Report

Bayesian Filtering and Calibration

Ivan Sidorov

Contents

1 Filtering

Firstly, it is necessary to establish formally the filtering problem [1].

Suppose the state $X_t \in \mathbb{R}^n$ at time t of a system is given by a stochastic differential equation:

$$\frac{dX_t}{dt} = b\big(t, X_t\big) \; + \; \sigma\big(t, X_t\big)\, W_t, \quad t \geq 0$$

where

$$b\colon \mathbb{R}^{n+1} \to \mathbb{R}^n, \quad \sigma\colon \mathbb{R}^{n+1} \to \mathbb{R}^{n\times p}$$

and W_t is p-dimensional white noise. The Ito interpretation of this equation is:

$$dX_t \; = \; b\big(t, X_t\big)\, dt \; + \; \sigma\big(t, X_t\big)\, dU_t$$

where U_t is p-dimensional Brownian motion. We also assume that the distribution of X_0 is known and independent of U_t.

In the continuous version of the filtering problem we assume that the observations $H_t \in \mathbb{R}^m$ are performed continuously and are of the form:

$$H_t \; = \; c\big(t, X_t\big) \; + \; \gamma\big(t, X_t\big)\, \widetilde{W}_t,$$

where

$$c\colon \mathbb{R}^{n+1} \to \mathbb{R}^m, \quad \gamma\colon \mathbb{R}^{n+1} \to \mathbb{R}^{m\times r}$$

and $\widetilde{W}_t$ denotes r-dimensional white noise, independent of U_t and X_0.

To obtain a tractable mathematical interpretation of observation equation we introduce:

$$Z_t \; = \; \int_0^t H_s\, ds,$$

and thereby we obtain the stochastic integral representation

$$dZ_t \; = \; c\big(t, X_t\big)\, dt \; + \; \gamma\big(t, X_t\big)\, dV_t, \quad Z_0 \; = \; 0,$$

where V_t is r-dimensional Brownian motion, independent of U_t and X_0.

Note that if H_s is known for $0 \leq s \leq t$, then Z_s is also known for $0 \leq s \leq t$, and conversely. So no information is lost or gained by considering Z_t as our "observations" instead of H_t. But this allows us to obtain a well-defined mathematical model of the situation.

[1] Øksendal, *Stochastic Differential Equations: An Introduction with Applications*, 6th ed., Springer, 2010. Chapter 6.

The filtering problem is the following:

Given the observations Z_s for $0 \le s \le t$, what is the best estimate $\widehat{X}_t$ of the state X_t based on these observations?

It is necessary to find a precise mathematical formulation of this problem: By saying that the estimate $\widehat{X}_t$ is based on the observations $\{Z_s;\, s \le t\}$ we mean that

$$\widehat{X}_t(\cdot) \text{ is } \mathcal{G}_t\text{-measurable},$$

where $\mathcal{G}_t$ is the σ-algebra generated by $\{Z_s(\cdot),\, s \le t\}$.

By saying that $\widehat{X}_t$ is the best such estimate we mean that

$$\int_\Omega \left|X_t - \widehat{X}_t\right|^2 dP \;=\; \mathbb{E}\big[|X_t - \widehat{X}_t|^2\big] \;=\; \inf_{Y \in \mathcal{K}} \mathbb{E}\big[|X_t - Y|^2\big],$$

Here $(\Omega, \mathcal{F}, P)$ is the probability space corresponding to the $(p+r)$-dimensional Brownian motion (U_t, V_t) starting at 0. $\mathbb{E}$ denotes expectation with respect to P, and

$$\mathcal{K} \;:=\; \mathcal{K}(Z,t) \;:=\; \big\{\, Y \colon \Omega \to \mathbb{R}^n;\; Y \in L^2(P) \text{ and } Y \text{ is } \mathcal{G}_t\text{-measurable}\big\},$$

while $L^2(P) = L^2(\Omega, P)$.

1.1 Motivation

1.1.1 Recursive Least Squares

We start with a simple Recursive Least Squeres model, which was invented by Carl Friedrich Gauss in 1821, as it was one of the first Adaptive Filters ever known [2]. So the model specification is as follows:

$$y = b_1 u_1 + b_2 u_2 + \cdots + b_m u_m = \theta^T \varphi$$

where

$$\theta = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \in \mathbb{R}^{m \times 1} \quad \text{and} \quad \varphi = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{pmatrix} \in \mathbb{R}^{m \times 1}.$$

[2] An adaptive filter is a time-varying signal-processing system whose parameters are automatically adjusted to optimize its filtering performance under nonstationary conditions. In other words, it iteratively updates its coefficients based on incoming data, enabling it to track and respond to changes in the statistical properties of the input signal.

To find the best θ we use the Least Squared Error method, which minimizes the Mean Squared Error:

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}\big(\hat{y}(t \mid \theta) - y(t)\big)^2 \quad \Rightarrow \quad \hat{\theta}^{LS} = \arg\min_{\theta} V_N(\theta).$$

The difference from the basic Least Squares Error algorithm lies in Prediction-Error Formalism introduced by Gauss which enabled to estimate parameters "on the fly", assuming we already have an optimal LS estimates given the previous observed data and a new observation comes in:

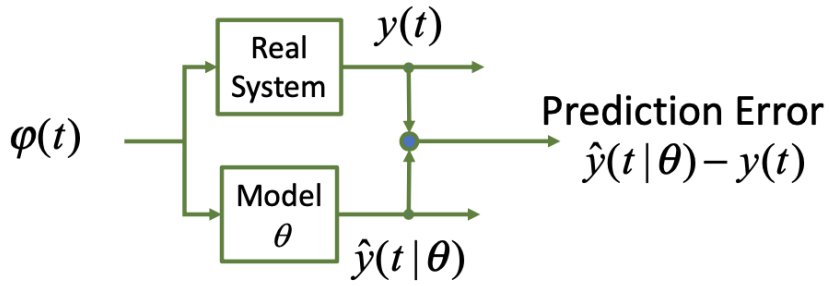

Figure 1: Prediction-Error Formalism [3]

The Recursive Least Squares Algorithm:

$$\hat{\theta}^{LS}(t) = \hat{\theta}^{LS}(t-1) \; + \; K_t \Big[\, y(t) \; - \; \hat{y}\big(t \mid \hat{\theta}^{LS}(t-1)\big)\Big]$$

where

$$\begin{cases} K_t \;=\; \dfrac{P_{t-1}\varphi(t)}{1-\varphi^T(t)\,P_{t-1}\,\varphi(t)}, \\ P_t \;=\; P_{t-1} \;-\; \dfrac{P_{t-1}\,\varphi(t)\,\varphi^T(t)\,P_{t-1}}{1+\varphi^T(t)\,P_{t-1}\,\varphi(t)}, \end{cases} \qquad t = 1, 2, \ldots$$

Where K is a Gain. And initial conditions are:

$$\hat{\theta}^{LS}(0) = \theta_0 \quad (\text{arbitrary, e.g. } \theta_0 = 0), \qquad P_0 = \text{positive-definite matrix (e.g. } P_0 = I).$$

[3] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 3: Recursive Least Squares with Forgetting Factor.

