Aerial Robotic Autonomy: Methods and Systems

Nonlinear Non-Gaussian Estimation

Kostas Alexis

Autonomous Robots Lab
Norwegian University of Science and Technology

Table of Contents

Table of Contents

Introduction

- In the real-world case, the processes will not be linear-Gaussian systems.
- We shall discuss some of the most common approaches to deal with nonlinear and/or non-Gaussian systems.
- We begin by contrasting full Bayesian estimation to Maximum A Posteriori (MAP) estimation for nonlinear systems.
- We shall introduce the Bayes filter.
- We discussed common methods such as the Extended Kalman Filter, the sigmapoint Kalman Filter and particle filter as approximations of the Bayes filter.
- We further discuss batch optimization for nonlinear systems, both in the discrete and continuous time case.

Main reference: Barfoot, T.D., 2017. State estimation for robotics. Cambridge University Press.

Introduction

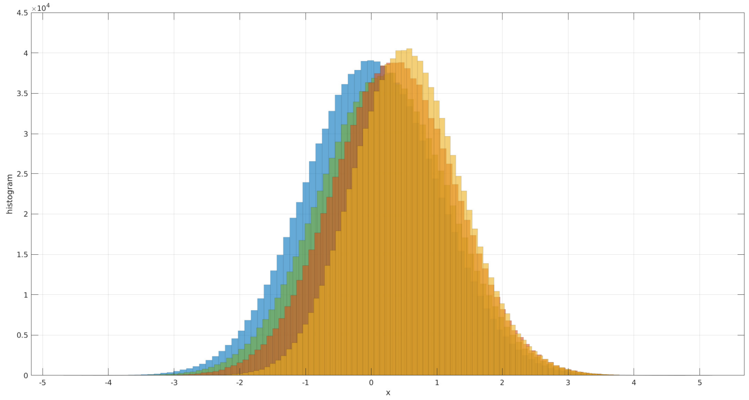


Figure: Skewed gaussian results.

Introduction

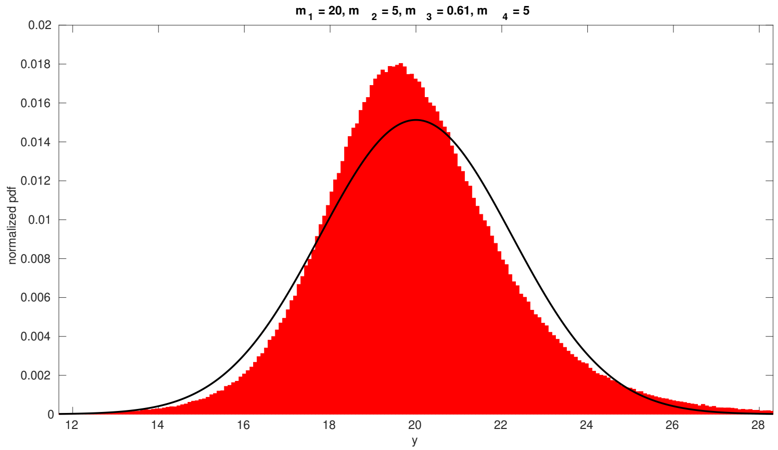


Figure: Skewed gaussian results.

Introduction

- For the case of linear-Gaussian estimation, and specifically for the case of linear motion and observation models driven by Gaussian noise, the full Bayesian and MAP estimation paradigms provide the same answer.
- The above is not the case when we move to nonlinear models.
- For the case of nonlinear models, the full Bayesian posterior is no longer Gaussian.

To develop intuition with respect to the above, we look at a simple one-dimensional nonlinear estimation problem: estimating the position of a landmark from an ideal stereo camera.

Introduction

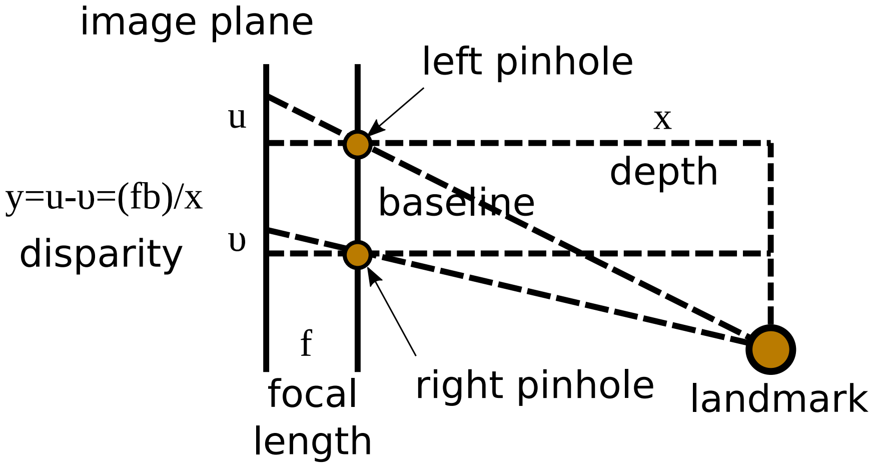


Figure: Idealized stereo camera model relating the landmark depth, x, to the (noise-free) disparity measurement, y.

Full Bayesian Estimation

Considering a simplified pinhole model for a stereo camera pair, it then holds

$$y = \frac{fb}{x} + n \tag{1}$$

where

- x: state, the position of a landmark (in metres)
- y: measurement, the disparity between the horizontal coordinates of the landmark in the left and right images (in pixels)
- f: the focal length (in pixels)
- b: the baseline, i.e., the horizontal distance between left and right cameras (in metres)
- n: the measurement noise (in pixels)

where the state

Full Bayesian Estimation

To perform Bayesian inference

$$p(x|y) = \frac{p(y|x)p(x)}{\int_{\infty}^{\infty} p(y|x)p(x)dx} \tag{2}$$

we require expressions for both $p(y|x)$ and $p(x)$. To meet this requirement, we make two assumptions.

Assumption 1: The measurement noise is zero-mean Gaussian, $n \sim \mathcal{N}(0, R)$ such that

$$p(y|x) = \mathcal{N}\left(\frac{fb}{x}, R\right) = \frac{1}{\sqrt{2\pi R}}\left(-\frac{1}{2R}\left(y - \frac{fb}{x}\right)^2\right) \tag{3}$$

Assumption 2: The prior is Gaussian, where

$$p(x) = \mathcal{N}(\check{x}, \check{P}) = \frac{1}{\sqrt{2\pi\check{P}}} \exp\left(-\frac{1}{2\check{P}}(x - \check{x})^2\right) \tag{4}$$

Full Bayesian Estimation

We continue by first noting that Bayesian estimation implies a certain order of operations:

$$\texttt{assign prior} \rightarrow \texttt{draw } x_{true} \rightarrow \texttt{draw } y_{meas} \rightarrow \texttt{compute posterior}$$

We start with the prior. The 'true' state is then drawn from the prior, and the measurement is generated by observing the true state through the camera model and adding noise. The estimator then reconstructs the posterior from the measurement and prior, without knowing x_{true}. This process is necessary to ensure a 'fair' comparison between state estimation algorithms.

Full Bayesian Estimation

For the particular example, let the following specific parameters hold

$$\check{x} = 20[\mathrm{m}],\ \check{P} = 9[\mathrm{m}^2],\ f = 400[\mathrm{pixel}],\ b = 0.1[\mathrm{m}],\ R = 0.09[\mathrm{pixel}^2] \tag{5}$$

The true state, x_{true}, and the noise-corrupted measurement, y_{meas}, are drawn randomly from $p(x)$ and $p(y|x)$ respectively. Each time we repeat the experiment, these values will change.

Full Bayesian Estimation

Considering $x_{true} = 22[\mathrm{m}]$ and $y_{meas} = (fb/x_{true}) + 1[\mathrm{pixel}]$ then the following prior and posterior distributions are calucated by numerically deriving the denominator in the Bayes' rule

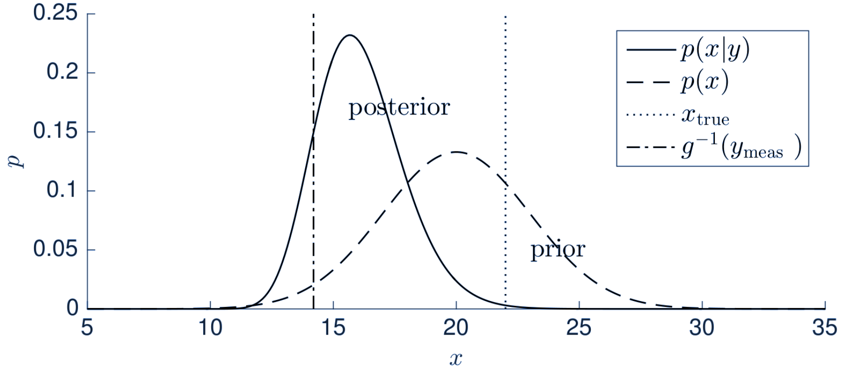


Figure: Example of Bayesian inference on one-dimensional stereo camera example. The posterior is not Gaussian due to the nonlinear measurement model.

Full Bayesian Estimation

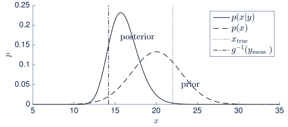


Figure: Example of Bayesian inference on one-dimensional stereo camera example. The posterior is not Gaussian due to the nonlinear measurement model.

