Module 2: State Estimation - Linear and Nonlinear Kalman Filters

In today's world of advanced machine learning, the Kalman filter remains an important tool to fuse measurements from several sensors to estimate in real-time the state of a robot system such as a self-driving car.

内容

1. A brief history and overview of the (linear) Kalman filter
2. Kalman filter as the BLUE (best Linear Unbiased Estimator)
3. Extending the Kalman filter to nonlinear systems through linearization.
4. Limitations of linearization
5. Unscented Kalman filter
   - a modern alternative to linearization through the unscented transform.

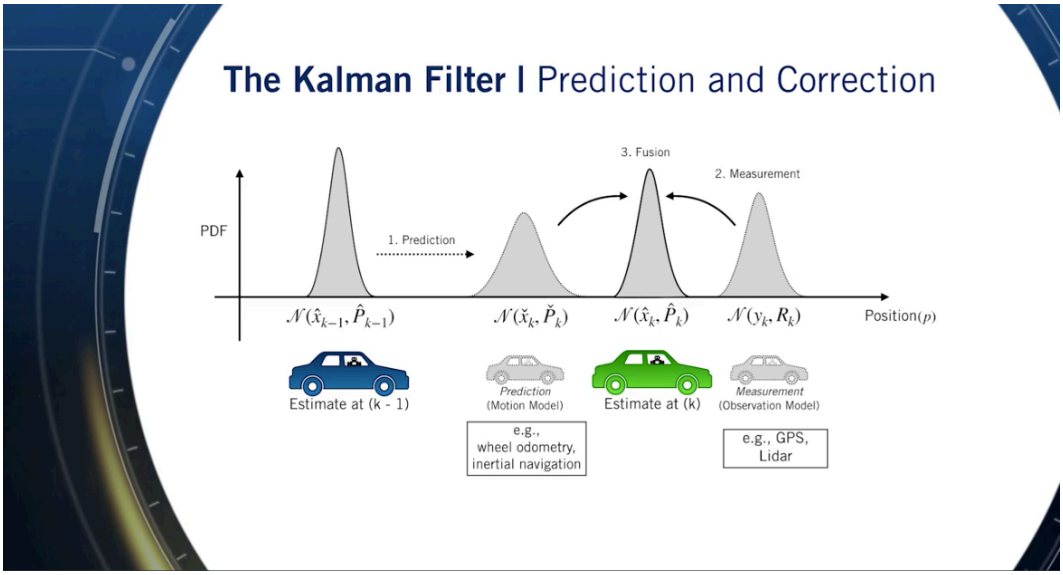
Lesson 1: The (Linear) Kalman Filter

内容

- Describe the Kalman filter as a two state filter: (1) prediction and (2) correction
- Understand the difference between motion and measurement models.
- Use the Kalman filter in a simple 1D localization example.

The Kalman Filter

- The (extended) Kalman Filter became widely known after its use in the Apollo Guidance Computer for circumlunar navigation.



The Kalman Filter | Prediction and Correction

- The Kalman filter is very similar to the linear recursive least squares filter.
  - While recursive least squares updates the estimate of a static parameter, Kalman filter is able to update the estimate of an evolving state.
- 1D Example（大事）

1. Starting from an initial probabilistic estimate at time $k-1$, the goal is to use a motion model which could be derived from wheel odometry or inertial sensor measurements to predict our new state.
2. Use the observation model derived from GPS for example, to correct that prediction of vehicle position at time k.
3. Each of these components, the initial estimate, the predicted state, and the final corrected state are all random variables that will be specified by their means and covariances.
    1. In this way, Kalman filter can be seen as a technique to fuse information from different sensors to produce a final estimate of some unknown state, taking into account uncertainty in motion and in measurements.

The Kalman Filter | Linear Dynamical System

- The Kalman Filter requires the following motion and measurement models:
  - Motion model: $x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1}$
    - u: control input
      - an external signal that affects the evolution of system state.
      - In the context of self-driving vehicles, this may be a wheel torque applied to speed up and change lanes, for example.
    - w: zero-mean noise
  - Measurement model: $y_k = H_k x_k + v_k$
- With the following noise properties
  - Measurement noise: $v_k \sim \mathcal{N}(0, R_k)$
  - Process or Motion noise: $w_k \sim \mathcal{N}(0, Q_k)$

The Kalman Filter | Recursive Least Squares + Process Model

まずは変数の意味を整理しましょう。

- R: measurement noise v_kのvariance. $v_k \sim \mathcal{N}(0, R_k)$
- Q: process or motion noise w_kのvariance. $w_k \sim \mathcal{N}(0, Q_k)$
- P: estimatorのcovariance. $\mathcal{L}_{RLS} = \mathbb{E}[(x_{1k} - \hat{x}_{1k})^2 + \ldots + (x_{nk} - \hat{x}_{nk})^2] = Trace(P_k)$

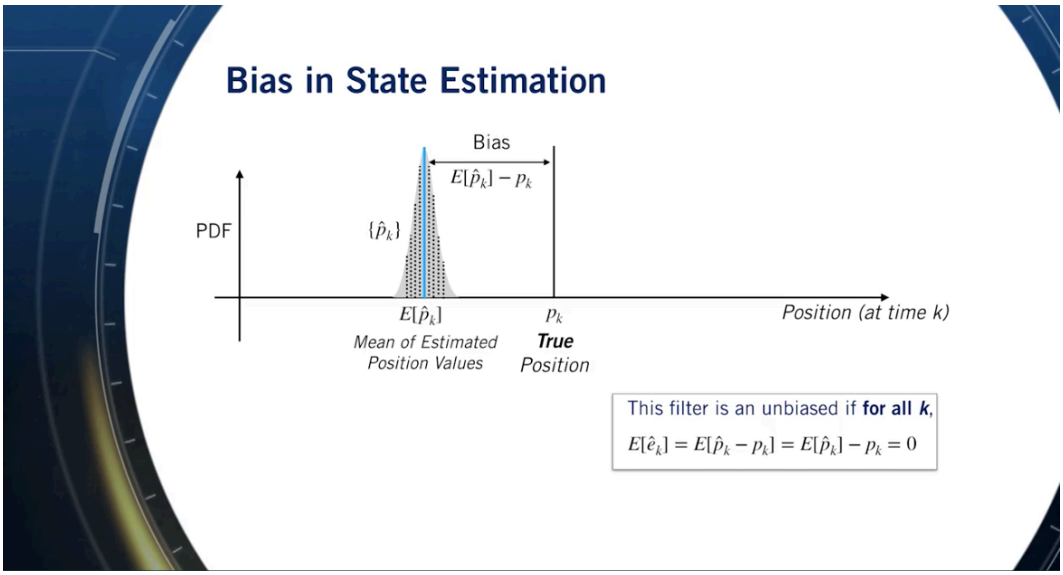
The Kalman filter is a recursive least squares estimator that also includes a motion model.