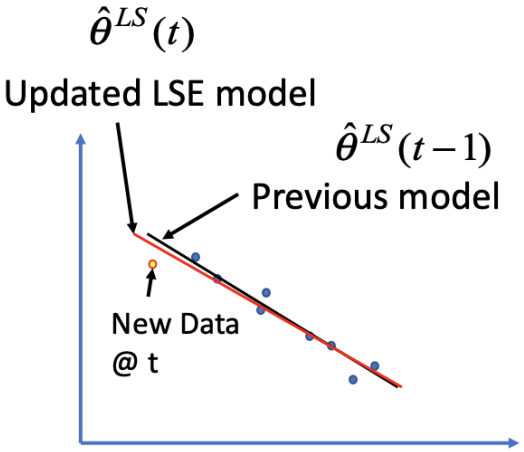

Figure 2: Recursive Least Squares [4]

To understand the physical meaning of the Gain we rewrite the optimal Gain in the following way: $K_t = P_t\varphi(t)$. Therefore, the more intuitive formula can be derived:

$$\Delta\theta = \hat{\theta}^{LS}(t) - \hat{\theta}^{LS}(t-1) = K_t\Big[y(t) - \hat{y}\big(t \mid \hat{\theta}^{LS}(t-1)\big)\Big] =$$

$$\boxed{P_t \times (Regressor) \times (Prediction\ Error)}$$

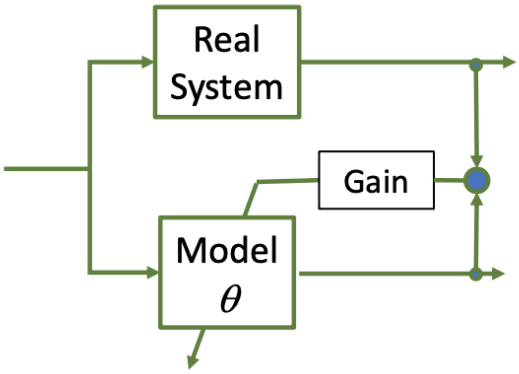

Figure 3: Error Correction [4]

So what is the physical meaning of the inverse matrix $P_t = \Big[\sum_{i=1}^{t}\varphi(i)\,\varphi^T(i)\Big]^{-1} \underset{non-singular}{\Leftrightarrow} P_t^{-1} = \sum_{i=1}^{t}\varphi(i)\,\varphi^T(i)$? If it becomes clear then the physical meaning of the Gain will be clear as well.

To characterize how the regressor data (input) are distributed, consider a unit vector v and quantify the total squared strength of $\varphi(1), \varphi(2), \ldots, \varphi(t)$ in the direction of vector v.

[4] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 3: Recursive Least Squares with Forgetting Factor.

$$\sum_{i=1}^{t}\left(\varphi^T(i)\,v\right)^2 = \sum_{i=1}^{t} v^T\,\varphi(i)\,\varphi^T(i)\,v = v^T\Big(\sum_{i=1}^{t}\varphi(i)\,\varphi^T(i)\Big)v = v^T\,P_t^{-1}\,v.$$

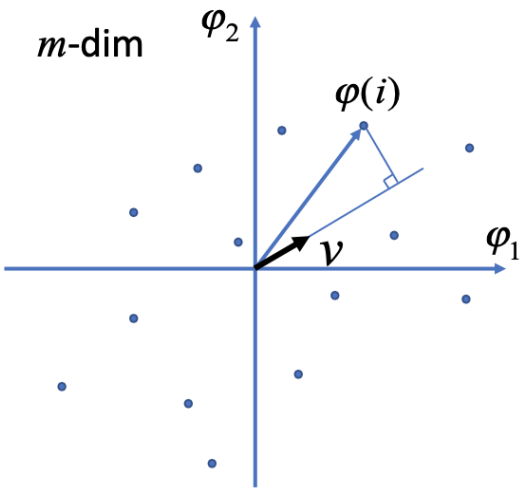

Figure 4: Strength of Observations [5]

Now we want to find the maximum of the total squared signal strength → find the direction of unit vector v that maximizes $v^T P_t^{-1} v$ subject to $\mid v \mid^2 = 1$.

$$L = v^T\Big(\sum_{i=1}^{t}\varphi(i)\varphi^T(i)\Big)v - \lambda(v^T v - 1)\frac{\partial L}{\partial v} = 0 \Rightarrow$$

$$2\sum_{i=1}^{t}\varphi(i)\varphi^T(i)v - 2\lambda v = 0 \Rightarrow \sum_{i=1}^{t}\varphi(i)\varphi^T(i)v = \lambda v$$

By definition λ, v are eigen value and eigen vector of the matrix P_t^{-1}.

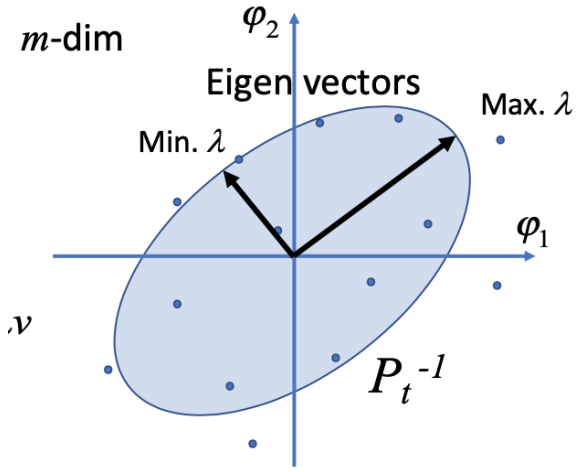

Figure 5: Eigen decomposition [5]

Then, assuming that $\sum_{i=1}^{t}\varphi(i)\,\varphi^T(i)$ is non-singular matrix, we can diagonalize P_t^{-1} and

[5] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 3: Recursive Least Squares with Forgetting Factor.

then invert it to get P_t:

$$P_t^{-1} = \sum_{i=1}^{t} \varphi(i)\varphi^T(i) = T \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_m \end{pmatrix} T^T,$$

where $T = \begin{pmatrix} v_1 & \cdots & v_m \end{pmatrix}$ is an orthonormal matrix.

$$P_t = (T^T)^{-1} \begin{pmatrix} \frac{1}{\lambda_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\lambda_m} \end{pmatrix} T^{-1} = T \begin{pmatrix} \frac{1}{\lambda_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\lambda_m} \end{pmatrix} T^T.$$

Now we are ready for the physical understanding of the Gain K_t:

Suppose that a new sample φ^* is observed, which is on the line of v_1 associated with the largest eigen value of matrix P_t. $\Rightarrow$ In the direction of v_1, we have already obtained a lot of data $\Rightarrow$ No need to make a large correction to $\hat{\theta}^{LS}(t-1)$ as $P_t\,\varphi^* = \frac{1}{\lambda_{\max}}\varphi^*$ is small and vice versa.