The posterior is asymmetrical, skewed to one side by the nonlinear observation model. However, it is still unimodal which could imply that we would be justified by approximating it as Gaussian.
We also see that the incorporation of the measurement results in a posterior that has less variance about the state than the prior: this is the main idea of Bayesian state estimation - *we want to incorporate measurements into the prior to become more certain about the posterior case.*

Maximum A Posteriori Estimation

Computing the full Bayesian posterior can be intractable in general. A very common approach is to seek out only the value of the state that maximizes the true posterior. This is the goal of the Maximum A Posteriori (MAP) estimation.

In other words we want to compute

$$\hat{x}_{map} = \arg\max_{x} p(x|y) \tag{6}$$

Equivalently, we can try minimizing the negative log likelihood

$$\hat{x}_{map} = \arg\min_{x}(-\ln(p(x|y))) \tag{7}$$

which can be easier when the PDFs involved are from the exponential family. As we are seeking only the most likely state, we can use Bayes' rule to write

$$\hat{x}_{map} = \arg\min_{x}(-\ln(p(y|x)) - \ln(p(x))) \tag{8}$$

which drops $p(y)$ since it does not depend on x.

Maximum A Posteriori Estimation

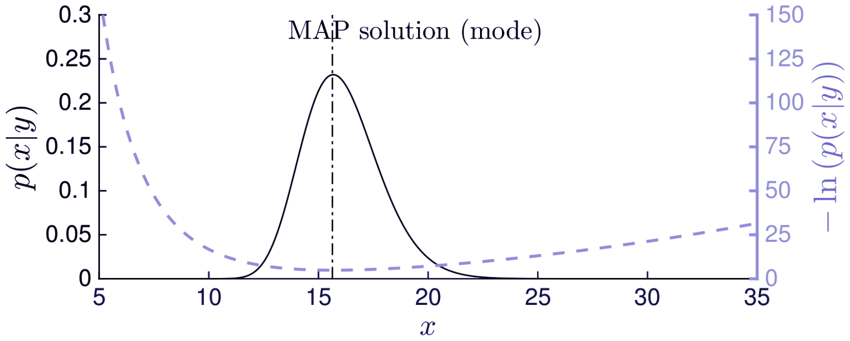


Figure: Posterior from the stereo camera example, $p(x|y)$, as well as the negative log likelihood of the posterior - $\ln(p(x|y)$. The MAP solution is simply the value of x that maximizes (or minimizes) either of these functions. The MAP solution is the *mode* of the posterior, which is not generally the same as the *mean*.

Maximum A Posteriori Estimation

Relating this back to the stereo camera example, we can write

$$\hat{x}_{map} = \arg\max_{x} J(x) \tag{9}$$

$$J(x) = \frac{1}{2R}\left(y - \frac{fb}{x}\right)^2 + \frac{1}{2\check{P}}(\check{x} - x)^2 \tag{10}$$

where we have dropped any further normalization constants not depending on x. The result, $\hat{x}_{map}$, can be found through various optimization techniques.

Since the MAP estimator, $\hat{x}_{map}$, fnds the most likely state given the data prior, the question is *how well does this estimator capture* x_{true}.

Maximum A Posteriori Estimation

To evaluate how well our estimators are performing, we try to compare our estimates $\hat{x}$ with respect to some 'ground truth' whenever this is available. We thus compute

$$e_{mean}(\hat{x}) = E_{XN}[\hat{x} - x_{true}] \tag{11}$$

where $E_{XN}[\cdot]$ represents the expectation operator. Importantly, we are averaging *both over* the random draw of x_{true} from the prior *and* the random draw of n from the measurement noise.

Since x_{true} is assumed to be independent of n, we have that $E_{XN}[x_{true}] = E_X[x_{true}] = \check{x}$ and thus

$$e_{mean}(\hat{x}) = E_{XN}[\hat{x}] - \check{x} \tag{12}$$

Importantly, under this performance measure, the MAP estimator is **biased** (i.e, $e_{mean}(\hat{x}_{map}) \neq 0$) due to the presence of a nonlinear measurement model, $g(\cdot)$ and the fact that the *mode and the mean of the posterior PDF are not the same!* As discussed, in the linear case for $g(\cdot)$, $e_{mean}(\hat{x}_{map}) = 0$.

Maximum A Posteriori Estimation

However, since we can trivially set the estimate to the prior, $\hat{x} = \check{x}$, and obtain $e_{mean}(\check{x}) = 0$, we need to define a secondary performance metric. This is typically the **average squared error**, e_{sq}

$$e_{sq} = E_{XN}[(\hat{x} - x_{true})^2] \tag{13}$$

In other words

- e_{mean} captures the mean of the estimator error
- e_{sq} captures the combined effects of bias and variance

High performance across both metrics results in the bias-variance tradeoff in the machine learning literature and good performance on both is needed for a practical state estimator.

Maximum A Posteriori Estimation

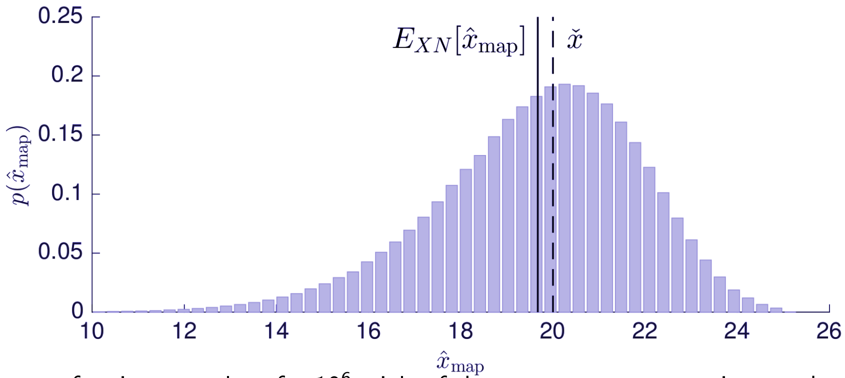


Figure: Histogram of estimator values for 10^6 trials of the stereo camera experiment where each time a new x_{true} is randomly drawn from the prior and a new y_{meas} is randomly drawn from the measurement model. The dash line marks the mean of the prior, $\check{x}$, and the solid line marks the expected value of the MAP estimator, $\hat{x}_{map}$, over all the trials. The gap between dashed and solid is $e_{mean} = -33.0[\text{cm}]$, which indicates a bias. The average squared error is $e_{sq} = 4.41[\text{m}^2]$.

Gaussian Process Regression

We shall take a *Gaussian process regression* approach to state estimation. This allows

- Represent trajectories in continuous time, thus enabling to query the solution at any time of interest.
- For the nonlinear case, to optimize our solution by iterating over the entire trajectory (difficult to do in the recursive formulation which typically iterates at just one timestep at a time).

Under certain class of prior motion models, GP regression enjoys a sparse structure that allows for efficient solutions.

Table of Contents

Recursive Discrete-Time Estimation

- Analogously to the linear case, we shall work first on discrete-time estimation.
- We shall work on nonlinear state estimation methods and introduce the Extended Kalman Filter, the Generalized Gaussian Filter, the Iterated Extended Kalman Filter, the Particle Filter, the (Iterated) Sigmapoint Kalman Filter .
- We shall then discuss relevance to Batch Discrete-Time Estimation and not Continuous-Time Estimation.

Main reference: Barfoot, T.D., 2017. State estimation for robotics. Cambridge University Press.

Problem Setup

- We require a set of motion and observation models upon which we base our estimation.
- As in the linear case, we shall consider discrete-time, time-invariant equations but also allow nonlinear expressions.

Let us define the following motion and observation models:

$$\text{motion model:}\quad \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k), \quad k = 1 \ldots K \tag{14}$$

$$\text{observation model:}\quad \mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k), \quad k = 0 \ldots K \tag{15}$$

where k is the discrete-time index and K its maximum, $\mathbf{f}$ the nonlinear motion model, and $\mathbf{g}$ the nonlinear observation model. For now, we do not make explicit assumptions that any of the random variables is Gaussian.

Problem Setup

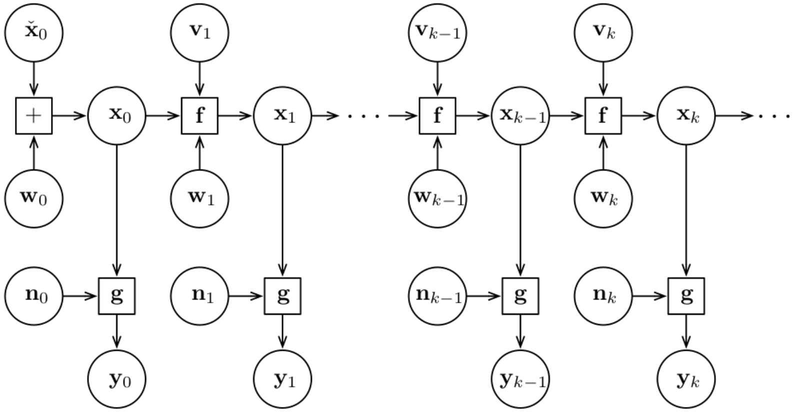


Figure: Graphical model representation of the Markov processes constituting the Non-Linear, Non-Gaussian (NLNG) system in Eqs. (14,15).

Problem Setup

Markov Process

In the simplest sense, a stochastic process has the Markov property if the conditional probability density functions (PDFs) of future states of the process, given the present state, depend only upon the present state, but not only other past states, i.e., they are conditionally independent of these older states. Such a process is called Markovian or a Markov process.