1. Prediction: Use the process model (motion model) to predict how states (evolving states and non-state parameters) evolved since the last time step. Propagate mean and uncertainty (covariance).
$$\check{x}_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1}$$
$$\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + Q_{k-1}$$
2. Optimal Gain: $K_k = \check{P}_k H_k^T (H_k \check{P}_k H_k^T + R_k)^{-1}$
    1. 因みにrecursive least squaresのoptimal gainの計算式は:
    $$K_k = P_{k-1}H_k^T(H_k P_{k-1} H_k^T + R_k)^{-1}$$
    $$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1})$$
    $$P_k = (1 - K_k H_k)P_{k-1}$$
    , ほとんど一緒、ただP_{k-1}を使わず、$\check{P}_k$を使う。
3. Correction: Use optimal gain to propagate the state covariance from prediction to corrected state.
$$\hat{x}_k = \check{x}_k + K_k(y_k - H_k\check{x}_k)$$
$$\hat{P}_k = (1 - K_k H_k)\check{P}_k$$
, 上記のrecursive least squaresの更新式と比べると$\hat{x}_{k-1}$を使わず、$\check{x}_k$を使う。P_{k-1}を使わず、$\check{P}_k$を使う。つまり全周期の予測値ではなく、現在周期のMotion Modelで予測した値を使う。
4. Prediction: $\check{x}_k$ (given motion model) at time k.
5. Corrected prediction: $\hat{x}_k$ (given measurement) at time k.

The Kalman Filter | Short Example

1. p: Position, $x = \begin{bmatrix} p \\ \frac{dp}{dt} = \dot{p} \end{bmatrix}$, $u = a = \frac{d^2p}{dt^2}$
2. Motion/Process Model: $x_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} u_{k-1} + w_{k-1}$
3. Position Observation: $y_k = [1 \quad 0]\, x_k + v_k$
4. Noise Densities: $v_k \sim \mathcal{N}(0,0.05)$, $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, (0.1)\mathbf{1}_{2\times 2})$
5. $\hat{x}_0 \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \right)$, $\Delta t = 0.5s$, $u_0 = -2m/s^2$, $y_1 = 2.2m$
6. Prediction:
   1. $\check{x}_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1}$, $\begin{bmatrix} \check{p}_1 \\ \check{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$
   2. $\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + Q_{k-1}$, $\check{P}_1 = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}^T + \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$
7. Correction:
   1. $K_k = \check{P}_k H_k^T (H_k \check{P}_k H_k^T + R_k)^{-1}$, $K_1 = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \left([1 \quad 0] \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.05 \right)^{-1} = \begin{bmatrix} 0.88 \\ 1.22 \end{bmatrix}$
   2. $\hat{x}_k = \check{x}_k + K_k(y_k - H_k\check{x}_k)$, $\hat{x}_1 = \begin{bmatrix} \hat{p}_1 \\ \hat{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \begin{bmatrix} 0.88 \\ 1.22 \end{bmatrix} \left(2.2 - [1 \quad 0] \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} \right) = \begin{bmatrix} 2.24 \\ 3.63 \end{bmatrix}$
   3. $\hat{P}_k = (1 - K_k H_k)\check{P}_k$, $\hat{P}_1 = \begin{bmatrix} 0.04 & 0.06 \\ 0.06 & 0.49 \end{bmatrix}$
      1. corrected state variance is smaller. This uncertainty reduction occurs because measurement model is fairly accurate. つまりv_kのvariance (0.05)が小さい。

- The Kalman Filter is very similar to RLS but includes a motion model that tells us how the state evolves over time.
- The Kalman Filter updates a state estimate through two stages:
  - prediction using the motion model
  - correction using the measurement model



Lesson 2: Kalman Filter and The Bias BLUEs

内容

- Define bias
- Define consistency
- Explain why the Kalman filter is the Best Linear Unbiased Estimator (BLUE)

Bias in State Estimation

- Drive the car for K time steps, record estimation error, repeat…
- An estimator or filter is unbiased if it produces an 'average' error of zero at a particular time step k, over many trials.
  - Build a histogram of the positions that the filter reports over multiple trials.
  - Compute the difference between the average of these estimates and the true position.
  - If this difference does approach zero as repeating this experiment many more times, and if this happens for all time intervals, then the filter is unbiased.
- Consider the error dynamics:
  - Predicted state error: $\check{e}_k = \check{x}_k - x_k$
  - Corrected estimate error: $\hat{e}_k = \hat{x}_k - x_k$
- Using the Kalman Filter equations, can derive:（証明がない、自分で調べよう、証明しよう）
  - $\check{e}_x = F_{k-1}\check{e}_{k-1} - w_k$
  - $\hat{e}_k = (1 - K_k H_k)\check{e}_k + K_k v_k$
- For the Kalman filter, for all k, $E[\check{e}_x] = E[F_{k-1}\check{e}_{k-1} - w_k] = F_{k-1}E[\check{e}_{k-1}] - E[w_k] = 0$, $E[\hat{e}_k] = E[(1 - K_k H_k)\check{e}_k + K_k v_k] = (1 - K_k H_k)E[\check{e}_k] + K_k E[v_k] = 0$
  - So long as $E[\hat{e}_0] = 0$, $E[v] = 0$, $E[w] = 0$ + white, uncorrelated noise. この条件だったら、証明は簡単。
  - This does not mean that the error on a given trial will be zero, but that, with enough trials, our expected error is zero.

Consistency in State Estimation

- This filter is consistent if for all k, $E[\hat{e}_k^2] = E[(\hat{p}_k - p_k)^2] = \hat{P}_k$（証明なし…）
- Practically, this means that our filter is neither overconfident, nor under confident in the estimate it has produced.
  - overconfident filter: essentially place too much emphasis on its own estimate and be less sensitive to future measurement updates.
  - easy to see how an overconfident filter might have a negative or dangerous effect on the performance of self-driving car.
- One can also show that for all k, $E[\check{e}_k\check{e}_k^T] = \check{P}_k$, $E[\hat{e}_k\hat{e}_k^T] = \hat{P}_k$（証明なし…）
  - Provided, $E[\check{e}_0\check{e}_0^T] = \check{P}_0$, $E[v] = 0$, $E[w] = 0$ + white noise

The Kalman Filter is the BLUE (Best Linear Unbiased Estimator)

- Given linear formulation, and zero-mean, white noise, the Kalman Filter is unbiased and consistent: $E[\hat{e}_k] = 0$, $E[\hat{e}_k\hat{e}_k^T] = \hat{P}_k$.
- In general, if white, uncorrelated zero-mean noise, the Kalman filter is the best (i.e., lowest variance（これも証明なしだろう…）) unbiased estimator that uses only a linear combination of measurements.
- However, most real systems are not linear.
- For self-driving cars, generally need to estimate non-linear quantities like vehicle poses, position, and orientation in 2D and 3D.