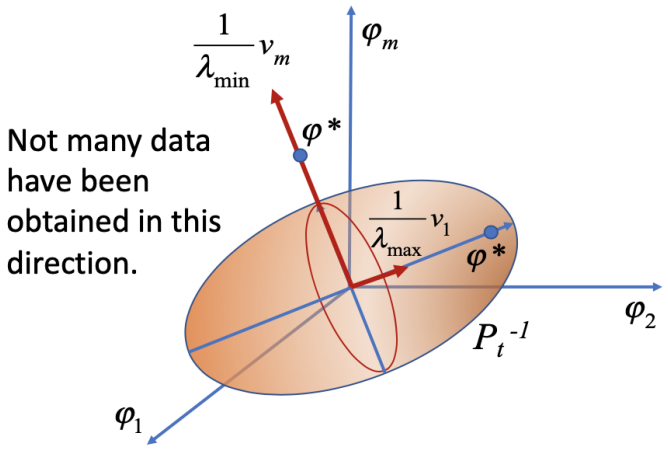

Figure 6: Error Correction Intuition [6]

1.1.2 Discrete Time State Observer

Real model: Linear Time-Varying System [7]

1) State (transition) equation: $x_{t+1} = A_t x_t + B_t u_t$, where $x_t \in \mathbb{R}^{n\times 1}$ is a state vector and $u_t \in \mathbb{R}^{r\times 1}$ is an input vector.

[6] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 3: Recursive Least Squares with Forgetting Factor.

[7] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 7: Discrete-Time Kalman Filter

2) Observation (measurement) equation: $y_{t+1} = H_t x_t$, where $y_t \in \mathbb{R}^{l\times 1}$ is an observation vector.

Luenberger Observer: $\hat{x}_{t+1} = A_t\hat{x}_t + B_t u_t + L(y_t - \hat{y}_t)$

There is no need for extending the discussion of Luenberger Observer and finding the formula for the optimal L right now as it is derived from Kalman Filter Section when you simply ignore the noise.

1.2 Linear Gaussian Filters

1.2.1 Kalman Filter

Linear time-varying system with additive noise can be described as follows.

Real model:

1) State (transition) equation:

$$x_{t+1} = A_t\, x_t + B_t\, u_t + G_t\, w_t,$$

where $x_t \in \mathbb{R}^{n\times 1}$ is the state vector, $u_t \in \mathbb{R}^{r\times 1}$ is the control (input) vector, and $w_t \in \mathbb{R}^{n\times 1}$ is the process noise. The noise has zero mean,

$$\mathbb{E}[w_t] = 0, \quad \mathbb{E}[w_t\, w_s^T] = \begin{cases} Q_t, & \text{if } t = s, \\ 0, & \text{if } t \neq s, \end{cases}$$

for some covariance matrix Q_t.

2) Observation (measurement) equation:

$$y_{t+1} = H_t\, x_t + v_t,$$

where $y_t \in \mathbb{R}^{l\times 1}$ is the measurement vector and $v_t \in \mathbb{R}^{l\times 1}$ is the measurement noise with

$$\mathbb{E}[v_t] = 0, \quad \mathbb{E}[v_t\, v_s^T] = \begin{cases} R_t, & \text{if } t = s, \\ 0, & \text{if } t \neq s, \end{cases}$$

and covariance matrix R_t.

We also assume that there is no cross-correlation between w_t and v_s:

$$\mathbb{E}[w_t\, v_s^T] = 0,$$

even though this restriction can be removed if necessary.

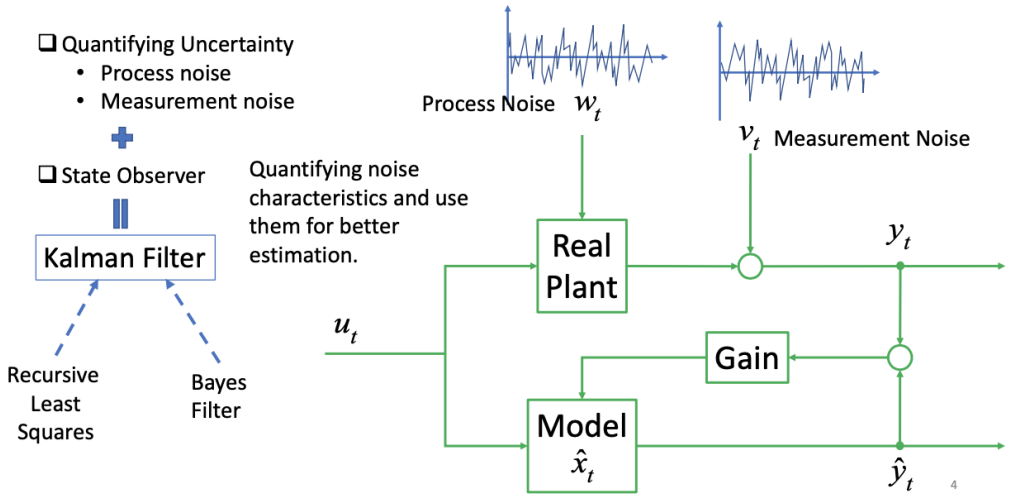

Figure 7: Kalman filter scheme [8]

Worth noticing that there is no fundamental difference between parameter estimation and state estimation. Parameters can be treated as state variables that are constant but initially unknown:

$$\hat{\theta}_{t+1} = I\,\hat{\theta}_t + K_t\,\big[y_t - \hat{y}_t\big].$$

For the optimal filtering problem, the task is to find a state estimate $\hat{x}_t$ that minimizes the mean squared prediction error

$$\overline{J}_t = \mathbb{E}\big[(\hat{x}_t - x_t)^2\big].$$

Two well-known statements about the Kalman filter are:

1) Assuming the noise is Gaussian, the Kalman filter is optimal among all linear and non-linear filters:

$$\hat{x}_{t+1} = A_t\,\hat{x}_t + B_t\,u_t + K_t\,\big[y_t - \hat{y}_t\big].$$

2) Assuming the filter structure is linear, meaning the correction is $K_t\,[\,y_t - \hat{y}_t\,]$, the Kalman filter is then the best linear filter regardless of noise distribution.

Here, we rely on the second statement and show how it is derived.

[8] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 7: Discrete-Time Kalman Filter

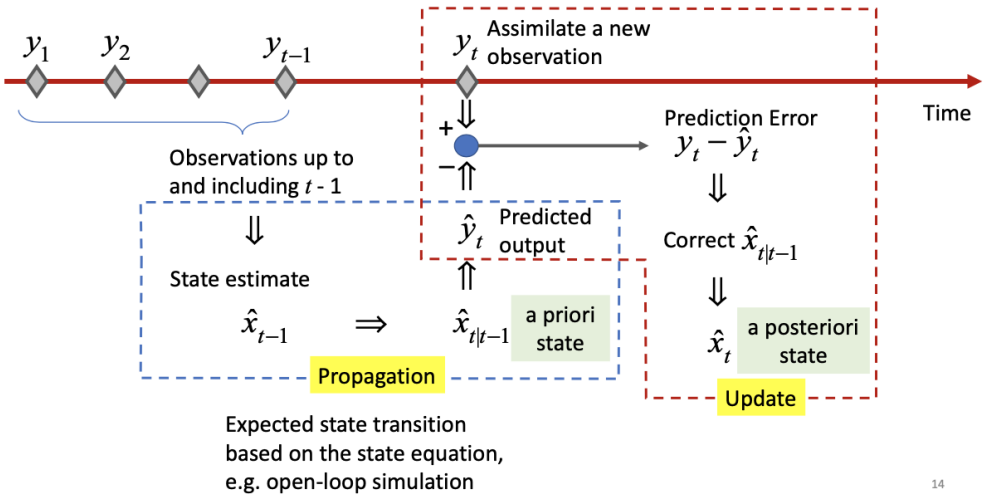

Figure 8: High-level flow of the Kalman filter [9]

So, the task is to predict the transition state using the state equation:

$$\hat{x}_t = A_{t-1}\hat{x}_{t-1} + B_{t-1}u_{t-1} + G_{t-1}w_{t-1},$$

without loss of generality we can set the deterministic part to zero ($u_{t-1} = 0$).

