ECEN426
Advanced Mechatronic Systems
Nonlinear Kalman Filters

Christopher Hollitt

September 28, 2021

Nonlinear estimation

The Kalman filter is a useful framework for estimation, however, in its basic form it is not suitable for our SLAM problem.

Recall the assumptions of the Kalman filter;

- The system can be modeled as linear. However, localisation of a robot in more than one dimension is nonlinear.
- The noise is zero mean gaussian noise. This is sometimes not reasonalbe, such as when measuring very short ranges. But we can usually get away with this assumption.

We will need to look at how we can extend the Kalman filter to cope with nonlinear problems.

We will also need to give the Kalman filter flexibility to change the size of the state vector that it is trying to estimate as we build a map.

Robot Localisation

As a foretaste of SLAM, let's consider the model of a robot moving in two dimensions.

1. We will regard the robot's state as being completely captured by its x and y position, and its heading angle θ (described in its usual anticlockwise orientation from the x axis).
2. For simplicity we will consider the robot to be intertialess, in the sense that we will assume that the robot's future state is not dependent on its velocity (either linear or angular).
3. The robot can drive forwards (or backwards) according to a command signal u_1, and can turn according to u_2.

Robot localisation

Consider the model for the robot. That is, we wish to predict the next state of the robot given where we currently are, and what we intend to do with the command signal.

$$
\begin{aligned}
s_{\mathrm{x}}(t+1) &= s_{\mathrm{x}}(t) + u_1 \cos\theta \\
s_{\mathrm{y}}(t+1) &= s_{\mathrm{y}}(t) + u_1 \sin\theta \\
\theta(t+1) &= \theta(t) + u_2(t)
\end{aligned}
$$

We then measure the range and heading to some beacon at known point (ξ, ν).

$$
\begin{aligned}
y_1(t+1) &= \sqrt{\left(s_{\mathrm{x}} - \xi\right)^2 + \left(s_{\mathrm{y}} - \nu\right)^2} && \text{Range} \\
y_2(t+1) &= \arctan\left(\frac{s_{\mathrm{y}} - \nu}{s_{\mathrm{x}} - \xi}\right) - \theta && \text{Bearing}
\end{aligned}
$$

Plan

We will begin by looking at several methods that extend the Kalman filter to nonlinear problems.

- The extended Kalman filter (EKF) will be the one we use, so we will look at that method in most detail.
- Other more recent methods are widely used in real SLAM methods, so we will briefly examine those so you have some familiarity with how they are intended to operate.

We will then specialise to the SLAM problem specifically, and formulate the EKF for that particular problem.

Estimation for Nonlinear Systems

We want to estimate the state of a general nonlinear system described by the nonlinear discrete-time equations

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)\big) \\ \boldsymbol{y}(t) &= \boldsymbol{h}\big(\boldsymbol{x}(t), \boldsymbol{v}(t)\big) \\ \boldsymbol{w} &\sim \mathcal{N}(0, \boldsymbol{Q}) \\ \boldsymbol{v} &\sim \mathcal{N}(0, \boldsymbol{R}) \\ t &\in \mathbb{Z}_+ \end{aligned}$$

We have made $\boldsymbol{Q}$ and $\boldsymbol{R}$ time invariant for simplicity, though they do not need to be so.

Similarly we have made $\boldsymbol{f}$ and $\boldsymbol{h}$ time invariant.

Relatives of the Kalman filter

There are a family of variants of the Kalman filter that do a better job when estimating state in non-linear systems.

Extended Kalman Filter (EKF) Allows for a nonlinear plant by forming a linearized model at each iteration. Still assumes that noise is Gaussian and evolves the estimation distribution according to the linearized model.

Unscented Kalman Filter (UKF) Does a better job of modelling noise in a non-linear system by attempting to capture the distortion of the noise distribution as the system acts upon it.

Particle Filter Runs a large number of "guesses" about the system in parallel. This provides an even better characterisation of the estimation distribution.

Extended Kalman Filters

An each time step an extended Kalman filter (EKF) linearises the system around the current estimated state. The linearised model is used to update the estimate error and (sometimes) the state estimate itself.

Typically we use the full equation for $\boldsymbol{f}$ to propagate the state estimate.

However, we need to use a linearised model to propagate the uncertainty in our estimate, because there is no easy way of working out what a distribution will look like when mapped by some arbitrary function.