Lesson 3: Going Nonlinear - The Extended Kalman Filter

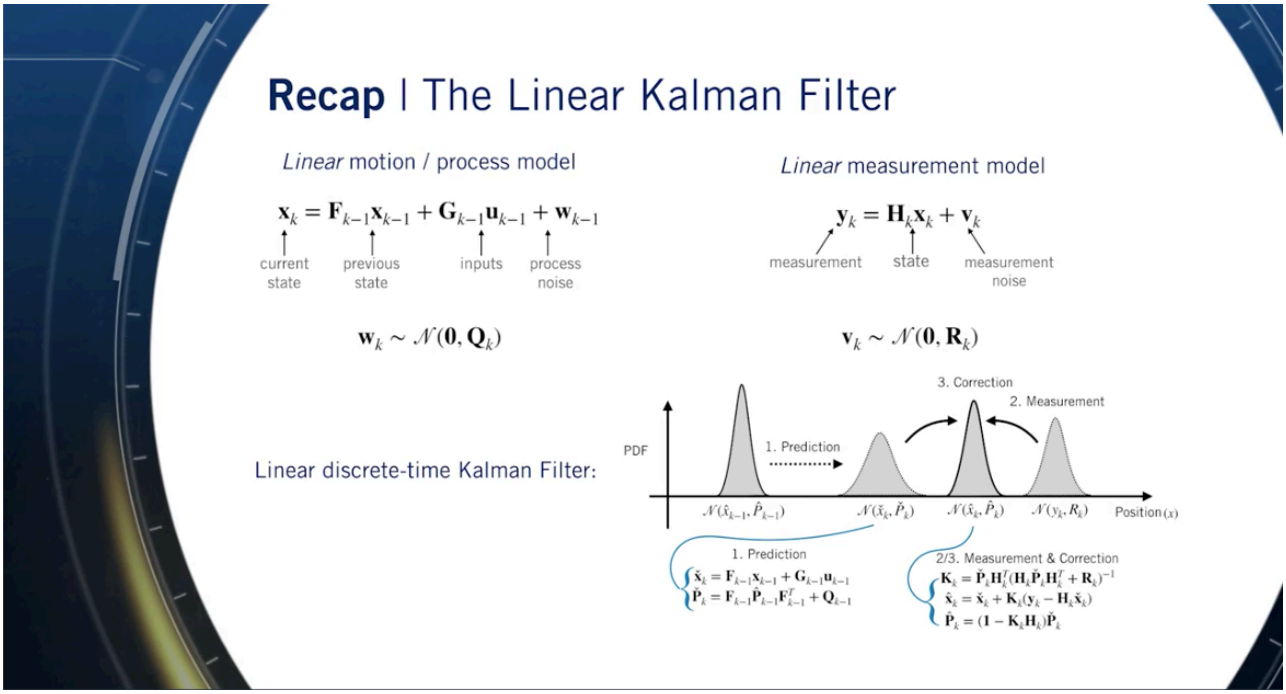
単語

- Identity matrix: In linear algebra, the identity matrix, or sometimes ambiguously called a unit matrix, of size n is the $n \times n$ square matrix with ones on the main diagonal and zeros elsewhere.
- Additive white Gaussian noise (AWGN): a basic noise model used in Information theory to mimic the effect of many random processes that occur in nature.
  - Additive, it is added to any noise that might be intrinsic to the information system.
  - White, it has uniform to power across the frequency band for the information system. It is an analogy to the color white which has uniform emissions at all frequencies in the visible spectrum.
  - Gaussian, it has a normal distribution in the time domain with an average time domain value of zero.

車両予測に全然EKF使ってないけど、良いの?

内容

- Describe how the EKF uses first-order linearization to turn a nonlinear problem into a linear one.
- Understand the role of Jacobian matrices in the EKF and how to compute them.
- Apply the EKF to a simple nonlinear tracking problem.



Recap | The Linear Kalman Filter

- A predictor corrector architecture
- Linear systems do not exist in reality.
- How to adapt the Kalman Filter to nonlinear discrete-time systems?

EKF | Linearizing a Nonlinear System

- Choose an operating point a and approximate the nonlinear function by a tangent line at that point.

- Mathematically, compute this linear approximation using a first-order Taylor expansion: （ディープラーニング勉強中既に分かっていた）

  $f(x) \approx f(a) + \frac{\partial f(x)}{\partial x}|_{x=a}(x-a) + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2}|_{x=a}(x-a)^2 + \ldots$（スライドに書いたテイラー展開は間違っている、３階導関数のところ）
- For the EKF, choose the operating point to be the most recent state estimate, known input, and zero noise. 前回true valueではなく、前回のstate estimateをoperating pointとして使う!
- Linearized motion model:

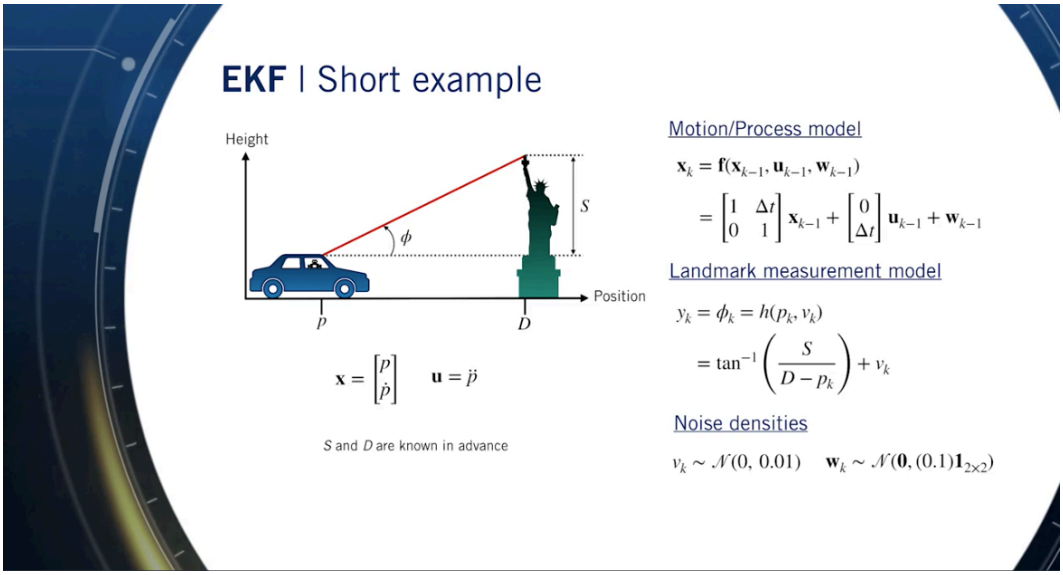
  $$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) \approx f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0) + \underbrace{\frac{\partial f_{k-1}}{\partial x_{k-1}}|_{\hat{x}_{k-1}, u_{k-1}, 0}}_{F_{k-1}}(x_{k-1} - \hat{x}_{k-1}) + \underbrace{\frac{\partial f_{k-1}}{\partial w_{k-1}}|_{\hat{x}_{k-1}, u_{k-1}, 0}}_{L_{k-1}} w_{k-1}$$

  - w_{k-1}が0で展開している。つまりzero noiseをoperating pointとして使っている。
- Linearized measurement model: $y_k = h_k(x_k, v_k) \approx h_k(\check{x}_k, 0) + \underbrace{\frac{\partial h_k}{\partial x_k}|_{\check{x}_k, 0}}_{H_k}(x_k - \check{x}_k) + \underbrace{\frac{\partial h_k}{\partial v_k}|_{\check{x}_k, 0}}_{M_k} v_k$
- Now a linear system in state-space! The matrices F_{k-1}, L_{k-1}, H_k and M_k are called the Jacobian matrices of the system.
  - F_{k-1}とH_kはLinear Kalman Filterには既に使っている。

EKF | Computing Jacobian matrices（ディープラーニング勉強中既に分かっていた）
- In vector calculus, a Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function. $\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$
- Intuitively, the Jacobian matrix tells you how fast each output of the function is changing along each input dimension