Let's start with the Propagation:

$$\hat{x}_{t|t-1} = \mathbb{E}[x_t] = \mathbb{E}[A_{t-1}x_{t-1} + G_{t-1}w_{t-1}] = A_{t-1}\mathbb{E}[x_{t-1}] = A_{t-1}\hat{x}_{t-1}.$$

This a priori state estimate is only based on information before measurement y_t is available.

$$\hat{y}_t = \mathbb{E}[y_t] = \mathbb{E}[H_t x_t + v_t] = H_t\mathbb{E}[x_t] = H_t\hat{x}_{t|t-1} = H_t A_{t-1}\hat{x}_{t-1}.$$

Now proceed with the state Update following the prediction law stated in the second statement

$$\hat{x}_t = \hat{x}_{t|t-1} + K_t\big[y_t - \hat{y}_t\big].$$

The optimal Gain in that case would be

$$K_t = \arg\min_{K_t} \overline{J}_t = \arg\min_{K_t} \mathbb{E}\big[(\hat{x}_t - x_t)^2\big].$$

Note that there are two types of prediction error:

1) A posteriori error: $e_t = \hat{x}_t - x_t$

2) A priori error: $\epsilon_t = \hat{x}_{t|t-1} - x_t$

These two prediction errors are related to each other:

[9] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 7: Discrete-Time Kalman Filter

$$e_t = \hat{x}_t - x_t = \hat{x}_{t|t-1} + K_t\big[y_t - \hat{y}_t\big] - x_t = \hat{x}_{t|t-1} + K_t\big[H_t x_t + v_t - H_t\hat{x}_{t|t-1}\big] - x_t$$

$$= \underbrace{\hat{x}_{t|t-1} - x_t}_{\epsilon_t} + K_t H_t\underbrace{\big(x_t - \hat{x}_{t|t-1}\big)}_{-\epsilon_t} + K_t v_t = \boxed{(I - K_t H_t)\,\epsilon_t \;+\; K_t\,v_t}.$$

And the formula in the box presents a trade-off the Gain is meant to solve: as the Gain becomes higher, the priori error is more reduced but the measurement noise is amplified.

Now basically we compute an optimal Gain by minimization of $\overline{J}_t$, which is done by matrix differentiation and accurate expectation; since both $\frac{d(\cdot)}{dK}$ and $\mathbb{E}[\cdot]$ are linear, their order can be swapped. Not to bother with these calculations, we immediately jump to the optimal Kalman Gain formula:

$$K_t = P_{t|t-1}\,H_t^T\big(H_t\,P_{t|t-1}\,H_t^T + R_t\big)^{-1},$$

where $P_{t|t-1} = \mathbb{E}[\epsilon_t\,\epsilon_t^T]$ is the a priori error covariance. We actually need to propagate and update this covariance too, and that is done as follows:

1) Update:

$$P_t = \mathbb{E}\big[e_t\,e_t^T\big] = \mathbb{E}\Big[\big((I - K_t H_t)\,\epsilon_t + K_t\,v_t\big)\,\big((I - K_t H_t)\,\epsilon_t + K_t\,v_t\big)^T\Big]$$

$$= (I - K_t H_t)\,\mathbb{E}[\epsilon_t\,\epsilon_t^T]\,(I - K_t H_t)^T + K_t\,\mathbb{E}[v_t\,v_t^T]\,K_t^T + (I - K_t H_t)\,\mathbb{E}[\epsilon_t\,v_t^T]\,K_t^T + K_t\,\mathbb{E}[v_t\,\epsilon_t^T]\,(I - K_t H_t)^T$$

$$= \boxed{(I - K_t H_t)\,P_{t-1}\,(I - K_t H_t)^T + K_t\,R_t\,K_t^T}.$$

That expression can be further simplified by inserting the Kalman Gain formula

$$P_t = (I - K_t H_t)\,P_{t|t-1},$$

which also implies that the a posteriori error covariance is less than or equal to the a priori error covariance $\big(P_t \leq P_{t|t-1}\big)$.

2) Propagation:

$$\epsilon_{t+1} = \hat{x}_{t+1|t} - x_{t+1} = A_t\,\hat{x}_t - \big(A_t\,x_t + G_t\,w_t\big) = A_t\,e_t - G_t\,w_t,$$

$$P_{t+1|t} = \mathbb{E}\big[\epsilon_{t+1}\,\epsilon_{t+1}^T\big] = \mathbb{E}\big[(A_t\,e_t - G_t\,w_t)\,(A_t\,e_t - G_t\,w_t)^T\big]$$

$$= A_t\,\mathbb{E}[e_t\,e_t^T]\,A_t^T + G_t\,\mathbb{E}[w_t\,w_t^T]\,G_t^T - A_t\,\mathbb{E}[e_t\,w_t^T]\,G_t^T - G_t\,\mathbb{E}[w_t\,e_t^T]\,A_t^T$$