We will denote the Jacobian of the function $\boldsymbol{f}$ with respect to $\boldsymbol{x}$ at the operating point $\hat{\boldsymbol{x}}(t-1|t-1)$ as

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\big(\hat{\boldsymbol{x}}(t-1|t-1)\big)$$

The Jacobian

Recall that the Jacobian gives us the hyperplane that is tangent to our function $\boldsymbol{f}$ at some chosen point $\boldsymbol{x}_0$.

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}_0) := \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{x}_0) & \frac{\partial f_1}{\partial x_2}(\boldsymbol{x}_0) & \cdots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{x}_0) \\ \frac{\partial f_2}{\partial x_1}(\boldsymbol{x}_0) & \frac{\partial f_2}{\partial x_2}(\boldsymbol{x}_0) & \cdots & \frac{\partial f_2}{\partial x_n}(\boldsymbol{x}_0) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\boldsymbol{x}_0) & \frac{\partial f_n}{\partial x_2}(\boldsymbol{x}_0) & \cdots & \frac{\partial f_n}{\partial x_n}(\boldsymbol{x}_0) \end{bmatrix}$$

In our case, we believe that we are at a point $\hat{\boldsymbol{x}}(t-1|t-1)$, so we form a linear model around that point. Of course, we are almost certainly *not* at $\hat{\boldsymbol{x}}(t-1|t-1)$, but if the nonlinearity is mild then finding the Jacobian at the slightly wrong location should not yield a model that is dramatically different.

Extended Kalman Filters – General Form

Model Formation:

$$\begin{aligned}
\boldsymbol{A}(t-1) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Big(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}(t-1)\Big) \\
\boldsymbol{G}(t-1) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}\Big(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}(t-1)\Big) \\
\boldsymbol{C}(t) &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\Big(\hat{\boldsymbol{x}}(t|t-1)\Big) \\
\boldsymbol{H}(t) &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{v}}\Big(\hat{\boldsymbol{x}}(t|t-1)\Big)
\end{aligned}$$

Prediction:

$$\hat{\boldsymbol{x}}(t|t-1) = \boldsymbol{f}\Big(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}(t-1), 0\Big) \tag{1}$$

$$\boldsymbol{P}(t|t-1) = \boldsymbol{A}\boldsymbol{P}(t-1|t-1)\boldsymbol{A}^{\mathsf{T}} + \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^{\mathsf{T}} \tag{2}$$

Correction:

$$\boldsymbol{L}(t) = \boldsymbol{P}(t|t-1)\boldsymbol{C}^{\mathsf{T}}\left[\boldsymbol{C}\boldsymbol{P}(t|t-1)\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{H}\boldsymbol{R}\boldsymbol{H}^{\mathsf{T}}\right]^{-1} \tag{3}$$

$$\hat{\boldsymbol{x}}(t|t) = \hat{\boldsymbol{x}}(t|t-1) + \boldsymbol{L}\Big(\boldsymbol{y}(t) - \boldsymbol{C}\hat{\boldsymbol{x}}(t|t-1)\Big)$$

$$\boldsymbol{P}(t|t) = (\boldsymbol{I} - \boldsymbol{L}\boldsymbol{C})\,\boldsymbol{P}(t|t-1)$$

EKF Notes

Note that we could use a linearised model to do the prediction step at (1).