EKF | Putting it all together
- With linearized models and Jacobians, can now use the Kalman Filter equations
- Linearized motion model: $x_k = f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0) + F_{k-1}(x_{k-1} - \hat{x}_{k-1}) + L_{k-1} w_{k-1}$
- Prediction:
  $$\check{x}_k = f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0)$$
  $$\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + L_{k-1}Q_{k-1}L_{k-1}^T$$
- Linearized measurement model: $y_k = h_k(\check{x}_k, 0) + H_k(x_k - \check{x}_k) + M_k v_k$
- Optimal gain: $K_k = \check{P}_k H_k^T (H_k \check{P}_k H_k^T + M_k R_k M_k^T)^{-1}$
- Correction:
  $$\hat{x}_k = \check{x}_k + K_k(y_k - h_k(\check{x}_k, 0))$$
  $$\hat{P}_k = (1 - K_k H_k)\check{P}_k$$
- Linear Kalman Filterの式との区別
  - First, in the prediction and correction steps, still use the nonlinear models to propagate the mean of the state estimate and to compute the measurement residual or innovation.
    - 理由：linearize motion model about the previous state estimate, linearize the measurement model about the predicted state.
    - By definition, the linearized model exactly coincides with the nonlinear model at the operating point.
  - Second, the appearance of L and M Jacobians related to the process and measurement noise.

- In many cases, both of these matrices will be identity since noise is often assumed to be additive.



EKF | Short Example

- Motion modelはまだlinear、Measurement modelはnonlinear。
- Motion model Jacobians: $F_{k-1} = \frac{\partial f_{k-1}}{\partial x_{k-1}}|_{\hat{x}_{k-1}, u_{k-1}, 0} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$,

  $L_{k-1} = \frac{\partial f_{k-1}}{\partial w_{k-1}}|_{\hat{x}_{k-1}, u_{k-1}, 0} = 1_{2\times 2}$
- Measurement model Jacobians: $H_k = \frac{\partial h_k}{\partial x_k}|_{\check{x}_k, 0} = \begin{bmatrix} \frac{S}{(D - \check{p}_k)^2 + S^2} & 0 \end{bmatrix}$, $M_k = \frac{\partial h_k}{\partial v_k}|_{\check{x}_k, 0} = 1$
- Data: $\hat{x}_0 \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}\right)$, $\Delta t = 0.5s$, $u_0 = -2m/s^2$, $y_1 = \pi/6 rad$, $S = 20m$,

  $D = 40m$, Using the Extended Kalman Filter equations, calculate updated position $\hat{p}_1$.
- EKFのPredictionの式：
  $$\check{x}_k = f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0)$$
  $$\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + L_{k-1}Q_{k-1}L_{k-1}^T$$
  - $\check{x}_k = f_0(\hat{x}_0, u_0, 0)$, $\begin{bmatrix} \check{p}_1 \\ \check{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}(-2) = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$ （この計算はLinear Kalman Filterと同じ）
  - $\check{P}_1 = F_0\hat{P}_0F_0^T + L_0Q_0L_0^T$,

    $\check{P}_1 = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}^T + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$ （計算結果はLinear Kalman Filterと同じ、LがIdentityだから）
  $$K_k = \check{P}_kH_k^T(H_k\check{P}_kH_k^T + M_kR_kM_k^T)^{-1}$$
- EKFのCorrection式：
  $$\hat{x}_k = \check{x}_k + K_k(y_k - h_k(\check{x}_k, 0))$$
  $$\hat{P}_k = (1 - K_kH_k)\check{P}_k$$

- $K_1 = \check{P}_1 H_1^T (H_1 \check{P}_1 H_1^T + M_1 R_1 M_1^T)^{-1} = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 0.011 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 0.011 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 0.011 \\ 0 \end{bmatrix} + 1(0.01)(1) \right)^{-1} = \begin{bmatrix} 0.40 \\ 0.55 \end{bmatrix}$
- $\hat{x}_1 = \check{x}_1 + K_1(y_1 - h_1(\check{x}_1, 0)), \hat{x}_1 = \begin{bmatrix} \hat{p}_1 \\ \hat{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \begin{bmatrix} 0.40 \\ 0.55 \end{bmatrix} (0.52 - 0.49) = \begin{bmatrix} 2.51 \\ 4.02 \end{bmatrix}$
- $\hat{P}_1 = (1 - K_1 H_1) \check{P}_1 = \left(1 - \begin{bmatrix} 0.40 \\ 0.55 \end{bmatrix} \begin{bmatrix} 0.011 & 0 \end{bmatrix} \right) \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$

  $= \left(1 - \begin{bmatrix} 0.0044 & 0 \\ 0.00605 & 0 \end{bmatrix} \right) \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} = \begin{bmatrix} 0.36 & 0.50 \\ 0.50 & 1.1 \end{bmatrix}$ (変わらないけど…)
  - In this case, even though the corrected mean of the state estimate ($\begin{bmatrix} 2.51 \\ 4.02 \end{bmatrix}$) is different from the predicted mean ($\begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$), the corrected covariance didn't change that much from the predicted covariance.
    - 理由：This is because the azimuth angle changes slowly at this distance, and doesn't provide much information about the vehicle state compared to a GPS measurement.

- The EKF uses linearization to adapt the Kalman Filter to nonlinear systems.
- Linearization works by computing a local linear approximation to a nonlinear function about a chosen operating point.
- Linearization relies on computing Jacobian matrices, which contain all the first-order partial derivatives of a function.

Lesson 4: An Improved EKF - The Error State Extended Kalman Filter

内容
- Describe the error-state formulation of the Extended Kalman Filter.
- Describe the advantages of the Error-state EKF over the vanilla EKF.

What's in a State?
- Think of the vehicle state as composed of two parts: $x = \hat{x} + \delta x$
  - x: True state
  - $\hat{x}$: Nominal state ("Large")
  - δx: Error state ("Small")
- Continuously update the nominal state by integrating the motion model.
- Modeling errors and process noise accumulate into the error state.

ES-EKFの考え方
The Error-State Extended Kalman Filter estimates the error state directly and uses it as a correction to the nominal state.
- 元のLinearized motion model: $x_k = f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0) + F_{k-1}(x_{k-1} - \hat{x}_{k-1}) + L_{k-1} w_{k-1}$
- Error state kinematicsに変換：$\underbrace{x_k - f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0)}_{\delta x_k} = F_{k-1} \underbrace{(x_{k-1} - \hat{x}_{k-1})}_{\delta x_{k-1}} + L_{k-1} w_{k-1}$
- 元のLinearized measurement modelは変換なくてそのままわかる：
  $y_k = h_k(\check{x}_k, 0) + H_k \underbrace{(x_k - \check{x}_k)}_{\delta x_k} + M_k v_k$