The system in Eqs. (14,15) is a Markov Process.

- Once we know the value of $\mathbf{x}_{k-1}$ we do not need to know the value of any previous states to evolve the system forward in time to compute $\mathbf{x}_k$.

Question: Can we have a recursive solution to NLNG systems as we had in the linear-Gaussian case? *Yes, but only* **approximately**.

Bayes Filter

The Bayes filter seeks to derive an entire PDF to represent the likelihood of the state, $\mathbf{x}_k$, using *only* measurements up to and including the current time. I.e., we want to compute

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) \tag{16}$$

which is also called the **belief** for $\mathbf{x}_k$. Recall that

$$p(\mathbf{x}_k | \mathbf{v}, \mathbf{y}) = \eta \, \underbrace{p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\text{forwards}} \underbrace{p(\mathbf{x}_k | \mathbf{v}_{k+1:K}, \mathbf{y}_{k+1:K})}_{\text{backwards}} \tag{17}$$

Accordingly, we will focus on turning the ‘forwards’ PDF into a recursive filter for nonlinear non-Gaussian systems (NLNG).

Bayes Filter

Assuming that all measurements are statistically independent, we can factor out the latest measurement

$$p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \eta p(\mathbf{y}_k|\mathbf{x}_k) p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) \tag{18}$$

where Bayes' rule was used to reverse the dependence and η is a normalization constant.

Focusing on the second factor, we introduce the **hidden state**, $\mathbf{x}_{k-1}$, and integrate over all possible variables

$$p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \int p(\mathbf{x}_k, \mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1} = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1} \tag{19}$$

where the introduction of the hidden state can be viewed as the opposite of marginalization.

So far, no approximations have been introduced!

Bayes Filter

Since our system enjoys the Markov property (on the estimator), we use it to express

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) \tag{20}$$

$$p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) \tag{21}$$

Substituting Eq. (20,21,19) into Eqs. (18), we have the **Bayes Filter**

$$\underbrace{p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\text{posterior belief}} = \eta \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\substack{\text{observation}\\ \text{correction}\\ \text{using } \mathbf{g}(\cdot)}} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k)}_{\substack{\text{motion}\\ \text{prediction}\\ \text{using } \mathbf{f}(\cdot)}} \underbrace{p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})}_{\text{prior belief}} d\mathbf{x}_{k-1} \tag{22}$$

Bayes Filter

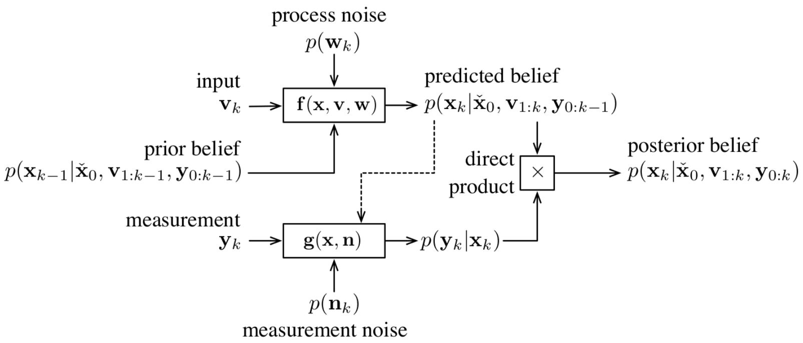

process noise
$p(\mathbf{w}_k)$
input
$\mathbf{v}_k$
$\mathbf{f}(\mathbf{x},\mathbf{v},\mathbf{w})$
predicted belief
$p(\mathbf{x}_k|\check{\mathbf{x}}_0,\mathbf{v}_{1:k},\mathbf{y}_{0:k-1})$
prior belief
$p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0,\mathbf{v}_{1:k-1},\mathbf{y}_{0:k-1})$
direct product
$\times$
posterior belief
$p(\mathbf{x}_k|\check{\mathbf{x}}_0,\mathbf{v}_{1:k},\mathbf{y}_{0:k})$
measurement
$\mathbf{y}_k$
$\mathbf{g}(\mathbf{x},\mathbf{n})$
$p(\mathbf{y}_k|\mathbf{x}_k)$
$p(\mathbf{n}_k)$
measurement noise

Figure: Graphical depictions of the Bayes filter. The dashed line indicates that in practice a hint could be passed to the 'correction step' about the states in which the probability mass of the belif function is concentrated. This can be used to reduce the need to work out the full density for the observation, $\mathbf{y}_k$, given the state, $\mathbf{x}_k$.

Bayes Filter

Eq. (22) takes a **predictor-corrector** form.

- **prediction step:** the prior belief, $p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})$, is propagated forward in time using the input, $\mathbf{v}_k$, and the motion model, $\mathbf{f}(\cdot)$.
- **correction step:** the predicted estimate is then updated using the measurement $\mathbf{y}_k$, and the measurement model, $\mathbf{g}(\cdot)$.
- **result:** posterior belief, $p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})$

Importantly, we require methods of passing PDFs through the nonlinear functions $\mathbf{f}(\cdot), \mathbf{g}(\cdot)$.

Bayes Filter

The Bayes filter is not implementable for all cases but the linear-Gaussian one

The Bayes filter is nothing more that a mathematical artifact, it cannot be implemented in practice with the *exception of the linear-Gaussian* case.

Bayes Filter

There are two reasons for the above, and necessary approximations are to be introduced

1. Probability density functions live in an infinite-dimensional space (as do all continuous functions) and as such an infinite amount of memory (i..e, infinite number of parameters) would be needed to completely represent the belief, $p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})$. To overcome this memory issue, the belief is *approximately* represented. One approach is to **approximate this function as a Gaussian PDF** (i.e., keep track of the first two moments, mean and covariance). Another approach is to **approximate the PDF using a finite number of random samples**.
2. The integral in the Bayes filter is computationally expensive; it would require infinite computing resources to evaluate exactly. To overcome this computational resource issue, the *integral must be evaluated approximately*. One approach is to **linearize the motion and observation models and then evaluate the integrals in closed form**. Another approach is to employ **Monte Carlo integration**.

Bayes Filter

- Much of the research in recursive state estimation has focused on better approximations to handle the above two issues.
- We shall look into some of the classic and modern approaches to approximate the Bayes filter.
- We must keep in our mind the assumption on which the Bayes filter relies: the Markov property. Does this hold when we start making these approximations to the Bayes filter? We will look that in further detail.

Extended Kalman Filter

- The Extended Kalman Filter (EKF) is the most commonly used estimation and data fusion tool in many domains.
- The EKF can often be effective for mildly nonlinear, non-Gaussian systems.
- EKF was a key tool to estimate spacecraft trajectories on the NASA Apollo program.

The Bayes Filter and the EKF

We shall show that if the belief is constrainted to be Gaussian, the noise is Gaussian, and we linearize the motion and observation models in order to carry out the integral (and also the normalized produced) in the Bayes Filter, we arrive at the Extended Kalman Filter.

Extended Kalman Filter

To derive the EKF, we first **constrain our belief function** for $\mathbf{x}_k$ to **be Gaussian**

$$p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \mathcal{N}\left(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\right) \tag{23}$$

where $\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k$ are the mean and covariance, respectively.
We further assume that the **noise variables**, $\mathbf{w}_k$ and $\mathbf{n}_k$ $(\forall k)$ are also **Gaussian**

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \tag{24}$$
$$\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \tag{25}$$

A Gaussian PDF can be transformed to non-Gaussian through a nonlinearity. We assume this is the case for the noise variables, i.e., the nonlinear motion and observation models may affect $\mathbf{w}_k, \mathbf{n}_k$.

Extended Kalman Filter

This means that these noise parameters as in Eqs. (14,15) are not necessarily additive after the nonlinearities. Additive form would look like

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k) + \mathbf{w}_k \tag{26}$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) + \mathbf{n}_k \tag{27}$$

In fact, Eqs (26,27) are a special case of Eqs. (14,15) and we shall show that we can recover approximate additive noise representations through linearization.

Extended Kalman Filter

As $\mathbf{g}(\cdot)$ and $\mathbf{f}(\cdot)$ are nonlinear, we cannot compute the integral of the Bayes filter in closed form. To overcome this, we **linearize the motion and observation models** *about* the current state estimate mean

$$\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k) \approx \check{\mathbf{x}}_k + \mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{w}'_k \tag{28}$$

$$g(\mathbf{x}_k, \mathbf{n}_k) \approx \check{\mathbf{y}}_k + \mathbf{G}_k(\mathbf{x}_k - \check{\mathbf{x}}_k) + \mathbf{n}'_k \tag{29}$$

where

$$\check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}),\ \mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}},\ \mathbf{w}'_k = \left.\frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{w}_k}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}} \mathbf{w}_k \tag{30}$$

and

$$\check{\mathbf{y}} = \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}),\ \mathbf{G}_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k}\right|_{\check{\mathbf{x}}_k, \mathbf{0}},\ \mathbf{n}'_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{n}_k}\right|_{\check{\mathbf{x}}_k, \mathbf{0}} \mathbf{n}_k \tag{31}$$

Extended Kalman Filter

Accordingly, the **statistical properties of the current state**, $\mathbf{x}_k$, given the old state and latest input, take the form

$$\mathbf{x}_k \approx \check{\mathbf{x}}_k + \mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{w}'_k \tag{32}$$

$$E[\mathbf{x}_k] \approx \check{\mathbf{x}}_k + \mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \underbrace{E[\mathbf{w}'_k]}_{\mathbf{0}} \tag{33}$$

$$E\left[(\mathbf{x}_k - E[\mathbf{x}_k])(\mathbf{x}_k - E[\mathbf{x}_k])^T\right] \approx \underbrace{E\left[\mathbf{w}'_k {\mathbf{w}'_k}^T\right]}_{\mathbf{Q}'_k} \tag{34}$$