$$= \boxed{A_t\,P_t\,A_t^T + G_t\,Q_t\,G_t^T}.$$

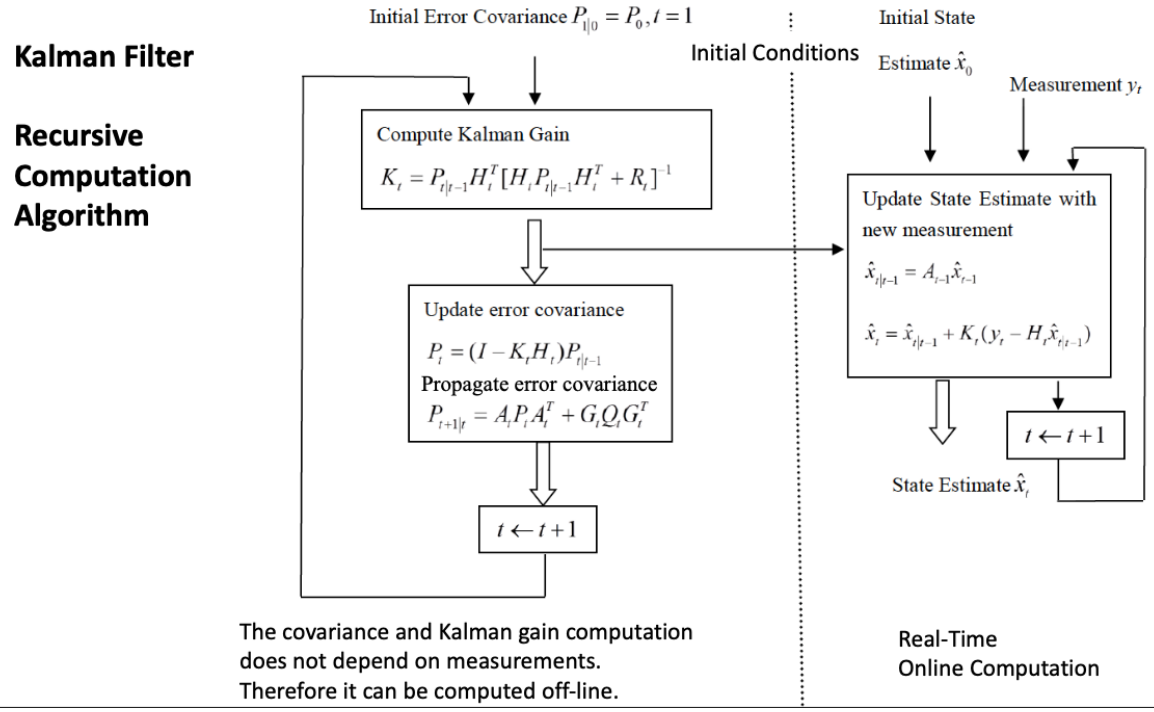

Figure 9: Kalman filter algorithm flow [10]

1.3 Non-Linear Gaussian Filters

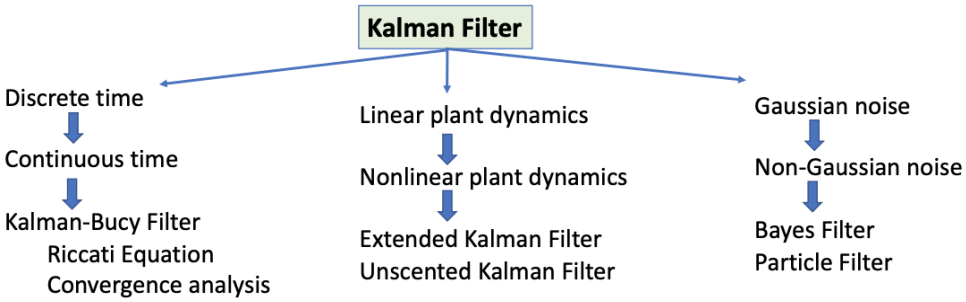

Figure 10: Filters Ierarchy [10]

Extension of Kalman Filter and Non-Linear Dynamical Systems:

1) State Equation: $\dot{x} = f(x, u, t) + w(t)$, $x \in \mathbb{R}^{n\times 1}, u \in \mathbb{R}^{r\times 1}, w \in \mathbb{R}^{n\times 1}$

2) Measurement Equation: $y = h(x, t) + v(t)$, $y \in \mathbb{R}^{l\times 1}, v \in \mathbb{R}^{l\times 1}$

Process and measurement noises are uncorrelated, white noise, with variances $Q(t)$ and $R(t)$.

[10] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 7: Discrete-Time Kalman Filter

1.3.1 Linearized Kalman Filter

The idea of dealing with non-linearity is to linearize the nonlinear state equation around a nominal trajectory

$$\dot{x}^* = f(x^*,\, u,\, t).$$

Now consider the deviation from the nominal trajectory: $x = x^* + \Delta x$. Assuming that the deviation is small, we can linearize the nonlinear state equation:

$$\dot{x} = f(x,\, u,\, t)+w(t) = f(x^*+\Delta x,\, u,\, t)+w(t) = f(x^*,\, u,\, t)+\left.\frac{\partial f}{\partial x}\right|_{x^*}\Delta x+w(t)+(\text{higher-order small term}).$$

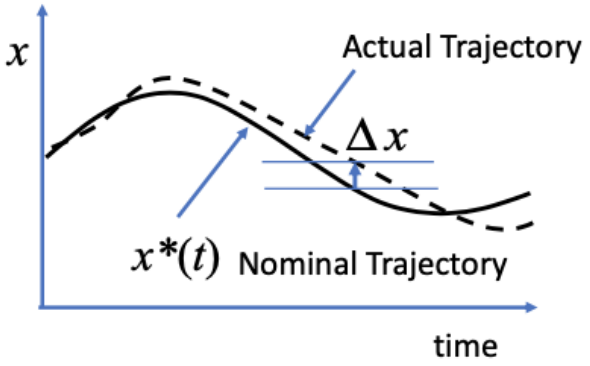

Figure 11: Linearized Kalman Filter [11]

Note that the nominal trajectory satisfies the state equation without process noise, and the derivative of the deviation is given by:

$$\begin{cases} \dot{x} = f(x^*,\, u,\, t) + \left.\frac{\partial f}{\partial x}\right|_{x^*}\Delta x + w(t), \\ \dot{x} = \dot{x}^* + \Delta\dot{x}, \\ \dot{x}^* = f(x^*,\, u,\, t) \end{cases} \quad \Rightarrow \quad \boxed{\Delta\dot{x} \approx \left.\frac{\partial f}{\partial x}\right|_{x^*}\Delta x + w(t).}$$

Now, considering the state process of Δx, we arrive at a new latent dynamics:

$$\dot{x} = F(t)\, x(t) + w(t),$$

where $F(t)$ is the Jacobian matrix. Similarly, the measurement equation can be linearized around the nominal trajectory:

$$y^* + \Delta y \approx h(x^*,\, t) + \left.\frac{\partial h}{\partial x}\right|_{x^*}\Delta x + v(t) \quad \Rightarrow \quad \boxed{\Delta y \approx \left.\frac{\partial h}{\partial x}\right|_{x^*}\Delta x + v(t).}$$

[11] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 9: Extended Kalman Filter and Unscented Kalman Filter.

By treating Δy similarly, we obtain a new observation dynamics:

$$\dot{y} = H(t)\, x(t) + v(t),$$

where $H(t)$ is also a Jacobian matrix.

The original Linear Kalman Filter can then be applied to this linear time-varying system:

1) State Equation:

$$\dot{x} = f(x,\, u,\, t) + w(t) \quad \Rightarrow \quad \dot{x} = F(t)\, x(t) + w(t),$$

where

$$F(t) = \left.\frac{\partial f}{\partial x}\right|_{x^*}.$$

2) Measurement Equation:

$$y = h(x,\, t) + v(t) \quad \Rightarrow \quad y(t) = H(t)\, x(t) + v(t),$$

where

$$H(t) = \left.\frac{\partial h}{\partial x}\right|_{x^*}.$$

3) State propagation and update:

$$\frac{d}{dt}\,\hat{x}(t) = F(t)\,\hat{x}(t) + K(t)\,\big[\,y(t) - \hat{y}(t)\big],$$

where

$$K(t) = P(t)\, H^T(t)\, R^{-1}(t).$$

4) Covariance propagation and update:

$$\frac{dP(t)}{dt} = F(t)\, P(t) + P(t)\, F^T(t) - P(t)\, H^T(t)\, R^{-1}(t)\, H(t)\, P(t) + G(t)\, Q(t)\, G^T(t).$$

where we get system of Riccati equations in continuous time

1.3.2 Extended Kalman Filter

Extended Kalman Filter is a significant improvement in two major aspects:

1) The Jacobian matrices are evaluated not at nominal state $x^*(t)$ but at an estimated state $x^(t)$: $F(t) = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}}, H(t) = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}}$

2) State propagation and update use the full nonlinear state equation and measurement

equation: $\boxed{\dot{\hat{x}} = f(\hat{x}, t) + K(t)\,[y(t) - h(\hat{x}(t), t)]}$

Covariance propagation and update, however, are based on the linearized model of the nonlinear system.