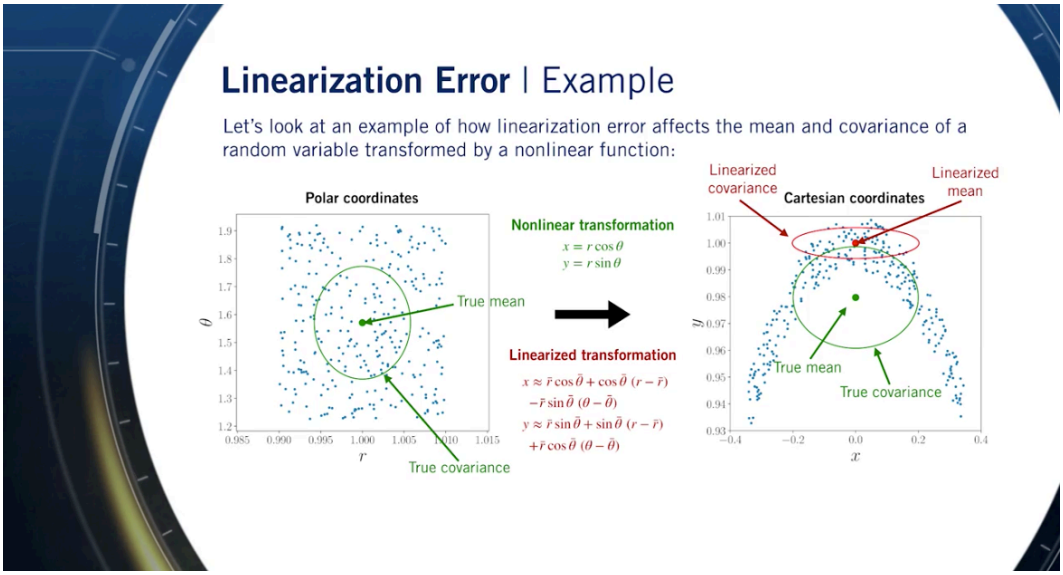
The Error-State Extended Kalman Filter
Loop:

1. Update nominal state with motion model: $\check{x}_k = f_{k-1}(x_{k-1}, u_{k-1}, 0)$
    1. x_{k-1} could be $\check{x}_{k-1}$ or $\hat{x}_{k-1}$, measurementはまだ来ていないかもしれないから。
2. Propagate uncertainty: $\check{P}_k = F_{k-1} P_{k-1} F_{k-1}^T + L_{k-1} Q_{k-1} L_{k-1}^T$
    1. P_{k-1} could be $\check{P}_{k-1}$ or $\hat{P}_{k-1}$, depending on whether we used a measurement to do a correction step.
3. Repeat 1, 2 until a measurement is available and we want to do a correction:
    1. Compute Kalman Gain: $K_k = \check{P}_k H_k^T (H_k \check{P}_k H_k^T + M_k R_k M_k^T)^{-1}$.（スライドには$M$がない、多分Identityだから、EKFと一緒）
    2. Compute error state: $\delta\hat{x}_k = K_k(y_k - h_k(\check{x}_k, 0))$.（$\hat{x}_k = \check{x}_k + K_k(y_k - h_k(\check{x}_k, 0))$だから、つまりEKFと一緒）
    3. Correct nominal state: $\hat{x}_k = \check{x}_k + \delta\hat{x}_k$.
    4. Correct state variance: $\hat{P}_k = (1 - K_k H_k)\check{P}_k$.（EKFと一緒）

Why Use the ES-EKF?

1. Better performance compared to the vanilla EKF: The “small” error state is more amenable to linear filtering than the “large” nominal state, which can be integrated nonlinearly.
2. Easy to work with constrained quantities (e.g., rotations in 3D): Break down the state using a generalized composition operator:（generalized composition operatorで具体的にどこ変わったかはまだ分かっていない）$x = \hat{x} \oplus \delta x$
    - x: True State
    - $\hat{x}$: Nominal State (over-parametrized, constrained)
    - δx: Error State (minimal parametrization, unconstrained)

- The error-state formulation separates the state into a “large” nominal state and a “small” error state.
- The ES-EKF uses local linearization to estimate the error state and uses it to correct the nominal state.
- The ES-EKF can perform better than the vanilla EKF, and provides a natural way to handle constrained quantities like rotations in 3D.
    - the evolution of the error state tends to be closer to linear.



Lesson 5: Limitations of the EKF

Limitations of the EKF | Linearization error

- The EKF works by linearizing the nonlinear motion and measurement models to update the mean and covariance of the state.
- The difference between the linear approximation and the nonlinear function is called linearization error.
- In general, linearization error depends on
  - How nonlinear the function is
  - How far away from the operating point the linear approximation is being used.

Linearization Error | Example

この例のpolar coordinatesからcartesian coordinatesへの変換はLidarに関するmeasurementによくある。

結果：The mean of the linearized distribution is in a very different place from the true mean, and the linearized covariance seriously underestimates the spread of the true output distribution along the y dimension.

問題：For example, the estimated position of lane markers produced from a laser scanner might differ significantly from the actual positions, leading our car to drive over the center line.

Linearization Errorが問題になる基準

The EKF is prone to linearization error when

1. The system dynamics are highly nonlinear.
2. The sensor sampling time is slow relative how fast the system is evolving.
   - For self-driving cars moving very fast relative to the sampling time of your sensors, you're going to get more linearization error than if it's moving slowly.

This has two important consequences（上記の結果と同じ）

1. The estimated mean state can become very different from the true state.
2. The estimated state covariance can fail to capture the true uncertainty in the state.

If your car's estimator diverges while driving, it could easily drive off the road into a ditch or cause a collision with another vehicle.

Limitations of the EKF | Computing Jacobians

Computing Jacobian matrices for complicated nonlinear functions is also a common source of error in EKF implementations!

- Analytical differentiation is prone to human error
- Numerical differentiation can be slow and unstable
- Automatic differentiation (e.g., at compile time) can also behave unpredictably

What if one or more of our models is non-differentiable?

Do we really need linearization for nonlinear Kalman filtering?

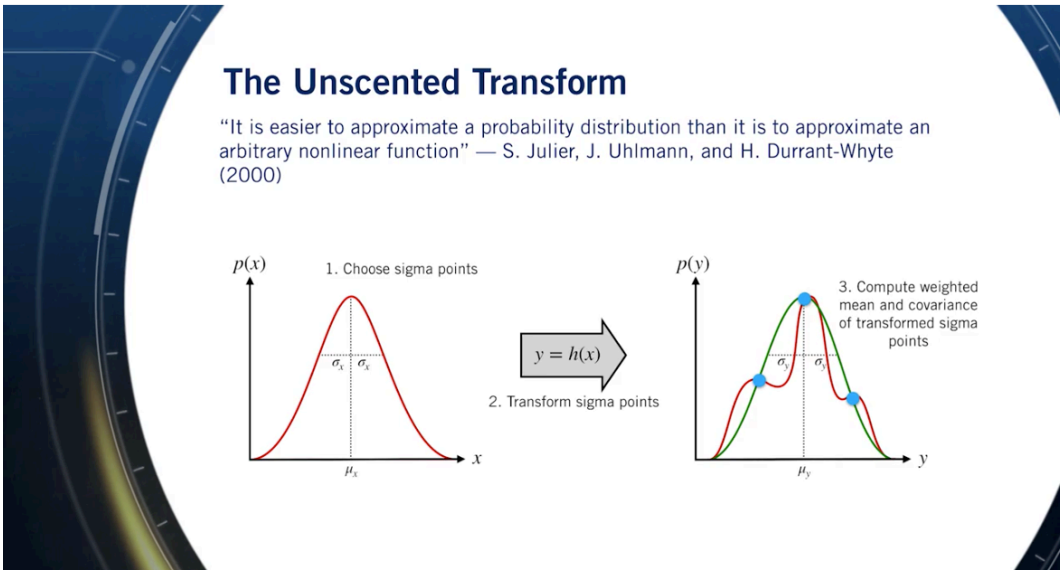
- The EKF uses analytical local linearization and, as a result, is sensitive to linearization errors.
- For highly nonlinear systems, the EKF estimate can diverge and become unreliable.
- Computing complex Jacobian matrices is an error-prone process and must be done with substantial care.