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) \approx \mathcal{N}(\check{\mathbf{x}}_k + \mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}), \mathbf{Q}'_k) \tag{35}$$

Extended Kalman Filter

For the **statistical properties of the current measurement**, $\mathbf{y}_k$, given the current state we have

$$\mathbf{y}_k \approx \check{\mathbf{y}}_k + \mathbf{G}_k(\mathbf{x}_k - \check{\mathbf{x}}_k) + \mathbf{n}'_k \tag{36}$$

$$E[\mathbf{y}_k] \approx \check{\mathbf{y}}_k + \mathbf{G}_k(\mathbf{x}_k - \check{\mathbf{x}}_k) + \underbrace{E[\mathbf{n}'_k]}_{\mathbf{0}} \tag{37}$$

$$E\left[(\mathbf{y}_k - E[\mathbf{y}_k])(\mathbf{y}_k - E[\mathbf{y}_k])^T\right] \approx \underbrace{E\left[\mathbf{n}'_k \mathbf{n}'^T_k\right]}_{\mathbf{R}'_k} \tag{38}$$

$$p(\mathbf{y}_k|\mathbf{x}_k) \approx \mathcal{N}(\check{\mathbf{y}}_k + \mathbf{G}_k(\mathbf{x}_k - \check{\mathbf{x}}_k), \mathbf{R}'_k) \tag{39}$$

Extended Kalman Filter

Then the Bayes filter becomes

$$\underbrace{p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\mathcal{N}(\hat{\mathbf{x}}_k,\hat{\mathbf{P}}_k)} = \eta \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\mathcal{N}(\check{\mathbf{y}}_k+\mathbf{G}_k(\mathbf{x}_k-\check{\mathbf{x}}_k),\mathbf{R}'_k)} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k)}_{\mathcal{N}(\check{\mathbf{x}}_k+\mathbf{F}_{k-1}(\mathbf{x}_{k-1}-\check{\mathbf{x}}_{k-1}),\mathbf{Q}'_k)} \underbrace{p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})}_{\mathcal{N}(\hat{\mathbf{x}}_{k-1},\check{\mathbf{P}}_{k-1})} d\mathbf{x}_{k-1} \tag{40}$$

Using the formula on passing a Gaussian through a (stochastic) nonlinearity, we can see that the integral is also Gaussian

$$\underbrace{p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\mathcal{N}(\hat{\mathbf{x}}_k,\hat{\mathbf{P}}_k)} = \eta \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\mathcal{N}(\check{\mathbf{y}}_k+\mathbf{G}_k(\mathbf{x}_k-\check{\mathbf{x}}_k),\mathbf{R}'_k)} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k)p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1})}_{\mathcal{N}(\check{\mathbf{x}}_k,\mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}^T_{k-1}+\mathbf{Q}'_k)} d\mathbf{x}_{k-1} \tag{41}$$

Extended Kalman Filter

We are now left to work out the normalized product of two Gaussian PDFs. We find that

$$\underbrace{p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k})}_{\mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k)} = \underbrace{\eta p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) d\mathbf{x}_{k-1}}_{\mathcal{N}\left(\check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \check{\mathbf{y}}_k), (\mathbf{1} - \mathbf{K}_k \mathbf{G}_k)(\mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}'_k)\right)} \qquad (42)$$

where $\mathbf{K}_k$ is the **Kalman gain matrix**.

Extended Kalman Filter

By comparing left and right sides of the posterior expression, we arrive to the **classic recursive update equations for the EKF**

$$\begin{aligned}
\text{predictor:}\quad \check{\mathbf{P}}_k &= \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}'_k && (43)\\
\check{\mathbf{x}}_k &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}) && (44)\\
\text{Kalman gain:}\quad \mathbf{K}_k &= \check{\mathbf{P}}_k\mathbf{G}_k^T(\mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^T + \mathbf{R}'_k)^{-1} && (45)\\
\text{corrector:}\quad \hat{\mathbf{P}}_k &= (\mathbf{1} - \mathbf{K}_k\mathbf{G}_k)\check{\mathbf{P}}_k && (46)\\
\hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k\underbrace{(\mathbf{y}_k - \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}))}_{\text{innovation}} && (47)
\end{aligned}$$

The update equations allow us to compute $\left\{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\right\}$ from the estimates of the previous timestep $\left\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\right\}$

Extended Kalman Filter

The EKF equations have similar structure to KF for the linear-Gaussian case. Yet there are two important differences

1. The nonlinear motion and observation models are used to propagate the mean of the estimate.
2. There are Jacobians embedded in the $\mathbf{Q}'_k$ and $\mathbf{R}'_k$ covariances of the noise. This is because we allowed the noise to be applied within the nonlinearities in Eqs. (14,15).

Extended Kalman Filter

- There is no guarantee that the EKF will perform well for a general nonlinear system. Most likely one has to give it a try.
- The main problem with the EKF is the **operating point of the linearization** is the mean of the estimate and this is not the true state - or necessarily close to it.
- This can cause the EKF to diverge wildly in certain cases.
- Even when the result is not dramatic, often we have **biased** and/or **inconsistent** estimates. Hopefully, not too much.

Generalized Gaussian Filter

The Bayes filter can be written exactly. To reach implementable filters we perform different approximations on the form of the estimated PDF and the handling methods. **An alternative to deriving such filters starts by assuming a priori that the estimated PDF is Gaussian.** In general, we begin with a Gaussian prior at time $k-1$

$$p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}) \tag{48}$$

we pass this forwards in time through the nonlinear model, $\mathbf{f}(\cdot)$, to propose a Gaussian prior at time k and thus formulate the **prediction step** which incorporates the latest input $\mathbf{v}_k$.

$$p(\mathbf{x}_{k}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\check{\mathbf{x}}_{k}, \check{\mathbf{P}}_{k}) \tag{49}$$

Generalized Gaussian Filter

For the **correction step**, we write the joint Gaussian for the state and the latest measurement, at time k

$$p(\mathbf{x}_k, \mathbf{y}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{x,k} \\ \boldsymbol{\mu}_{y,k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx,k} & \boldsymbol{\Sigma}_{xy,k} \\ \boldsymbol{\Sigma}_{yx,k} & \boldsymbol{\Sigma}_{yy,k} \end{bmatrix} \right) \tag{50}$$

we now write the conditional Gaussian density for $\mathbf{x}_k$, the posterior estimate

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \mathcal{N}\left(\underbrace{\boldsymbol{\mu}_{x,k} + \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} (\mathbf{y}_k - \boldsymbol{\mu}_{y,k})}_{\hat{\mathbf{x}}_k}, \underbrace{\boldsymbol{\Sigma}_{xx,k} - \boldsymbol{\Sigma}_{xy,k} \boldsymbol{\Sigma}_{yy,k}^{-1} \boldsymbol{\Sigma}_{yx,k}}_{\hat{\mathbf{P}}_k} \right) \tag{51}$$

where $\hat{\mathbf{x}}_x, \hat{\mathbf{P}}_k$ are the mean and covariance respectively. The nonlinear observation model, $\mathbf{g}(\cdot)$, is used in the computation of $\boldsymbol{\mu}_{y,k}$.

Generalized Gaussian Filter

Then we can write the **generalized Gaussian correction-step equations** as

$$\mathbf{K}_k = \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1} \tag{52}$$

$$\hat{\mathbf{P}}_k = \check{\mathbf{P}}_k - \mathbf{K}_k\boldsymbol{\Sigma}_{xy,k}^{T} \tag{53}$$

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \boldsymbol{\mu}_{y,k}) \tag{54}$$

where we have let $\boldsymbol{\mu}_{x,k} = \check{\mathbf{x}}_k, \boldsymbol{\Sigma}_{xx,k} = \check{\mathbf{P}}_k$ and $\mathbf{K}_k$ is the Kalman gain.

Unfortunately, if the motion and observation models are nonlinear, then we cannot compute all the remaining quantities exactly: $\boldsymbol{\mu}_{y,k}$, $\boldsymbol{\Sigma}_{yy,k}$, $\boldsymbol{\Sigma}_{xy,k}$ because putting a Gaussian PDF through a nonlinearity returns to a non-Gaussian result.

Iterated Extended Kalman Filter

The goal is now to derive the **Iterated Extended Kalman Filter (IEKF)** exploiting the alternative derivation approach. For the **prediction step** we directly write notionally that for the prior at time k it holds

$$p(\mathbf{x}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) = \mathcal{N}(\check{\mathbf{x}}_k, \check{\mathbf{P}}_k) \tag{55}$$

which incorporates $\mathbf{v}_k$.

The **correction step** is more involved and we shall perform the derivations.

Iterated Extended Kalman Filter

Our nonlinear measurement model is given by

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k) \tag{56}$$

by **linearizing** around an arbitrary **operating point**, $\mathbf{x}_{op,k}$

$$\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k) \approx \mathbf{y}_{op,k} + \mathbf{G}_k(\mathbf{x}_k - \mathbf{x}_{op,k}) + \mathbf{n}'_k \tag{57}$$

$$\mathbf{y}_{op,k} = \mathbf{g}(\mathbf{x}_{op,k}, \mathbf{0}), \;\; \mathbf{G}_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{op,k}, \mathbf{0}}, \;\; \mathbf{n}'_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{n}_k}\right|_{\mathbf{x}_{op,k}, \mathbf{0}} \mathbf{n}_k \tag{58}$$

where the *Jacobians are evaluated at* $\mathbf{x}_{op,k}$.

