \check{\mathbf{x}}_k \\ \mathbf{0} \end{bmatrix}, \qquad \boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \check{\mathbf{P}}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{bmatrix} \tag{40}$$

  where we see that $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation
- we let $L = \dim \boldsymbol{\mu}_z$
- convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation:

$$\begin{aligned}
\mathbf{L}\mathbf{L}^T &= \boldsymbol{\Sigma}_{zz} \quad \text{(Cholesky decomposition, } \mathbf{L} \text{ lower-triangular)} && \text{(41a)} \\
\mathbf{z}_0 &= \boldsymbol{\mu}_z && \text{(41b)} \\
\mathbf{z}_i &= \boldsymbol{\mu}_z + \sqrt{L+\kappa}\,\text{col}_i \mathbf{L} && \text{(41c)} \\
\mathbf{z}_{i+L} &= \boldsymbol{\mu}_z - \sqrt{L+\kappa}\,\text{col}_i \mathbf{L} \qquad i = 1 \ldots L && \text{(41d)}
\end{aligned}$$

SPKF: correction

- unstack each sigmapoint into state and observation noise,

$$\mathbf{z}_i = \begin{bmatrix} \check{\mathbf{x}}_{k,i} \\ \mathbf{n}_{k,i} \end{bmatrix} \tag{42}$$

- pass each sigmapoint through the nonlinear observation model exactly:

$$\check{\mathbf{y}}_{k,i} = \mathbf{g}\left(\check{\mathbf{x}}_{k,i}, \mathbf{n}_{k,i}\right) \tag{43}$$

UNIVERSITY OF TORONTO

SPKF: correction

- recombine the transformed sigmapoints into the desired moments:

$$\boldsymbol{\mu}_{y,k} = \sum_{i=0}^{2L} \alpha_i \, \check{\mathbf{y}}_{k,i} \tag{44a}$$

$$\boldsymbol{\Sigma}_{yy,k} = \sum_{i=0}^{2L} \alpha_i \left(\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k}\right) \left(\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k}\right)^T \tag{44b}$$

$$\boldsymbol{\Sigma}_{xy,k} = \sum_{i=0}^{2L} \alpha_i \left(\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_{k}\right) \left(\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k}\right)^T \tag{44c}$$

where

$$\alpha_i = \begin{cases} \frac{\kappa}{L+\kappa} & i = 0 \\ \frac{1}{2}\frac{1}{L+\kappa} & \text{otherwise} \end{cases} \tag{45}$$

- these are plugged into the generalized Gaussian correction-step equations above to complete the correction step

UNIVERSITY OF TORONTO

SPKF commentary

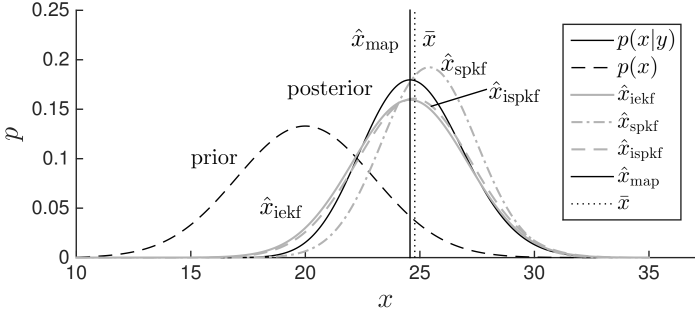
- a big advantage of the SPKF (over the EKF) is that it does not require any analytical derivatives of the motion and observation models
- in fact, the models could even just be black-box software functions - no need to express them mathematically!
- there are some additional efficiencies that can be found in the calculations when the observation model becomes:

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) + \mathbf{n}_k, \tag{46}$$

- there is even an iterated SPKF that can be formulated using the idea of statistical Jacobians

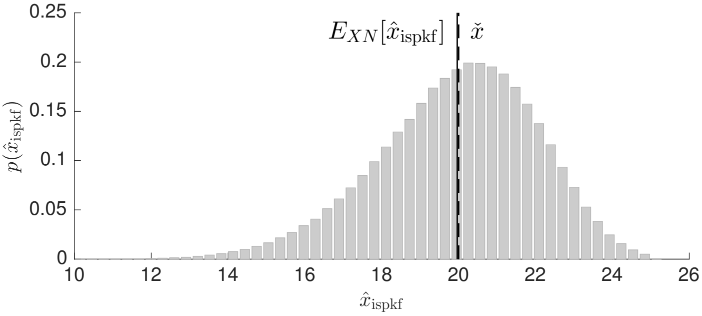
UNIVERSITY OF TORONTO

SPKF example



– we see that the SPKF does converge to a slightly different place than the IEKF on our simple stereo camera example
– it does not go to the MAP solution...

SPKF example



– over 1,000,000 we find the ISPKF actually converges quite close to the mean of the true posterior, not the mode

Summary

- we investigated a few more alternatives in our taxonomy of filters
- by using Monte Carlo sampling, we came up with the particle filter, which can be used for general nonlinear, non-Gaussian problems, but is quite inefficient in most cases
- by using the sigmapoint transformation, we came up with the sigmapoint Kalman filter, which is only slightly less efficient than the EKF, but is quite a bit more accurate on nonlinear problems
- the choice of EKF, PF, or SPKF depends on how nonlinear and non-Gaussian your problem is

UNIVERSITY OF TORONTO