Lesson 6: An Alternative to the EKF - The Unscented Kalman Filter

Unscented Transform gives much higher accuracy than analytical EKF style linearization, for a similar amount of computation, and without needing to compute any Jacobians.

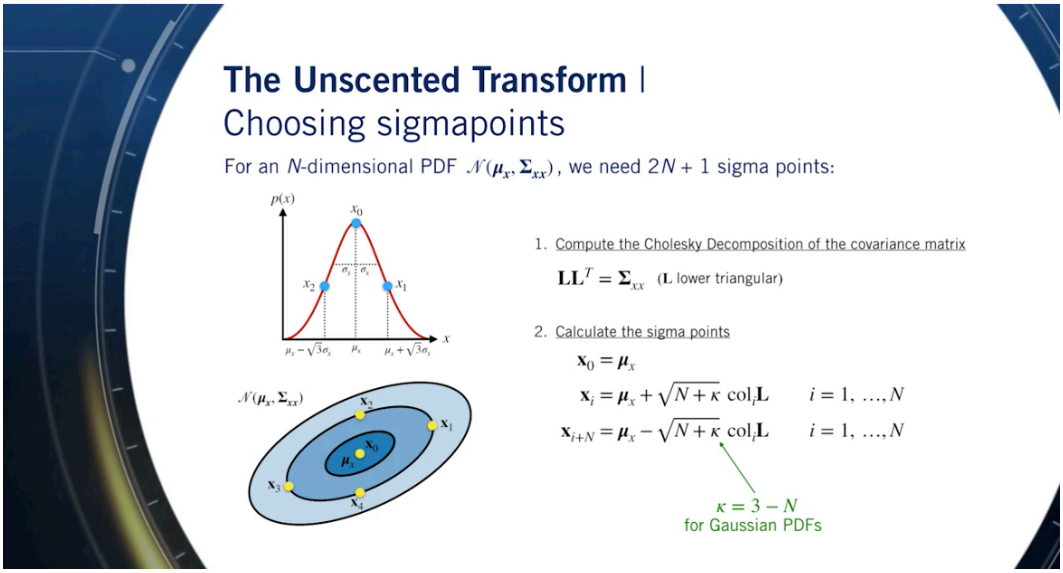
内容

- Use the Unscented Transform to pass a probability distribution through a nonlinear function.
- Describe how the Unscented Kalman Filter (UKF) uses the Unscented Transform in the prediction and correction steps.
- Explain the advantages of the UKF over the EKF.
- Apply the UKF to a simple nonlinear tracking problem.



The Unscented Transform（Screenshotを参考）
Step1: Choose sigma points

- These aren't random samples, they're deterministic samples chosen to be a certain number of standard deviations away from the mean.
- The Unscented Transform is sometimes called the Sigma Point Transform.

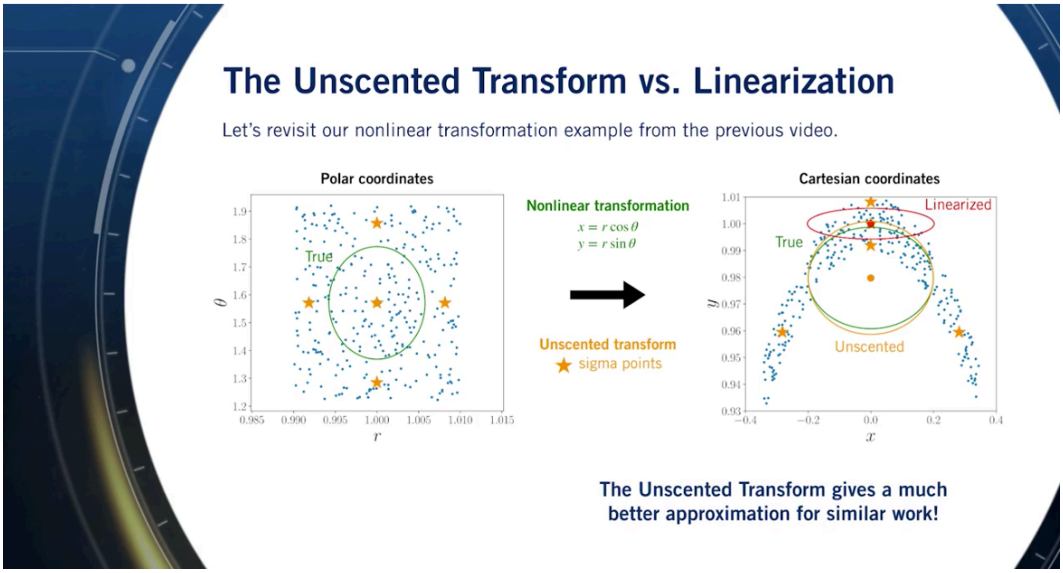


The Unscented Transform | Choosing sigma points（上記Screenshotを参考）

1. sigma pointsの数：$2N + 1$
    1. N: the number of dimensions of the probability distribution.
2. Cholesky Decompositionを計算するため、chol function in Matlab, or the Cholesky function in NumPyを利用する。
3. The parameter κ is a tuning parameter.
    1. For Gaussian distributions, setting $N + \kappa = 3$ is a good choice.

The Unscented Transform | Transforming and recombining

1. Pass each of the $2N+1$ sigma points through the nonlinear function $h(x)$: $y_i = h(x_i)$, $i = 0,\ldots,2N$
2. Compute the mean and covariance of the output PDF
   1. Mean: $\mu_y = \sum_{i=0}^{2N} \alpha_i y_i$
   2. Covariance: $\Sigma_{yy} = \sum_{i=0}^{2N} \alpha_i (y_i - \mu_y)(y_i - \mu_y)^T$
   3. Weights: $\alpha_i = \begin{cases} \frac{\kappa}{N+\kappa}, & i = 0 \\ \frac{1}{2(N+\kappa)}, & \text{otherwise} \end{cases}$



The Unscented Transform vs. Linearization（上記Screenshotを参考）

The Unscented Kalman Filter (UKF)
Can easily use the Unscented Transform in Kalman Filtering framework with nonlinear models:

- Nonlinear motion model: $x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})$, $w_k \sim \mathcal{N}(0, Q_k)$
- Nonlinear measurement model: $y_k = h_k(x_k, v_k)$, $v_k \sim \mathcal{N}(0, R_k)$

Instead of approximating the system equations by linearizing, calculate sigma points and use the Unscented Transform to approximate the PDFs directly.

UKF | Prediction Step

To propagate the state from time $k-1$ to time k, apply the Unscented Transform using the current best guess for the mean and covariance.