Iterated Extended Kalman Filter

Using the linearized model, we express the **joint density for the state and the measurement** at time k as **approximately Gaussian**

$$p(\mathbf{x}_k, \mathbf{y}_k | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k-1}) \approx \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{x,k} \\ \boldsymbol{\mu}_{y,k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx,k} & \boldsymbol{\Sigma}_{xy,k} \\ \boldsymbol{\Sigma}_{yx,k} & \boldsymbol{\Sigma}_{yy,k} \end{bmatrix} \right)$$
$$= \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}}_k \\ \mathbf{y}_{op,k} + \mathbf{G}_k(\check{\mathbf{x}}_k - \mathbf{x}_{op,k}) \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}}_k & \check{\mathbf{P}}_k \mathbf{G}_k^T \\ \mathbf{G}_k \check{\mathbf{P}}_k & \mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k \end{bmatrix} \right) \tag{59}$$

Iterated Extended Kalman Filter

If the measurement, $\mathbf{y}_k$, is known then we write the Gaussian conditional probability for $\mathbf{x}_k$ (i.e., the posterior)

$$p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \mathcal{N}\left(\underbrace{\boldsymbol{\mu}_{x,k} + \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1}(\mathbf{y}_k - \boldsymbol{\mu}_{y,k})}_{\hat{\mathbf{x}}_k}, \underbrace{\boldsymbol{\Sigma}_{xx,k} - \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1}\boldsymbol{\Sigma}_{xy,k}}_{\hat{\mathbf{P}}_k}\right) \quad (60)$$

where again $\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k$ are the mean and covariance, respectively.

Iterated Extended Kalman Filter

As shown shown earlier, the **generalized Gaussian correction-step** equations are (repeated)

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

Substituting in the moments $\boldsymbol{\mu}_{y,k}, \mathbf{\Sigma}_{yy,k}, \mathbf{\Sigma}_{xy,k}$ from above we have

$$
\begin{aligned}
\mathbf{K}_k &= \check{\mathbf{P}}_k\mathbf{G}_k^T(\mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^T + \mathbf{R}_k')^{-1} && (61) \\
\hat{\mathbf{P}}_k &= (\mathbf{1} - \mathbf{K}_k\mathbf{G}_k)\check{\mathbf{P}}_k && (62) \\
\hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{y}_{op,k} - \mathbf{G}_k(\check{\mathbf{x}}_k - \mathbf{x}_{op,k})) && (63)
\end{aligned}
$$

these equations are very similar to the classical Kalman gain and corrector equations with *the only difference being the operating point of the linearization*.

Iterated Extended Kalman Filter

If in these revised equations (repeated here)

$$\begin{aligned}
\mathbf{K}_k &= \check{\mathbf{P}}_k \mathbf{G}_k^T (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k)^{-1} \\
\hat{\mathbf{P}}_k &= (\mathbf{1} - \mathbf{K}_k \mathbf{G}_k)\check{\mathbf{P}}_k \\
\hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{y}_{op,k} - \mathbf{G}_k(\check{\mathbf{x}}_k - \mathbf{x}_{op,k}))
\end{aligned}$$

we set the operating point of the linearization to be the mean of the predicted prior, $\mathbf{x}_{op,k} = \check{\mathbf{x}}_k$ then Eqs. (61-63) are **identical** to the classical EKF Eqs. (45-47).

Iterated Extended Kalman Filter

However, the Eqs. (61-63) indicate that **we can do better than the classical implementation**. Specifically, if we iteratively recompute Eqs. (61-63) each time setting the operating point to be the mean of the posterior at the last iteration

$$\mathbf{x}_{op,k} \leftarrow \hat{\mathbf{x}}_k \tag{64}$$

where in the first iteration we take $\mathbf{x}_{op,k} = \check{\mathbf{x}}_k$.

This allows to **linearize about better and better estimates, thereby improving our approximation at each iteration**. We terminate the process when the derived change in $\mathbf{x}_{op,k}$ between two iterations is very small.

Importantly, the covariance equation needs to be computed only once, after the other two equations converged based on the criterion above.

IEKF is a MAP Estimator

Relationship between the EKF/IEKF estimate and the full Bayesian posterior?

At the level of the correction step at a single timestep, the IEKF estimate corresponds to a *(local)* maximum of the full posterior. In other words, it is a MAP estimate. However, since the EKF is not iterated, it can be far from a local maximum. In fact, there is very little we can say about its relationship to the full posterior.

Iterated Extended Kalman Filter

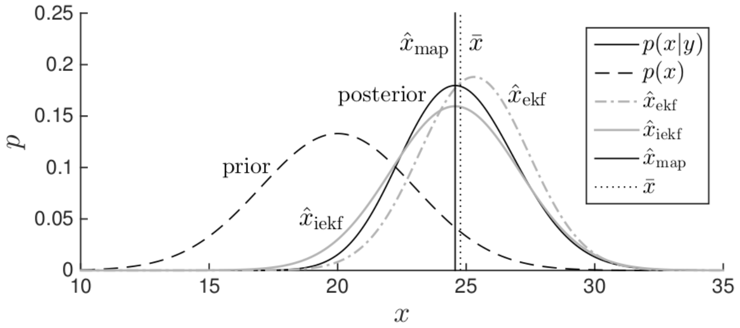


Figure: Stereo camera example, comparing the inference (i.e., “corrective”) step of the EKF and IEKF to the full Bayesian posterior ,$p(x|y)$. The mean of the IEKF matches up against the MAP solution, $\hat{x}_{map}$, while the EKF does not. The actual mean of the posterior is denoted as $\bar{x}$.

Iterated Extended Kalman Filter

The choice to iteratively re-linearize about the best guess leads to a MAP solution. *Thus, the "mean" of the IEKF Gaussian estimator does not actually match the mean of the full Bayesian posterior;* **it matches the mode**.

Alternatives for Passing PDFs through Nonlinearities

For both EKF/IEKF we performed linearization of the nonlinear model about an operating point and then passed our Gaussian PDFs through the linearized model analytically. *Other approaches are possible*. Three common techniques

1. Linearization (as in the EKF)
2. Monte Carlo (brute force) method for sampling
3. Sigmapoint (or unscented) transformation

The different methods can be used in conjunction with the Bayes filter to derive alternatives to EKF/IEKF.

Monte Carlo Method

The Monte Carlo method of transforming a PDF through a nonlinearity is the “brute force” approach to the problem. Essentially, we draw a large number of samples from the input density, transform each sample through the nonlinearity exactly, and then build the output density from the transformed samples (e.g., by computing statistical moments).

The *law of large numbers* ensures that this process will converge to the correct answer as the number of samples approaches infinity.

Monte Carlo Method

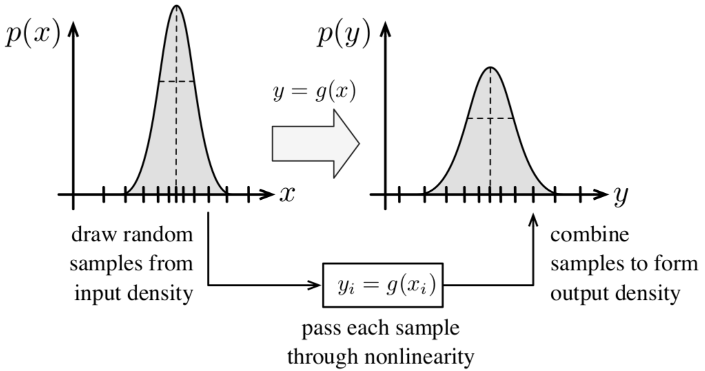

$p(x)$
x
$y = g(x)$
$p(y)$
y
draw random samples from input density
$y_i = g(x_i)$
pass each sample through nonlinearity
combine samples to form output density

Figure: Monte Carlo method to transform a PDF through a nonlinearity. A large number of random samples are drawn from the input density, passed through the nonlinearity, and then used to form the output density.

Monte Carlo Method

Main Disadvantage of the Monte Carlo method

- It can be extremely computationally expensive, especially when the dimensionality grows.

Advantages of the Monte Carlo method

- It works with any PDF, not just Gaussian.
- It handles any type of nonlinearity (no requirement for differentiable or even continuous)
- We do not need to know the mathematical form of the nonlinear function - in practice could be any software function.
- It is an “anytime” algorithm - we can easily trade-off accuracy against speed by choosing the number of samples appropriately (*up to some extent in practice*).

As this method can be very accurate, it is also a benchmark for the others.

Monte Carlo Method

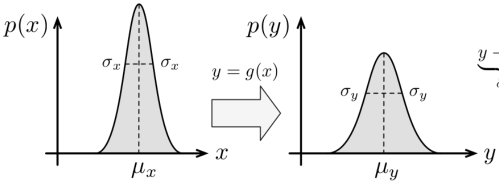
Note: the mean of the output density is not the same as the mean of the input density after being passed through the nonlinearity. **This is key and the Monte Carlo method can approach the correct answer with a large number of samples, but other methods cannot.**

Linearization

The linearization-based approach is still the most widely used (EKF/IEKF). In this case, the mean is actually passed through the nonlinearity exactly, while the covariance is approximately passed through a linearized version of the function.