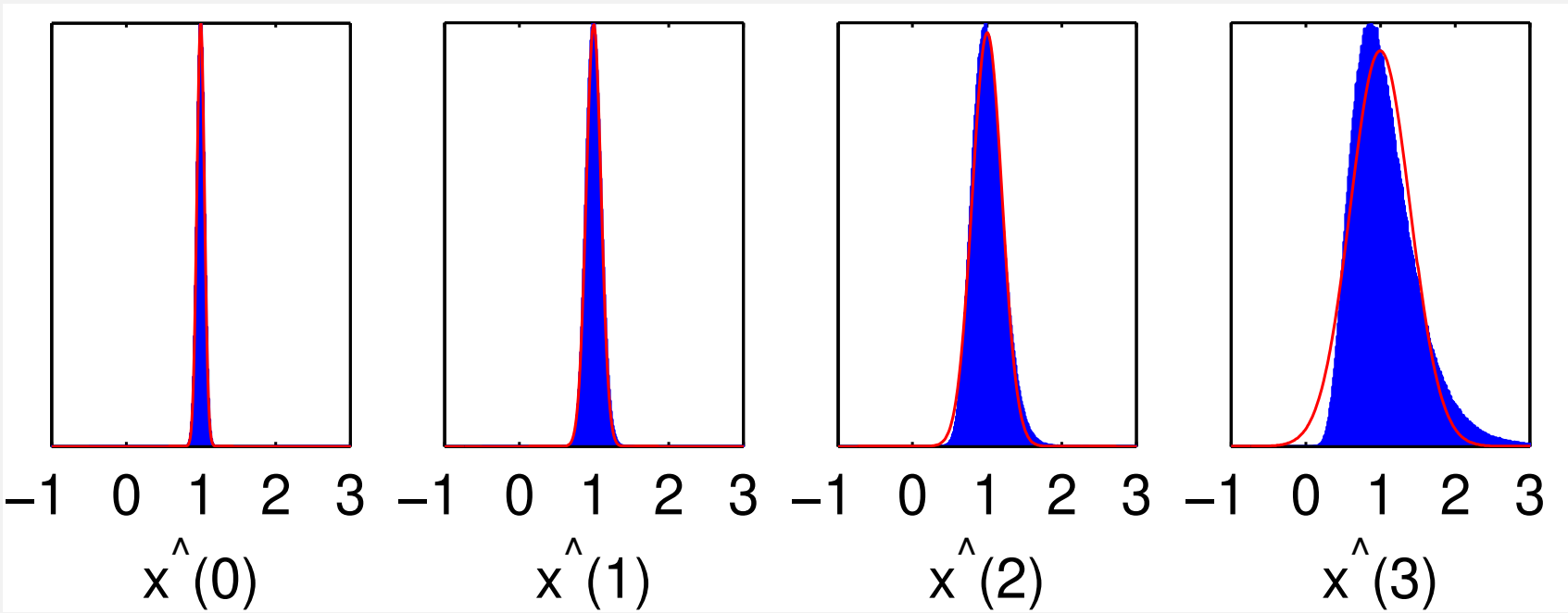
- In general we would need to calculate a linearised $\boldsymbol{B}$ at the operating point, in addition to the other linearisations.
- You would consider this only if calculating $\boldsymbol{f}$ is slow compared to doing the additional linearisation *and* if the resulting accuracy is adequate.

If noise is additive then we can extract it from $\boldsymbol{f}$ and/or $\boldsymbol{h}$, which makes calculation of linearised models of the noise effects unnecessary and simplifies the other equations.

- If $\boldsymbol{x}(t) = f\big(\boldsymbol{x}(t-1), \boldsymbol{u}(t-1)\big) + \boldsymbol{w}(t-1)$, then we don't need to calculate $\boldsymbol{G}$ and we use $\boldsymbol{Q}$ in (2).
- If $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{v}(t)$, then we don't need to calculate $\boldsymbol{H}$ and we just use $\boldsymbol{R}$ in (3).

EKF example

If we propagate a Gaussian distribution through some nonlinear function then the output may not be Gaussian. Its mean may also not be that predicted by mapping the prior mean with f.



The example shows an initial $\hat{x} \sim \mathcal{N}\left(1, \frac{1}{400}\right)$ propagated through $f = x^2$. The red gaussian arises from a linear mapping using $\boldsymbol{A} = \frac{\mathrm{d}f}{\mathrm{d}x} = 2x$ to manage $\boldsymbol{P}$. $\boldsymbol{Q} = 0$ for this example.

Extended Kalman Filter Notes

Extended Kalman filters *tend* to underestimate $\boldsymbol{P}$, because most nonlinear functions spread a distribution more than their linear approximations. Because the EKF is iterative, the resulting inaccuracy in $\boldsymbol{P}$ grows with time.

Underestimating $\boldsymbol{P}$ is bad, as it makes the estimator believe predictions more than measurements. If we have a questionable estimator, we would usually prefer the opposite behaviour.

The linearisation for the EKF is performed about the current estimate. If the estimate is poor, then the linearisation can be a very bad approximation to the underlying nonlinear function, leading to rapid divergence of the estimator from the true state.

It can be critical to have good initial guess in an extended Kalman filter, otherwise it may never converge on a correct estimate.

Iterated Extended Kalman Filter

One of the problems with the Extended Kalman filter is that the linearisation is taken around the prior estimated state. If that prior is a long way from the true state, then it may be better to relinearise around the posterior to improve the model.