1. Compute sigma points:

$$\hat{L}_{k-1}\hat{L}_{k-1}^T = \hat{P}_{k-1}$$
$$\hat{x}_{k-1}^{(0)} = \hat{x}_{k-1}$$
$$\hat{x}_{k-1}^{(i)} = \hat{x}_{k-1} + \sqrt{N+\kappa}\, col_i \hat{L}_{k-1}, i = 1\ldots N$$
$$\hat{x}_{k-1}^{(i+N)} = \hat{x}_{k-1} - \sqrt{N+\kappa}\, col_i \hat{L}_{k-1}, i = 1\ldots N$$

2. Propagate sigma points: $\check{x}_k^{(i)} = f_{k-1}(\hat{x}_{k-1}^{(i)}, u_k, 0), i = 0 \ldots 2N$

$$\alpha^{(i)} = \begin{cases} \frac{\kappa}{N+\kappa}, & i = 0 \\ \frac{1}{2(N+\kappa)}, \text{otherwise} \end{cases}$$

3. Compute predicted mean and covariance:

$$\check{x}_k = \sum_{i=0}^{2N} \alpha^{(i)} \check{x}_k^{(i)}$$

$$\check{P}_k = \sum_{i=0}^{2N} \alpha^{(i)} (\check{x}_k^{(i)} - \check{x}_k)(\check{x}_k^{(i)} - \check{x}_k)^T + Q_{k-1}$$

    1. Additive process noise Q_{k-1}忘れないで!

UKF | Correction step

To correct the state estimate using measurements at time k, use the nonlinear measurement model and the sigma points from the prediction step to predict the measurements.

1. Predict measurements from propagated sigma points: $\hat{y}_k^{(i)} = h_k(\check{x}_k^{(i)}, 0), i = 0 \ldots 2N$
2. Estimate mean and covariance of predicted measurements:

$$\hat{y}_k = \sum_{i=0}^{2N} \alpha^{(i)} \hat{y}_k^{(i)}$$

$$P_y = \sum_{i=0}^{2N} \alpha^{(i)} (\hat{y}_k^{(i)} - \hat{y}_k)(\hat{y}_k^{(i)} - \hat{y}_k)^T + R_k$$

3. Compute cross-covariance and Kalman gain:

$$P_{xy} = \sum_{i=0}^{2N} \alpha^{(i)} (\check{x}_k^{(i)} - \check{x}_k)(\hat{y}_k^{(i)} - \hat{y}_k)^T$$

$$K_k = P_{xy} P_y^{-1}$$

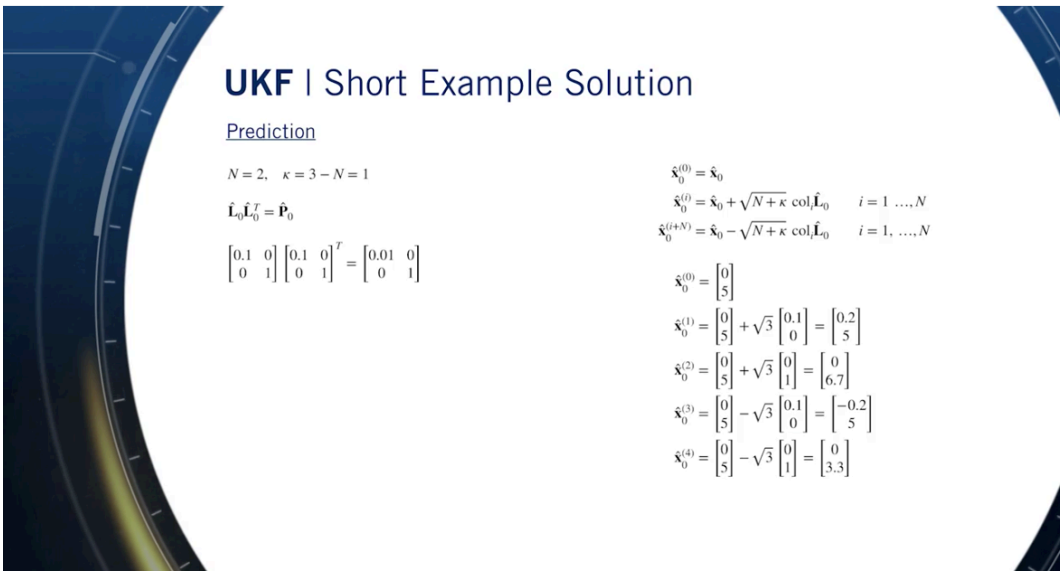
    1. To compute the Kalman gain, need the cross-covariance between the predicted state and the predicted measurements, which tells how the measurements are correlated with the state.

4. Compute corrected mean and covariance:

$$\hat{x}_k = \check{x}_k + K_k(y_k - \hat{y}_k)$$

$$\hat{P}_k = \check{P}_x - K_k P_y K_k^T$$

    1. このステップはEKFと一緒。



UKF | Short Example Solution

Prediction

$$\check{\mathbf{x}}_1^{(i)} = \mathbf{f}_0(\hat{\mathbf{x}}_0^{(i)}, \mathbf{u}_0, \mathbf{0}) \qquad i = 0, \ldots, 2N$$

$$\check{\mathbf{x}}_1^{(0)} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(1)} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.2 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 2.7 \\ 4 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(2)} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 6.7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 3.4 \\ 5.7 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(3)} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -0.2 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 2.3 \\ 4 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(4)} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 3.3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 1.6 \\ 2.3 \end{bmatrix}$$

$$\alpha^{(i)} = \begin{cases} \dfrac{\kappa}{N+\kappa} = \dfrac{1}{2+1} = \dfrac{1}{3} & i = 0 \\ \dfrac{1}{2}\dfrac{1}{N+\kappa} = \dfrac{1}{2}\dfrac{1}{2+1} = \dfrac{1}{6} & \text{otherwise} \end{cases}$$

$$\check{\mathbf{x}}_k = \sum_{i=0}^{2N} \alpha^{(i)} \check{\mathbf{x}}_k^{(i)}$$

$$= \frac{1}{3}\begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 2.7 \\ 4 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 3.4 \\ 5.7 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 2.3 \\ 4 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 1.6 \\ 2.3 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$$

UKF | Short Example Solution

Prediction

$$\check{\mathbf{P}}_k = \sum_{i=0}^{2N} \alpha^{(i)} \left(\check{\mathbf{x}}_k^{(i)} - \check{\mathbf{x}}_k\right)\left(\check{\mathbf{x}}_k^{(i)} - \check{\mathbf{x}}_k\right)^T + \mathbf{Q}_{k-1}$$

$$= \frac{1}{3}\left(\begin{bmatrix} 2.5 \\ 4 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)\left(\begin{bmatrix} 2.5 \\ 4 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)^T +$$

$$\frac{1}{6}\left(\begin{bmatrix} 2.7 \\ 4 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)\left(\begin{bmatrix} 2.7 \\ 4 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)^T + \frac{1}{6}\left(\begin{bmatrix} 3.4 \\ 5.7 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)\left(\begin{bmatrix} 3.4 \\ 5.7 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)^T +$$

$$\frac{1}{6}\left(\begin{bmatrix} 2.3 \\ 4 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)\left(\begin{bmatrix} 2.3 \\ 4 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)^T + \frac{1}{6}\left(\begin{bmatrix} 1.6 \\ 2.3 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)\left(\begin{bmatrix} 1.6 \\ 2.3 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}\right)^T + \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

This is the same result as in the linear Kalman Filter example because the motion model is already linear!