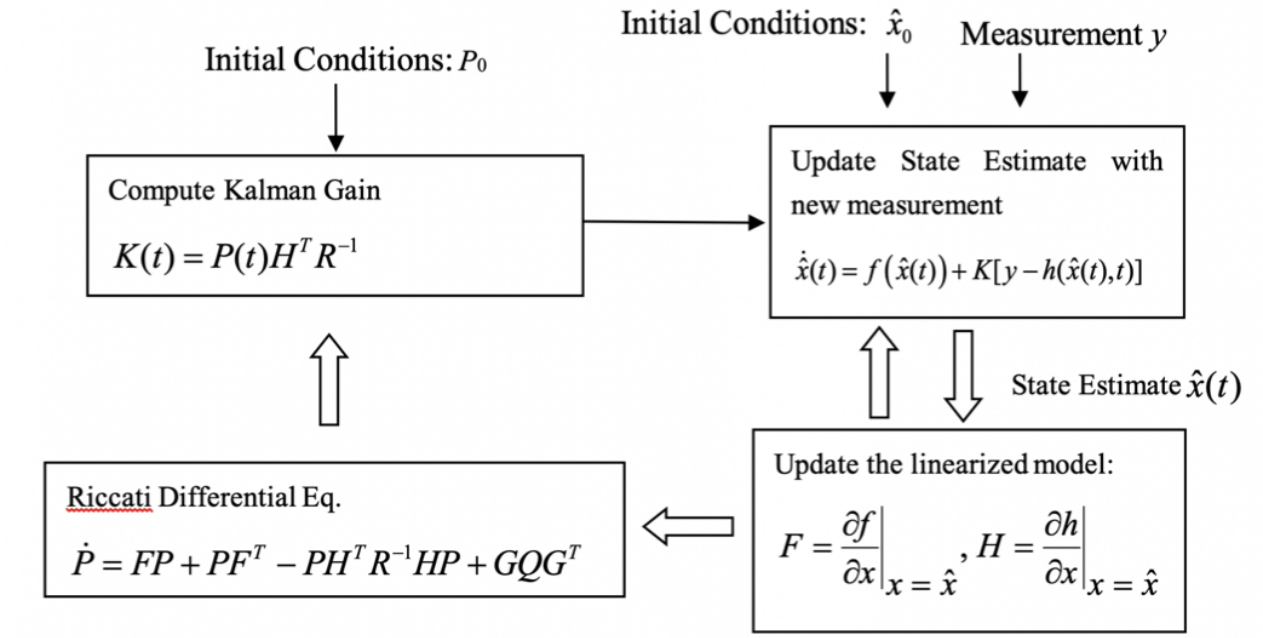

Figure 12: Extended Kalman Filter [12]

1.3.3 Unscented Kalman Filter

Main changes before we dive into it:

1) Unscented Kalman Filter, originally developed by Julier and Uhlmann [1997], uses a different method for computing error covariance matrices.

2) It does not use the Riccati Equations (continuous time) or the covariance propagation and update laws (discrete time). Instead it uses a special technique, called Unscented Transform, for propagating and updating covariance matrices.

3) The key idea is to estimate the error covariance based on a special set of sample points, termed "sigma points", which propagate directly through the original nonlinear model.

Firstly, we consider an approximation algorithm called Unscented Transform in details as it would be much applied to modify Kalman Filter later.

Consider a simple case where one-dimensional random variable X has a Gaussian distribution. Unscented transform represents Gaussian distribution with three special sample points, called Sigma Points:

[12] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 9: Extended Kalman Filter and Unscented Kalman Filter.

$$\tilde{x}^0 = \bar{x}, \quad W_0 = \frac{\kappa}{1+\kappa},$$
$$\tilde{x}^1 = \bar{x} + \sqrt{1+\kappa}\cdot\sigma, \quad W_1 = \frac{1}{2(1+\kappa)},$$
$$\tilde{x}^2 = \bar{x} - \sqrt{1+\kappa}\cdot\sigma, \quad W_2 = \frac{1}{2(1+\kappa)}.$$

It is seen that the first and second moments estimations are unbiased for an arbitrary parameter κ:

$$\sum_{i=0}^{2} W_i\tilde{x}^i = \frac{\kappa}{1+\kappa}\bar{x} + \frac{1}{2(1+\kappa)}\left[(\bar{x}+\sqrt{1+\kappa}\cdot\sigma) + (\bar{x}-\sqrt{1+\kappa}\cdot\sigma)\right] = \frac{\kappa}{1+\kappa}\bar{x} + \frac{2}{2(1+\kappa)}\bar{x} = \bar{x},$$
$$\sum_{i=0}^{2} W_i(\tilde{x}^i-\bar{x})^2 = \frac{\kappa}{1+\kappa}(\bar{x}-\bar{x}) + \frac{1}{2(1+\kappa)}\left[(\bar{x}+\sqrt{1+\kappa}\cdot\sigma-\bar{x})^2 + (\bar{x}-\sqrt{1+\kappa}\cdot\sigma-\bar{x})^2\right]$$
$$= \frac{2}{2(1+\kappa)}(\sqrt{1+\kappa}\cdot\sigma)^2 = \sigma^2.$$

Now, consider a nonlinear transformation from x to y, $y = g(x)$. The distribution of y is no longer Gaussian, but its mean $\mathbb{E}[y]$ and variance $\mathbb{E}[(y-\mathbb{E}[y])^2]$ can characterize the distribution.

We can show that the weighted mean of Sigma points transformed through the nonlinear function can approximate the true mean to the third order: $\bar{y}_{sample} = \sum_{i=0}^{2} W_i\tilde{y}^i = \mathbb{E}[y] + O(4)$. Furthermore, we can show that the weighted variance can approximate the true variance to the second order: $\sigma^2_{sample} = \sum_{i=0}^{2} W_i(\tilde{y}^i - \bar{y}_{sample})^2 = \mathbb{E}[(y-\mathbb{E}[y])^2] + O(3)$.