Typically, the operating point of the linearization process is the mean of the PDF.

Linearization



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

Figure: One-dimensional Gaussian PDF transformed through a deterministic nonlinear function, $g(\cdot)$. In the example, we linearize the nonlinearity to propagate the variance approximately.

Linearization

The linearization method **is highly inaccurate** because

- The outcome of passing a Gaussian PDF through a nonlinear function will not be another Gaussian PDF. By keeping only the mean and covariance of the posterior PDF, we are approximating the posterior (by throwing away higher statistical moments).
- We are approximating the covariance of the true output PDF by linearizing the nonlinear function.
- The operating point about which we linearize the nonlinear function is often not the true mean of the prior PDF, but rather our estimate of the mean of the input PDF. This is an approximation that introduces error.
- We are approximating the mean of the true output PDF by simply passing the mean of the prior PDF through the nonlinear function. This does not represent the true mean of the output.

Furthermore, another disadvantage is that

- We need to be able to either calculate the Jacobian of the nonlinearity in closed form, or compute it numerically (which introduces yet another approximation).

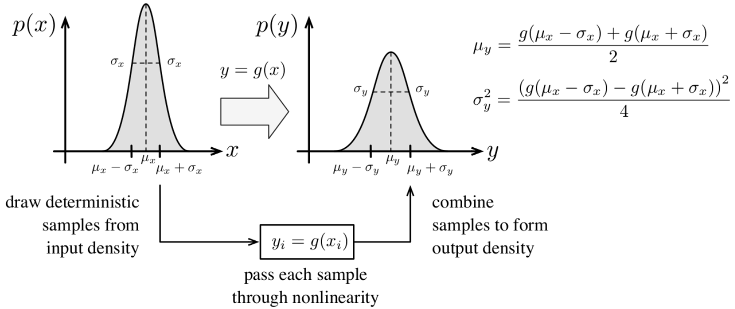
Linearization

Advantages of the linearization method, include

- If the function is only slightly nonlinear, and the input PDF is Gaussian, the linearization method is very simple to understand and quick to implement.
- The procedure is reversible (if the nonlinearity is locally invertible). This means we can recover the input PDF exactly by passing the output PDF through the inverse of the nonlinearity (using the same linearization procedure).
- This is not true for all methods discussed, and it is not applicable to the sigmapoint (unscented) transformation (to follow).

Sigmapoint(Unscented) Transformation

The Sigmapoint(unscented) Transformation acts as the "compromise" between the expensive accuracy of the Monte Carlo method and the speed and inaccuracy of the linearization method. **The sigmapoint transformation (or unscented transformation) is a family of transformations.**



$p(x)$
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

Figure: One-dimensional Gaussian PDF transformed through a deterministic nonlinear function, $g(\cdot)$. Here the basic sigmapoint(unscented) transformation is used in which only 2 deterministic samples (one on each side of the mean) approximate the input density.

Sigmapoint(Unscented) Transformation

In general, a version of the sigmapoint (SP), or unscented, transformation is used that includes one additional sample beyond the basic version at the mean of the input density. The steps are `Step 1.` A set of $2L+1$ sigmapoints is computed from the input density $\mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$

$$\mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma}_{xx}, \ \text{(Cholesky decomposition, } \mathbf{L} \text{ lower-triangular)} \quad (65)$$

$$\mathbf{x}_0 = \boldsymbol{\mu}_x \quad (66)$$

$$\mathbf{x}_i = \boldsymbol{\mu}_x + \sqrt{L+\kappa}\,\mathrm{col}_i\mathbf{L}, \ i = 1 \ldots L \quad (67)$$

$$\mathbf{x}_{i+L} = \boldsymbol{\mu}_x - \sqrt{L+\kappa}\,\mathrm{col}_i\mathbf{L}, \ i = 1 \ldots L \quad (68)$$

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

$$\boldsymbol{\mu}_x = \sum_{i=0}^{2L} a_i \mathbf{x}_i, \quad \boldsymbol{\Sigma}_{xx} = \sum_{i=0}^{2L} a_i(\mathbf{x}_i - \boldsymbol{\mu}_x)(\mathbf{x}_i - \boldsymbol{\mu}_x)^T \quad (69)$$

$$a_i = \begin{cases} \frac{\kappa}{L+\kappa}, & i = 0 \\ \frac{1}{2}\frac{1}{L+\kappa}, & \text{otherwise} \end{cases} \quad (70)$$

with a_i summing up to 1.

Sigmapoint(Unscented) Transformation

`Step 2.` Each of the sigmapoints is individually passed through the nonlinearity, $\mathbf{g}(\cdot)$

$$\mathbf{y}_i = \mathbf{g}(\mathbf{x}_i), \ \ i = 0 ... 2L \tag{71}$$

`Step 3.` The mean of the output density, $\boldsymbol{\mu}_y$, is computed as

$$\boldsymbol{\mu}_y = \sum_{i=0}^{2L} a_i, \mathbf{y}_i \tag{72}$$

`Step 4.` The covariance of the output density, $\boldsymbol{\Sigma}_{yy}$, is computed as

$$\boldsymbol{\Sigma}_{yy} = \sum_{i=0}^{2L} a_i (\mathbf{y}_i - \boldsymbol{\mu}_y)(\mathbf{y}_i - \boldsymbol{\mu}_y)^T \tag{73}$$

`Step 5.` The output density, $\mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy})$, is returned.

Sigmapoint(Unscented) Transformation

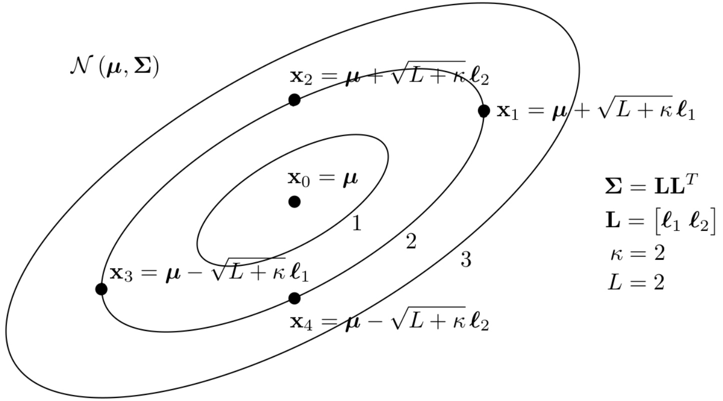

$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
$\mathbf{x}_2 = \boldsymbol{\mu} + \sqrt{L+\kappa}\,\boldsymbol{\ell}_2$
$\mathbf{x}_1 = \boldsymbol{\mu} + \sqrt{L+\kappa}\,\boldsymbol{\ell}_1$
$\mathbf{x}_0 = \boldsymbol{\mu}$
$\mathbf{x}_3 = \boldsymbol{\mu} - \sqrt{L+\kappa}\,\boldsymbol{\ell}_1$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

Figure: Two-dimensional ($L = 2$) Gaussian PDF, whose covariance is displayed using elliptical equiprobable contours of $1, 2, 3$ standard deviations, and the corresponding $2L + 1 = 5$ sigmapoints for $\kappa = 2$.

Particle Filter

As discussed, drawing a large number of samples is one way to approximate a PDF and by passing each of them through a nonlinearity and subsequently recombining them on the other side, we can get an approximation of the transformation of a PDF. This idea is extended to an approximation of the Bayes filter, called the **Particle Filter (PF)**.

The Particle Filter negotiates nonlinearity and non-Gaussian noise

The particle filter is able to handle non-Gaussian noise, as well as nonlinear observation and motion models!

The Particle Filter is practical

The particle filter does not require that we have analytical expressions for $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, not for their derivatives. It thus can handle very complex systems, for which maybe we have a simulator available.

Particle Filter

- There are many different variants of the particle filter.
- In the presented approach we rely on *sample importance resampling* where the so-called **proposal PDF** is the prior PDF, in the Bayes filter, propagated forward using the motion model and the latest motion measurement, $\mathbf{v}_k$.
- This version of the particle filter is also called the *bootstrap algorithm*, the *condensation algorithm*, or the *survival-of-the-fittest algorithm*.

Particle Filter Algorithm

The main steps of the particle filter are

`Step 1.` Draw M samples from the joint density comprising the prior and the motion noise

$$\begin{bmatrix} \hat{\mathbf{x}}_{k-1,m} \\ \mathbf{w}_{k,m} \end{bmatrix} \leftarrow p(\mathbf{x}_{k-1}|\check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{1:k-1})p(\mathbf{w}_k) \tag{74}$$

where m is the unique particle index. In practice, we can draw from each factor of this joint density separately.

`Step 2.` Generate a **prediction of the posterior PDF** by using $\mathbf{v}_k$ which is done by passing each prior particle/noise sample through the nonlinear motion model

$$\check{\mathbf{x}}_{k,m} = \mathbf{f}(\hat{\mathbf{x}}_{k-1,m}, \mathbf{v}_k, \mathbf{w}_{k,m}) \tag{75}$$

These new "predicted states" together approximate the density, $p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k-1})$

Particle Filter Algorithm

`Step 3.` **Correct the posterior PDF** by incorporating $\mathbf{y}_k$ by
`step 3a:` first assigning 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

$$w_{k,m} = \frac{p(\check{\mathbf{x}}_{k,m} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k})}{p(\check{\mathbf{x}}_{k,m} | \check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k-1})} = \eta p(\mathbf{y}_k | \check{\mathbf{x}}_{k,m}) \tag{76}$$

where η a normalization constant. In practice, this is done by simulating an expected sensor reading, $\check{\mathbf{y}}_{k,m}$, using the nonlinear observation model (possibly not available analytically)

$$\check{\mathbf{y}}_{k,m} = \mathbf{g}(\check{\mathbf{x}}_{k,m}, \mathbf{0}) \tag{77}$$

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

Particle Filter Algorithm

`step 3b:` Resample the posterior based on the weight assigned to each predicted posterior particle

$$\hat{\mathbf{x}}_{k,m} \overbrace{\leftarrow}^{\text{resample}} \{\check{\mathbf{x}}_{k,m}, w_{k,m}\} \tag{78}$$

(see next slide)

Particle Filter Resampling

A key aspect of PF is the need to resample the posterior density according to the weights assigned to each current sample. One way for achieving this shall be described.