UKF | Short Example Solution

Correction

$$\check{\mathbf{L}}_k \check{\mathbf{L}}_k^T = \check{\mathbf{P}}_k$$

$$\begin{bmatrix} 0.60 & 0 \\ 0.83 & 0.64 \end{bmatrix}\begin{bmatrix} 0.60 & 0 \\ 0.83 & 0.64 \end{bmatrix}^T = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

Redrawn sigma points!

$$\check{\mathbf{x}}_1^{(0)} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(1)} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \sqrt{3}\begin{bmatrix} 0.60 \\ 0.83 \end{bmatrix} = \begin{bmatrix} 3.54 \\ 5.44 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(2)} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \sqrt{3}\begin{bmatrix} 0 \\ 0.64 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 5.10 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(3)} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} - \sqrt{3}\begin{bmatrix} 0.60 \\ 0.83 \end{bmatrix} = \begin{bmatrix} 1.46 \\ 2.56 \end{bmatrix}$$

$$\check{\mathbf{x}}_1^{(4)} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} - \sqrt{3}\begin{bmatrix} 0 \\ 0.64 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 2.90 \end{bmatrix}$$

$$\check{\mathbf{y}}_1^{(i)} = h_1(\check{\mathbf{x}}_1^{(i)}, 0) \qquad i = 0, \ldots, 2N$$

$$\check{y}_1^{(0)} = \tan^{-1}\left(\frac{20}{40 - 2.5}\right) = 0.49$$

$$\check{y}_1^{(1)} = \tan^{-1}\left(\frac{20}{40 - 3.54}\right) = 0.50$$

$$\check{y}_1^{(2)} = \tan^{-1}\left(\frac{20}{40 - 2.5}\right) = 0.49$$

$$\check{y}_1^{(3)} = \tan^{-1}\left(\frac{20}{40 - 1.46}\right) = 0.48$$

$$\check{y}_1^{(4)} = \tan^{-1}\left(\frac{20}{40 - 2.5}\right) = 0.49$$

UKF | Short Example Solution

Correction

$$\hat{y}_1 = \sum_{i=0}^{2N} \alpha^{(i)} \hat{y}_1^{(i)}$$
$$= \frac{1}{3}(0.49) + \frac{1}{6}(0.50) + \frac{1}{6}(0.49) + \frac{1}{6}(0.48) + \frac{1}{6}(0.49)$$
$$= 0.49$$

$$P_y = \sum_{i=0}^{2N} \alpha^{(i)} \left(\hat{y}_1^{(i)} - \hat{y}_k\right)\left(\hat{y}_1^{(i)} - \hat{y}_k\right)^T + R_k$$
$$= \frac{1}{3}(0.49 - 0.49)^2 + \frac{1}{6}(0.50 - 0.49)^2 + \frac{1}{6}(0.49 - 0.49)^2 + \frac{1}{6}(0.48 - 0.49)^2 + \frac{1}{6}(0.49 - 0.49)^2 + 0.01$$
$$= 0.01$$

UKF | Short Example Solution

Correction

$$\mathbf{P}_{xy} = \sum_{i=0}^{2N} \alpha^{(i)} \left(\check{\mathbf{x}}_k^{(i)} - \check{\mathbf{x}}_k\right)\left(\hat{y}_1^{(i)} - \hat{y}_k\right)^T$$
$$= \frac{1}{3}\left(\begin{bmatrix}2.5\\4\end{bmatrix} - \begin{bmatrix}2.5\\4\end{bmatrix}\right)(0.49 - 0.49) + \frac{1}{6}\left(\begin{bmatrix}3.54\\5.44\end{bmatrix} - \begin{bmatrix}2.5\\4\end{bmatrix}\right)(0.50 - 0.49) + \frac{1}{6}\left(\begin{bmatrix}2.5\\5.10\end{bmatrix} - \begin{bmatrix}2.5\\4\end{bmatrix}\right)(0.49 - 0.49)$$
$$+ \frac{1}{6}\left(\begin{bmatrix}1.46\\2.56\end{bmatrix} - \begin{bmatrix}2.5\\4\end{bmatrix}\right)(0.48 - 0.49) + \frac{1}{6}\left(\begin{bmatrix}2.5\\2.90\end{bmatrix} - \begin{bmatrix}2.5\\4\end{bmatrix}\right)(0.49 - 0.49)$$
$$= \begin{bmatrix}0.004\\0.006\end{bmatrix}$$

$$\mathbf{K}_1 = \mathbf{P}_{xy} P_y^{-1} = \begin{bmatrix}0.004\\0.006\end{bmatrix}\frac{1}{0.01} = \begin{bmatrix}0.40\\0.55\end{bmatrix}$$

$$\check{\mathbf{x}}_1 = \check{\mathbf{x}}_1 + \mathbf{K}_1\left(y_1 - \hat{y}_1\right) = \begin{bmatrix}2.5\\4\end{bmatrix} + \begin{bmatrix}0.40\\0.55\end{bmatrix}(0.52 - 0.49) = \begin{bmatrix}2.51\\4.02\end{bmatrix} = \begin{bmatrix}\check{p}_1\\\check{p}_1\end{bmatrix}$$

UKF | Short example
条件はEKFの例と一緒。

Prediction:
1. $\hat{P}_{k-1} = \hat{P}_0$のCholesky decomposition、sigma points取得（$\hat{x}_{k-1}^{(i)} = \hat{x}_0^{(i)}$）
2. transformed sigma points計算、transformed sigma pointsのmean計算（$\check{x}_k = \check{x}_1$）
3. transformed sigma pointsのcovariance計算（$\check{P}_k = \check{P}_1$）

Correction:
1. $\check{P}_k = \check{P}_1$のCholesky decomposition、sigma points取得（$\check{x}_k^{(i)} = \check{x}_1^{(i)}$）、transformed sigma points計算（$\hat{y}_k^{(i)} = \hat{y}_1^{(i)}$）

2. transformed sigma pointsのmeanとcovariance計算（$\hat{y}_k = \hat{y}_1$、P_y）
3. cross-covarianceの計算（P_{xy}）、Kalman gainの計算（$K_k = K_1$）、time kの予測値$\hat{x}_k = \hat{x}_1$の計算（stateの更新）

- The UKF uses the unscented transform to adapt the Kalman filter to nonlinear systems.
- The unscented transform works by passing a small set of carefully chosen samples through a nonlinear system, and computing the mean and covariance of the outputs.
- The unscented transform does a better job of approximating the output distribution than analytical local linearization, for similar computational cost.

	EKF	**ES-EKF**	**UKF**
Operating Principle	Linearization (Full State)	Linearization (Error State)	Unscented Transform
Accuracy	Good	Better	Best
Jacobians	Required	Required	Not required
Speed	Slightly faster	Slightly faster	Slightly slower

Because of UKF's accuracy and simplicity, recommend using the UKF over the EKF whenever possible in projects.