That is, we use the prior to propagate the covariance as in the EKF and then update our estimate at the new state. This update may in turn change the estimated state significantly, so we might want to relinearise again. That is, we *iterate* the linearisation process until the answer converges.

The Unscented Kalman Filter

Both the linear and extended Kalman filters assume that we can capture the statistics of our estimate with only the mean and variance. This is because the mean and variance are sufficient statistics for a gaussian. However, in the unscented Kalman filter, we seek to do better by using a *set* of states to characterise our estimates.

We choose a set of points (called *sigma points*) that describe $(\hat{\boldsymbol{x}}, \boldsymbol{P})$. In particular we want the set of sigma points to have the same mean and variance as our current estimate. Each of the sigma points is then propagated through the model $\boldsymbol{f}$. Finally, we estimate the distribution after the mapping by taking the mean and variance of the mappings of the set of sigma points.

In an EKF we attempt to propagate a known distribution through an approximation of a function. In an UKF we attempt to propagate an approximation of a distribution through a known function.

Most of the time the UKF approach is likely to be more accurate, but it tends to be slower.

Choosing the Sigma points

There are a number of sensible strategies for choosing the sigma points. One intuitively appealing approach for estimating a system in $\mathbb{R}^n$ is to choose $2n$ points.

$$\begin{aligned} \boldsymbol{x}^{(i)} &= \hat{\boldsymbol{x}}(t-1|t-1) + \tilde{\boldsymbol{x}}^{(i)} \quad i \in [1..2n] \\ \tilde{\boldsymbol{x}}^{(i)} &= \left[\sqrt{n\boldsymbol{P}}\right]_i \quad i \in [1..n] \\ \tilde{\boldsymbol{x}}^{(n+i)} &= \left[\sqrt{n\boldsymbol{P}}\right]_i \quad i \in [1..n] \end{aligned}$$

To find the $\tilde{\boldsymbol{x}}$ we calculate $n\boldsymbol{P}$, find a matrix sqaure root (using a Cholesky factorisaton for example) and then extract the i^{th} column. Note that $\tilde{\boldsymbol{x}}$ here is *not* the error in our estimate, as was previously denoted by this notation.

In the case where $\boldsymbol{P}$ is diagonal this reduces to placing a sigma point one standard deviation from the mean in each dimension (hence the name sigma points).

Operation of the UKF

We then propagate the $2n$ points using $\boldsymbol{f}$ and calculate prior estimates for the mean and variance of the resulting distribution.

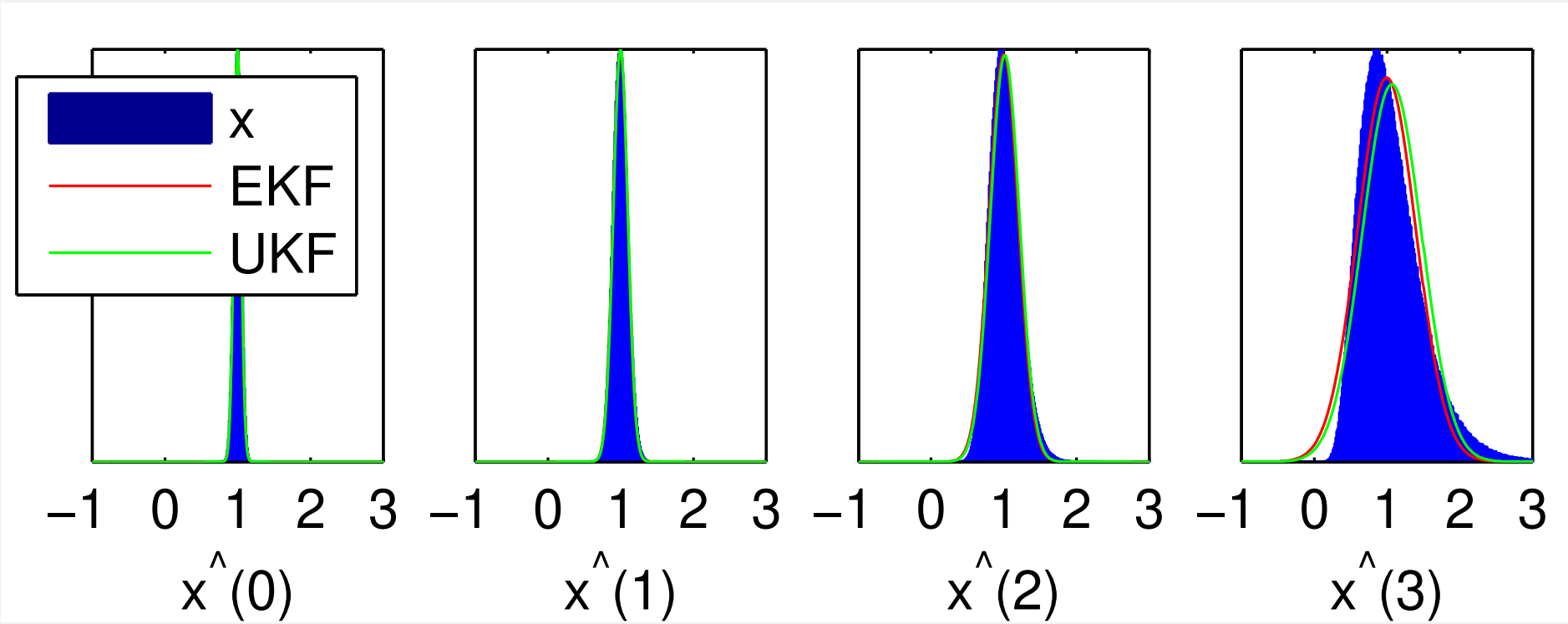
$$\boldsymbol{x}^{(i)}(t|t-1) = \boldsymbol{f}(\hat{\boldsymbol{x}}(t-1|t-1)^{(i)}, u(t))$$