Particle Filter Resampling

We assume we have M samples and that each of these is assigned an unnormalized weight, $w_m \in \mathbb{R} > 0$. From the weights, we create bins with boundaries, β_m

$$\beta_m = \frac{\sum_{n=1}^{m} w_n}{\sum_{\ell=1}^{M} w_\ell} \tag{79}$$

where the β_m define the boundaries of M bins on the interval $[0, 1]$ as

$$0 \leq \beta_1 \leq \beta_2 \leq ... \leq \beta_{M-1} \leq 1$$

where we will have $\beta_M \equiv 1$.

Particle Filter Resampling

- Subsequently, we select a random number, ρ, sampled from a uniform density on $[0, 1)$.
- For M iterations we add to the new list of samples, the sample whose bin contains ρ.
- At each iteration we step ρ forward by $1/M$.
- The algorithm guarantees that all bins whose size is greater than $1/M$ will have sample in the new list.

Particle Filter - Practical Considerations

1. How many samples to use? Depends on the specific problem. Typically hundreds of particles for a low-dimensional problem (e.g, a 2D robot with $\mathbf{x} = [x, y, \theta]$).
2. We can dynamically pick the number of particles online using a heuristic sample such as $\sum w_{k,m} \geq w_{thres}$ (the latter experimentally derived).
3. We do not always need to resample every time we go through the algorithm. We may delay resampling, but then carry the weights forward to the next iteration.
4. It is always smart to include a subset of samples that are uniformly drwn from the entire state space. This protects from outlier sensor measurements/vehicle movements.
5. For high-dimensional state estimation problems, PF is expensive. Check for variations.
6. The PF is an "anytime" algorithm analogously to the Monte Carlo method.
7. The Cramér-Rao lower bound (CRLB) is set by the uncertainty in the measurements that we have available. CRLB is the best we could do.

Sigmapoint (Unscented) Kalman Filter

The SigmaPoint (or unscented) Kalman Filter (SPKF or UKF) is motivated from the idea to completely eliminate the step of linearization and instead use the sigmapoint transformation to pass PDFs through the nonlinear motion and observation models.

Prediction Step

The prediction step is a fairly straightforward application of the sigmapoint transformation since we are trying to bring our prior forward in time through the motion model. To that end, we employ the following steps to go from the **prior belief**, $\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\}$, to the **predicted belief**, $\{\check{\mathbf{x}}_k, \check{\mathbf{P}}_k\}$

`Step 1.` Both the prior belief and the motion noise have uncertainty, so these are stacked together

$$\boldsymbol{\mu}_z = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \hat{\mathbf{P}}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k \end{bmatrix} \tag{80}$$

where $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation. Let $L = \dim(\boldsymbol{\mu}_z)$.

Prediction Step

`Step 2.` Convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation (Eqs. repeated)

$$\mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma}_{zz}, \quad \text{(Cholesky decomposition, } \mathbf{L} \text{ lower-triangular)} \tag{81}$$

$$\mathbf{z}_0 = \boldsymbol{\mu}_z \tag{82}$$

$$\mathbf{z}_i = \boldsymbol{\mu}_z + \sqrt{L+\kappa}\,\mathrm{col}_i\mathbf{L}, \quad i = 1...L \tag{83}$$

$$\mathbf{z}_{i+L} = \boldsymbol{\mu}_z - \sqrt{L+\kappa}\,\mathrm{col}_i\mathbf{L}, \quad i = 1...L \tag{84}$$

`Step 3.` Unstack each sigmapoint into state and motion noise

$$\mathbf{z}_i = \begin{bmatrix} \hat{\mathbf{x}}_{k-1,i} \\ \mathbf{w}_{k,i} \end{bmatrix} \tag{85}$$

and pass each sigmapoint through the nonlinear motion model

$$\check{\mathbf{x}}_{k,i} = \mathbf{f}(\hat{\mathbf{x}}_{k-1,i}, \mathbf{v}_k, \mathbf{w}_{k,i}), \quad i = 0...2L \tag{86}$$

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

Prediction Step

`Step 4.` Recombine the transformed sigmapoints into the predicted belief $\{\check{\mathbf{x}}_k, \check{\mathbf{P}}_k\}$

$$\check{\mathbf{x}}_k = \sum_{i=0}^{2L} a_i \check{\mathbf{x}}_{k,i} \tag{87}$$

$$\check{\mathbf{P}}_k = \sum_{i=0}^{2L} a_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)(\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)^T \tag{88}$$

$$a_i = \begin{cases} \frac{\kappa}{L+\kappa}, & i = 0 \\ \frac{1}{2}\frac{1}{L+\kappa}, & \text{otherwise} \end{cases} \tag{89}$$

Correction Step

Recal that the conditional Gaussian density for $\mathbf{x}_k$ (i.e., the posterior) takes the form

$$p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{0:k}) = \mathcal{N}\left(\underbrace{\boldsymbol{\mu}_{x,k} + \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1}(\mathbf{y}_k - \boldsymbol{\mu}_{y,k})}_{\hat{\mathbf{x}}_k}, \underbrace{\boldsymbol{\Sigma}_{xx,k} - \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1}\boldsymbol{\Sigma}_{yx,k}}_{\hat{\mathbf{P}}_k}\right) \quad (90)$$

where we $\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k$ are the mean and covariance, respectively. In this form we write the generalized Gaussian correction-step equations as

$$\mathbf{K}_k = \boldsymbol{\Sigma}_{xy,k}\boldsymbol{\Sigma}_{yy,k}^{-1} \quad (91)$$
$$\hat{\mathbf{P}}_k = \check{\mathbf{P}}_k - \mathbf{K}_k\boldsymbol{\Sigma}_{xy,k}^{T} \quad (92)$$
$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \boldsymbol{\mu}_{y,k}) \quad (93)$$

and we shall use the SP transformation to derive improved estimates of $\boldsymbol{\mu}_{y,k}, \boldsymbol{\Sigma}_{yy,k}, \boldsymbol{\Sigma}_{xy,k}$.

Correction Step

The following steps are involved

`Step 1.` Both the predicted belief and the observation noise are uncertain so we stack them

$$\boldsymbol{\mu}_z = \begin{bmatrix} \check{\mathbf{x}}_k \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{zz} = \begin{bmatrix} \check{\mathbf{P}}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{bmatrix} \tag{94}$$

where $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation. Let $L = \dim(\boldsymbol{\mu}_z)$.

`Step 2.` Convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation

$$\mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma}_{zz}, \quad \text{(Cholesky decomposition, } \mathbf{L} \text{ lower-triangular)} \tag{95}$$

$$\mathbf{z}_0 = \boldsymbol{\mu}_z \tag{96}$$

$$\mathbf{z}_i = \boldsymbol{\mu}_z + \sqrt{L+\kappa}\,\mathrm{col}_i \mathbf{L}, \quad i = 1 \ldots L \tag{97}$$

$$\mathbf{z}_{i+L} = \boldsymbol{\mu}_z - \sqrt{L+\kappa}\,\mathrm{col}_i \mathbf{L}, \quad i = 1 \ldots L \tag{98}$$

Correction Step

`Step 3.` Unstack each sigmapoint into state and observation noise

$$\mathbf{z}_i = \begin{bmatrix} \check{\mathbf{x}}_{k,i} \\ \mathbf{n}_{k,i} \end{bmatrix} \tag{99}$$

and pass each sigmapoint through the nonlinear observation model

$$\check{\mathbf{y}}_{k,i} = \mathbf{g}(\check{\mathbf{x}}_{k,i}, \mathbf{n}_{k,i}) \tag{100}$$

Correction Step

`Step 4.` Recombine the transformed sigmapoints into the desired moments

$$\boldsymbol{\mu}_{y,k} = \sum_{i=0}^{2L} a_i \check{\mathbf{y}}_{k,i} \tag{101}$$

$$\boldsymbol{\Sigma}_{yy,k} = \sum_{i=0}^{2L} a_i (\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})(\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})^T \tag{102}$$

$$\boldsymbol{\Sigma}_{xy,k} = \sum_{i=0}^{2L} a_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)(\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})^T \tag{103}$$

$$a_i = \begin{cases} \frac{\kappa}{L+\kappa}, & i = 0 \\ \frac{1}{2}\frac{1}{L+\kappa}, & \text{otherwise} \end{cases} \tag{104}$$

which are plugged into the generalized Gaussian correction-step equations to complete the correction step.