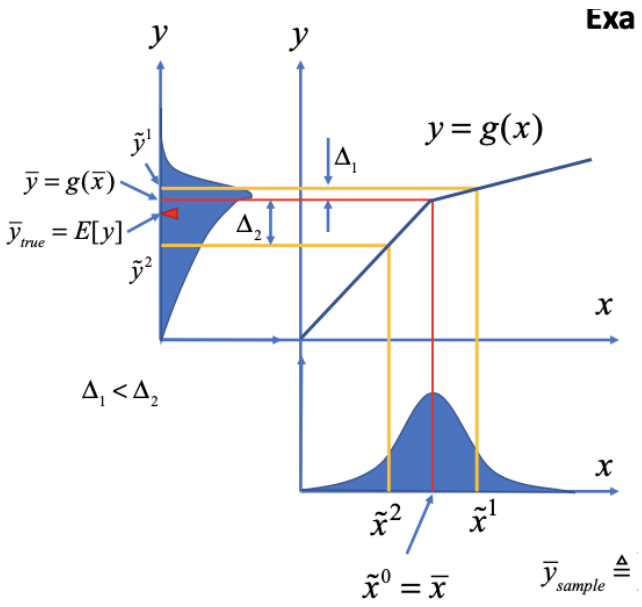

Figure 13: Unscented Transform: 1 dimension [13]

[13] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 9: Extended Kalman Filter and Unscented Kalman Filter.

In general, for an n-dimensional Gaussian distribution

$$p(x) = \det(2\pi P_x)^{-\frac{1}{2}} \exp\Big\{-\tfrac{1}{2}(x-\bar{x})^T P_x^{-1}(x-\bar{x})\Big\},$$

we use $(2n+1)$ Sigma points. Due to properties of covariance matrix P_x it can be diagonalized: $P_x = VDV^T$, where $V = (v_1, v_2, \ldots, v_n),\ D = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$.

Sigma points are taken along the individual eigen vectors with unit length, $v_1, v_2, \ldots, v_n$.

$$\begin{aligned}
&\tilde{x}^0 = \bar{x}, \quad W_0 = \frac{\kappa}{n+\kappa},\\
&\tilde{x}^i = \bar{x} + \sqrt{n+\kappa}\cdot\sigma_i\cdot v_i, \quad W_i = \frac{1}{2(n+\kappa)},\\
&\tilde{x}^{i+n} = \bar{x} - \sqrt{n+\kappa}\cdot\sigma_i\cdot v_i, \quad W_{i+n} = \frac{1}{2(n+\kappa)}.
\end{aligned}$$

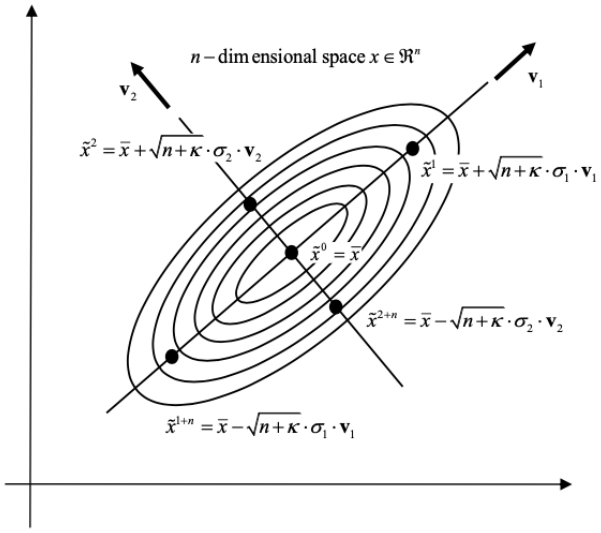

Figure 14: Unscented Transform: n dimensions [14]

Now, consider the Unscented Kalman Filter. The Unscented Transform is used to sample the state first, as we know $\hat{x}_{t-1}, P_{t-1}$ using which we can generate $2(n+1)$ sigma points:

$$\begin{aligned}
&\tilde{x}^0_{t-1} = \hat{x}_{t-1} : \bar{x},\\
&\tilde{x}^i_{t-1} = \hat{x}t - 1 + \sqrt{n+\kappa}\cdot\sigma_i\cdot v_i,\\
&\tilde{x}^{i+n}_{t-1} = \hat{x}_{t-1} - \sqrt{n+\kappa}\cdot\sigma_i\cdot v_i.
\end{aligned}$$

Then we propagate them

$$\tilde{x}^i_{t|t-1} = \mathbb{E}\big[f(\tilde{x}^i_{t-1}) + w_{t-1}\big] = f(\tilde{x}^i_{t-1}),$$

[14] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 9: Extended Kalman Filter and Unscented Kalman Filter.

and then we compute the weighted average

$$\boxed{\tilde{x}^i_{t|t-1,sample} = \sum_{i=0}^{2n} W_i\, \tilde{x}^i_{t|t-1}}.$$

Then we continue with the propagation of the covariance:

$$P_{t|t-1} = \mathbb{E}\big[(\hat{x}_{t|t-1} - x_t)(\hat{x}_{t|t-1} - x_t)^T\big] = \mathbb{E}\big[(\hat{x}_{t|t-1} - f(x_{t-1}, t-1) - w_{t-1})(\hat{x}_{t|t-1} - f(x_{t-1}, t-1) - w_{t-1})^T\big]$$

$$= \mathbb{E}\big[(\hat{x}_{t|t-1} - f(x_{t-1}, t-1))(\hat{x}_{t|t-1} - f(x_{t-1}, t-1))^T\big] + \mathbb{E}[w_{t-1} w_{t-1}^T]$$

$$\boxed{P_{t|t-1,sample} = \sum_{i=0}^{2n} W_i \big(\tilde{x}^i_{t|t-1} - \hat{x}_{t|t-1,sample}\big)\big(\tilde{x}^i_{t|t-1} - \hat{x}_{t|t-1,sample}\big)^T + Q_{t-1}}.$$

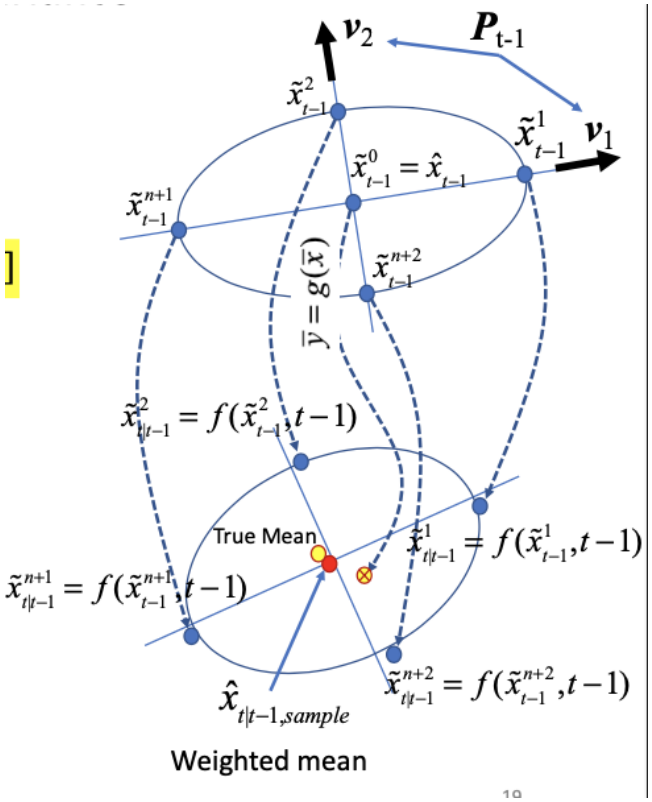

Figure 15: Unscented Propagation: 1 [15]

It is worth saying that in literature you can find similar algorithms called Quadrature Kalman Filter, which basically instead of Unscented Transform that is used for integral approximation uses Quadratures (Sigma Points/Weights ⇒ Quadrature Points/Weights).