$$\hat{\boldsymbol{x}}(t|t-1) = \frac{1}{2n}\sum_{i=1}^{2n} \boldsymbol{x}^{(i)}(t|t-1)$$

$$\boldsymbol{P}(t|t-1) = \boldsymbol{Q} + \frac{1}{2n}\sum_{i=1}^{2n}\left(\boldsymbol{x}^{(i)}(t|t-1) - \hat{\boldsymbol{x}}(t|t-1)\right) \times \left(\boldsymbol{x}^{(i)}(t|t-1) - \hat{\boldsymbol{x}}(t|t-1)\right)^{\mathrm{T}}$$

That is, we form our prior estimate by finding the mean and sample variance of the sigma points.

EKF, UKF Comparison



The unscented Kalman filter typically yields a better estimate than the extended Kalman filter. For this example, the true mean of the distribution is 1.0709 and the UKF estimates 1.0704. By comparison the EKF gives an estimate of the mean as 1.0.

The EKF gives a variance of 1.6, whereas that of the UKF is 1.66.

Operation of the UKF

We must then perform the correction step of the Kalman filter. Strictly speaking we should find new sigma points to describe the prior estimate. However, we can often cheat and reuse the same points as before, as long as the nonlinearity in f is not too severe.

We begin the correction phase by making measurement predictions based on each of the sigma points. We then combine them to find our overall prediction for $\boldsymbol{y}$, remembering that we need to add $\boldsymbol{R}$ to the measurement uncertainty.

$$\hat{\boldsymbol{y}}^{(i)} = \boldsymbol{h}(\boldsymbol{x}^{(i)})$$

$$\hat{\boldsymbol{y}} = \frac{1}{2n}\sum_{i=1}^{2n} \hat{\boldsymbol{y}}^{(i)}$$

$$\boldsymbol{P}_y = \boldsymbol{R} + \frac{1}{2n}\sum_{i=1}^{2n} \left(\hat{\boldsymbol{y}}^{(i)} - \hat{\boldsymbol{y}}\right)\left(\hat{\boldsymbol{y}}^{(i)} - \hat{\boldsymbol{y}}\right)^{\mathsf{T}}$$

We can then finish the correction using the normal Kalman filter equations.