Sigmapoint (Unscented) Kalman Filter Advantages

1. It does not require the nonlinear motion and observatiion models in closed form; they can be black-box functions.
2. It does not require any analytical derivatives
3. It uses only basic linear algebra operations

Taxonomy of Filters

Let us remind the generally assumed motion and observation models:

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

where k is the discrete-time index and K its maximum, $\mathbf{f}$ the nonlinear motion model, and $\mathbf{g}$ the nonlinear observation model. Estimation as we originallyc considered it is a Non-Linear, Non-Gaussian problem.

Taxonomy of Filters

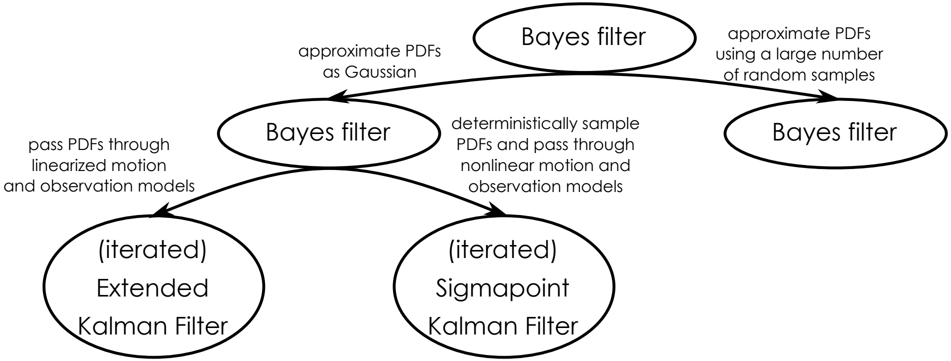

Bayes filter
approximate PDFs as Gaussian
approximate PDFs using a large number of random samples
Bayes filter
Bayes filter
pass PDFs through linearized motion and observation models
deterministically sample PDFs and pass through nonlinear motion and observation models
(iterated) Extended Kalman Filter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

Figure: Taxonomy of the different filtering methods and their relationships with the Bayes filter.

Taxonomy of Filters

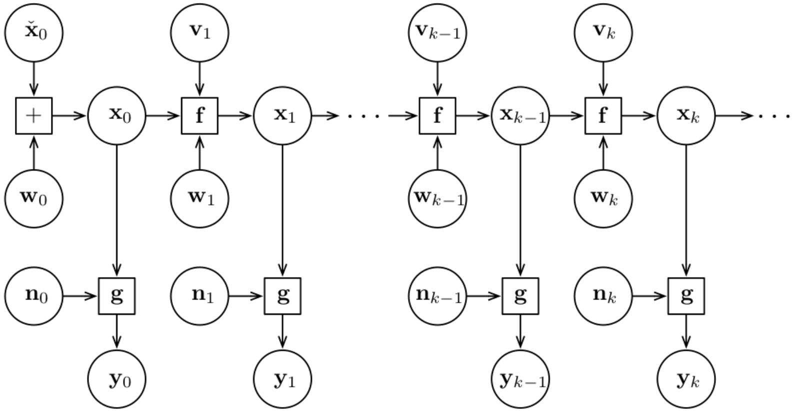


Figure: Graphical model representation of the Markov processes constituting the Non-Linear, Non-Gaussian (NLNG) system in Eqs. (14,15).

Taxonomy of Filters

Markov Process

In the simplest sense, a stochastic process has the Markov property if the conditional probability density functions (PDFs) of future states of the process, given the present state, depend only upon the present state, but not only other past states, i.e., they are conditionally independent of these older states. Such a process is called Markovian or a Markov process.

Extended Kalman Filter

Classic recursive update equations for the EKF

$$\begin{aligned}
\text{predictor:}\quad & \check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}'_k \\
& \check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}) \\
\text{Kalman gain:}\quad & \mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{G}_k^T(\mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^T + \mathbf{R}'_k)^{-1} \\
\text{corrector:}\quad & \hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{G}_k)\check{\mathbf{P}}_k \\
& \hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\underbrace{(\mathbf{y}_k - \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}))}_{\text{innovation}}
\end{aligned}$$

The update equations allow us to compute $\left\{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\right\}$ from the estimates of the previous timestep $\left\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\right\}$

Iterated Extended Kalman Filter

IEKF revisits the correction step of EKF which then takes the form

$$\begin{aligned}
\mathbf{K}_k &= \check{\mathbf{P}}_k \mathbf{G}_k^T (\mathbf{G}_k \check{\mathbf{P}}_k \mathbf{G}_k^T + \mathbf{R}'_k)^{-1} \\
\hat{\mathbf{P}}_k &= (\mathbf{1} - \mathbf{K}_k \mathbf{G}_k) \check{\mathbf{P}}_k \\
\hat{\mathbf{x}}_k &= \check{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{y}_k - \mathbf{y}_{op,k} - \mathbf{G}_k (\check{\mathbf{x}}_k - \mathbf{x}_{op,k}))
\end{aligned}$$

these equations are very similar to the classical Kalman gain and corrector equations with *the only difference being the operating point of the linearization*.

Iterated Extended Kalman Filter

Relationship between the EKF/IEKF estimate and the full Bayesian posterior?

At the level of the correction step at a single timestep, the IEKF estimate corresponds to a *(local)* maximum of the full posterior. In other words, it is a MAP estimate. However, since the EKF is not iterated, it can be far from a local maximum. In fact, there is very little we can say about its relationship to the full posterior.

Particle Filter Algorithm

`Step 1.` Draw M samples from the joint density comprising the prior and the motion noise

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

where m is the unique particle index.
`Step 2.` Generate a **prediction of the posterior PDF** by using $\mathbf{v}_k$ (by passing each prior particle/noise sample through the nonlinear motion model)

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

These new "predicted states" together approximate the density, $p(\mathbf{x}_k|\check{\mathbf{x}}_0, \mathbf{v}_{1:k}, \mathbf{y}_{1:k-1})$

Particle Filter

`Step 3.` **Correct the posterior PDF** by incorporating $\mathbf{y}_k$ by
`step 3a:` first assigning 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

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

η a normalization constant. This is done by simulating an expected sensor reading, $\check{\mathbf{y}}_{k,m}$

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

then we assume $p(\mathbf{y}_k|\check{\mathbf{x}}_{k,m}) = p(\mathbf{y}_k|\check{\mathbf{y}}_{k,m})$.
`step 3b:` Resample the posterior based on the weight assigned to each predicted posterior particle

$$\hat{\mathbf{x}}_{k,m} \overbrace{\leftarrow}^{\text{resample}} \{\check{\mathbf{x}}_{k,m}, w_{k,m}\}$$

Sigmapoint (Unscented) Kalman Filter

Prediction step: We employ the following steps to go from the **prior belief**, $\{\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{P}}_{k-1}\}$, to the **predicted belief**, $\{\check{\mathbf{x}}_k, \check{\mathbf{P}}_k\}$

`Step 1.` Both the prior belief and the motion noise have uncertainty, so these are stacked together

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

where $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation. Let $L = \dim(\boldsymbol{\mu}_z)$.
`Step 2.` Convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation (Eqs. repeated)

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

Sigmapoint (Unscented) Kalman Filter

`Step 3.` Unstack each sigmapoint into state and motion noise

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

and pass each sigmapoint through the nonlinear motion model

$$\check{\mathbf{x}}_{k,i} = \mathbf{f}(\hat{\mathbf{x}}_{k-1,i}, \mathbf{v}_k, \mathbf{w}_{k,i}), \;\; i = 0...2L$$

where the latest input, $\mathbf{v}_k$, is required.
`Step 4.` Recombine the transformed sigmapoints into the predicted belief $\{\check{\mathbf{x}}_k, \check{\mathbf{P}}_k\}$

$$\check{\mathbf{x}}_k = \sum_{i=0}^{2L} a_i \check{\mathbf{x}}_{k,i}, \;\; \check{\mathbf{P}}_k = \sum_{i=0}^{2L} a_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)(\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)^T, \;\; a_i = \begin{cases} \frac{\kappa}{L+\kappa}, \;\; i = 0, \;\; \frac{1}{2}\frac{1}{L+\kappa}, \;\; \text{otherwise} \end{cases}$$

Sigmapoint (Unscented) Kalman Filter

Correction step: The following steps are involved `Step 1.` Both the predicted belief and the observation noise are uncertain so we stack them

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

where $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ is still a Gaussian representation. Let $L = \dim(\boldsymbol{\mu}_z)$.
`Step 2.` Convert $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}\}$ to a sigmapoint representation

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

Sigmapoint (Unscented) Kalman Filter

`Step 3.` Unstack each sigmapoint into state and observation noise

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

and pass each sigmapoint through the nonlinear observation model

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

`Step 4.` Recombine the transformed sigmapoints into the desired moments

$$\boldsymbol{\mu}_{y,k} = \sum_{i=0}^{2L} a_i \check{\mathbf{y}}_{k,i}, \ \boldsymbol{\Sigma}_{yy,k} = \sum_{i=0}^{2L} a_i (\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})(\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})^T \tag{105}$$

$$\boldsymbol{\Sigma}_{xy,k} = \sum_{i=0}^{2L} a_i (\check{\mathbf{x}}_{k,i} - \check{\mathbf{x}}_k)(\check{\mathbf{y}}_{k,i} - \boldsymbol{\mu}_{y,k})^T, \ a_i = \begin{cases} \frac{\kappa}{L+\kappa}, & i = 0 \\ \frac{1}{2}\frac{1}{L+\kappa}, & \text{otherwise} \end{cases}$$

which are plugged into the generalized Gaussian correction-step equations to complete the correction step.

Thank you

Q&A

Assignments

Other matters