Using $2n+1$ sigma points

Slightly better estimation performance can be gained by using an extra sigma point at the current estimate and tweaking the position of the other points slightly. The constant γ can be adjusted to improve higher order errors in the estimate.

$$
\begin{aligned}
x^{(0)} &= \hat{\boldsymbol{x}}(t-1|t-1) \\
x^{(i)} &= \hat{\boldsymbol{x}}(t-1|t-1) + \tilde{\boldsymbol{x}}^{(i)} \quad i \in [1..2n] \\
\tilde{\boldsymbol{x}}^{(i)} &= \left[\sqrt{(n+\gamma)\boldsymbol{P}}\right]_i \quad i \in [1..n] \\
\tilde{\boldsymbol{x}}^{(n+i)} &= \left[\sqrt{(n+\gamma)\boldsymbol{P}}\right]_i \quad i \in [1..n] \\
w^{(0)} &= \frac{\gamma}{n+\gamma} \\
w^{(i)} &= \frac{1}{2(n+\gamma)} \quad i \in [1..2n]
\end{aligned}
$$

Then, $\bar{y} = \sum_{i=0}^{2n} w^{(i)} y^{(i)}$ and $\boldsymbol{P}_y = \sum_{i=0}^{2n} w^{(i)} \left(y^{(i)} - \bar{y}\right)\left(y^{(i)} - \bar{y}\right)^{\mathsf{T}}$

Other sensible choices of sigma points

If using $2n$ or $2n+1$ sigma points is too computationally intensive, then it is possible to use a set of sigma points arranged in a simplex around the mean. This essentially is the smallest number of points capable of capturing the first two moments (the mean and variance) of the underlying distribution.

The simplex sigma points algorithm operting in n dimensions requires only $n+1$ sigma points.

That approach does have numerical stability problems, which can be overcome by using the spherical simplex algorithm, which perturbs the simplex positions to stabilise the calculations.

The Particle Filter

The unscented Kalman filter can cope with more extreme nonlinearities than the extended Kalman filter, but it still fails when the the nonlinearirty become too severe. It fails, for example, when presented with a multimodal noise source.

The basic premise of the unscented Kalman filter was that we could represent the distribution in our estimate using a number of samples and then propagate those samples through the nonlinear system. One might wonder what would happen if we increase the number of such samples.

This is the approach taken by the *particle filter*, where the role of the sigma points us taken over by independent particles.

The Particle Filter

This is in fact the approach that has been taken in the examples above to generate the "true" distributions produced by the mapping $f : x \mapsto x^2$. In those cases 1×10^6 particles were used to make the graphs smooth; this is too extreme in practice, as it would take far too long to calculate each update. Hundreds to thousands of particles might be used in a real problem.

It should be apparent that the particle filter will be extremely computationally demanding. Good performance requires a lot of particles (particularly in a higher dimensional system) to adequately represent the estimate uncertainty. Each of those particles will take about as long to update as a simple Kalman filter.

The details of the particle filter are more easily handled with the machinery of Bayesian statistics, so we will take an overview to see the analogy with our preceding methods.

The Particle Filter

The first step of the particle filter algorithm is to propagate each particle's previous posterior.

$$\hat{\boldsymbol{x}}^{(i)}(t|t-1) = f\left(\hat{\boldsymbol{x}}(t-1|t-1), \boldsymbol{u}, \boldsymbol{w}^{(i)}\right) \quad i \in [1..n]$$

Note that $\boldsymbol{w}^{(i)}$ in the above equation is a particular sample drawn from the process noise distribution (which need not be Gaussian). That is, each particle has a "guess" at what the noise might be.

An observation is then taken and a weight is determined for each particle. The weight describes how likely the observation (y) is given the particle's estimate of the state.

$$w^{(i)} = \mathrm{P}\left\{ y|x^{(i)} \right\}$$

Calculating the weight requires knowledge of the system's underlying nonlinear dynamics and of the measurement noise.

The Particle Filter

We then normalise the various likelihoods so that the sum of the likelihoods is one.

$$w^{(i)} \mapsto \frac{w^{(i)}}{\sum_{i=1}^{n} w^{(i)}}$$

This step will generally be slow.

The set of particle estimates $\hat{\boldsymbol{x}}^{(i)}$ along with their weights $w^{(i)}$ describe our belief in the true state. Given this description we can calculate any statistic of the particle population that we like. For example, if we wanted to know the "best" estimate of the system state we might calculate the weighted mean of the particle estimates.

Resampling

One practical problem is that many particles tend to wander off from the true state of the system. This is because they make incorrect guesses about the process noise. In a running particle filter one finds that the weights of many particles very quickly go to zero.

Particles with zero weight are of no use, so we resample the particles to move them back into a useful part of the estimated distribution.

There are many alternate strategies used to perform the resampling. As a couple of examples,

- We could resample all n particles from the estimated distribution.
- We could take particles with low weight and duplicate a high weight particle (while distributing the weight between the two).

This is an active area of research and as yet there is no consensus on the right approach.