Now we aim to update state and covariance using Sigma points. The standard Kalman gain

[15] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 9: Extended Kalman Filter and Unscented Kalman Filter.

and covariance update laws, however, are not applicable to Unscented Transformation. Instead, we will consider an alternative method based on innovation. The Kalman Gain is given by

$$K_t = P_{t|t-1}\, H_t^T \big(H_t\, P_{t|t-1}\, H_t^T + R_t\big)^{-1},$$

but the matrix H_t is not available for our nonlinear system.

Instead, we use the following formula

$$K_t = P_{xy}\, P_y^{-1},$$

where

$$P_{xy} = \mathbb{E}\big[(x_t - \hat{x}_{t|t-1})(y_t - \hat{y}_t)^T\big], \quad P_y = \mathbb{E}\big[(y_t - \hat{y}_t)(y_t - \hat{y}_t)^T\big],$$

which is called "Innovation Covariance". This new formula can be proven by construction:

$$\begin{aligned} P_y &= \mathbb{E}\big[(y_t - \hat{y}_t)(y_t - \hat{y}_t)^T\big] \\ &= \mathbb{E}\Big[\big(H_t x_t + v_t - H_t \hat{x}_{t|t-1}\big)\big(H_t x_t + v_t - H_t \hat{x}_{t|t-1}\big)^T\Big] \\ &= H_t\, \mathbb{E}\big[(x_t - \hat{x}_{t|t-1})(x_t - \hat{x}_{t|t-1})^T\big]\, H_t^T + \mathbb{E}[v_t\, v_t^T] \\ &= H_t\, P_{t|t-1}\, H_t^T + R_t \end{aligned}$$

$$\begin{aligned} P_{xy} &= \mathbb{E}\big[(\hat{x}_{t|t-1} - x_t)(\hat{y}_t - y_t)^T\big] \\ &= \mathbb{E}\Big[\big(\hat{x}_{t|t-1} - x_t\big)\big(H_t \hat{x}_{t|t-1} - H_t x_t - v_t\big)^T\Big] \\ &= \mathbb{E}\big[(\hat{x}_{t|t-1} - x_t)(\hat{x}_{t|t-1} - x_t)^T\big]\, H_t^T - \mathbb{E}\big[(\hat{x}_{t|t-1} - x_t) v_t^T\big] \\ &= P_{t|t-1}\, H_t^T \end{aligned}$$

And now state and covariance can be updated by applying Unscented Transform to the a priori covariance $P_{t|t-1,sample}$. The process is already familiar: first we generate $2n+1$ sigma points then map them

$$\tilde{y}_t^i = h\big(\tilde{x}_{t|t-1}^i, t\big),$$

and eventually we take a weighted mean

$$\hat{y}_{t,sample} = \sum_{i=0}^{2n+1} W_i\, \tilde{y}_t^i.$$

Now we can compute the Innovation Covariance:

$$\begin{aligned} P_y &= \mathbb{E}\big[(y_t - \hat{y}_t)(y_t - \hat{y}_t)^T\big] \\ &= \mathbb{E}\Big[\big(h(x_t,t) + v_t - h(\hat{x}_{t|t-1},t)\big)\big(h(x_t,t) + v_t - h(\hat{x}_{t|t-1},t)\big)^T\Big] \\ &= \mathbb{E}\Big[\big(h(x_t,t) - h(\hat{x}_{t|t-1},t)\big)\big(h(x_t,t) - h(\hat{x}_{t|t-1},t)\big)^T\Big] + \mathbb{E}[v_t\, v_t^T] \end{aligned}$$

$$\boxed{P_y = \sum_{i=0}^{2n} W_i \left(\tilde{y}_t^i - \hat{y}_{t,\text{sample}}\right)\left(\tilde{y}_t^i - \hat{y}_{t,\text{sample}}\right)^T + R_t}.$$

Similarly we compute the cross-covariance

$$P_{xy} = \sum_{i=0}^{2n} W_i \left(\tilde{x}_{t|t-1}^i - \hat{x}_{t|t-1,\text{sample}}\right)\left(\tilde{y}_t^i - \hat{y}_{t,\text{sample}}\right)^T.$$

Therefore it is possible to update the state

$$\hat{x}_t = \hat{x}_{t|t-1,sample} + K_t\left[y_t - \hat{y}_{t,sample}\right].$$

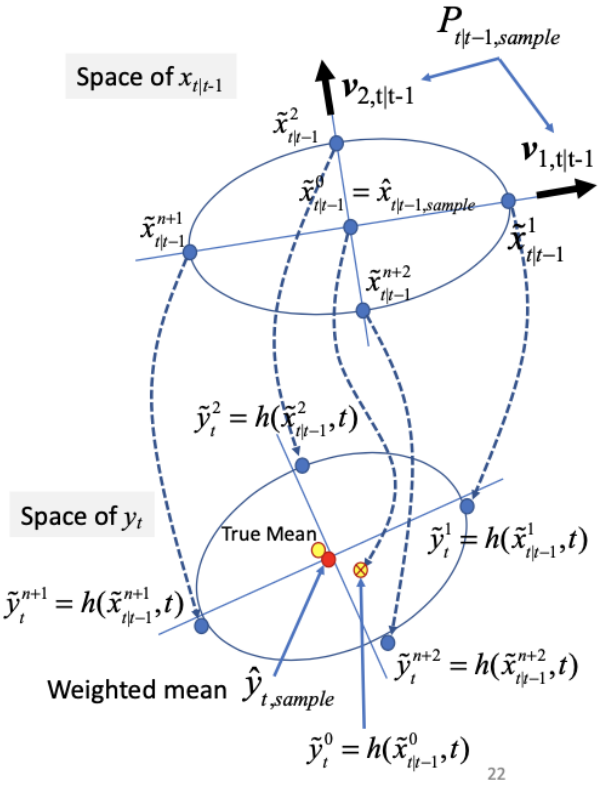

Figure 16: Unscented Propagation: 2 [16]

As for the covariance update, the covariance update formula includes measurement matrix H_t, which must be replaced. We can use the innovation covariance for this:

$$P_t = (I - K_t H_t)\, P_{t|t-1} = P_{t|t-1} - K_t H_t P_{t|t-1} = P_{t|t-1} - K_t P_y P_y^{-1} H_t P_{t|t-1}$$

$$\boxed{P_{t|t-1} - K_t P_y K_t^T},$$

where the Kalman Gain $K_t = P_{t|t-1} H_t^T P_y^{-1}$ is used to eliminate the measurement matrix H_t.

[16] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 9: Extended Kalman Filter and Unscented Kalman Filter.

And using the sigma points it can be computed as follows

$$K_t \approx P_{t|t-1,sample}\, H_t^T\, P_y^{-1}.$$

1.3.4 Higher Order Correlation Unscented Kalman Filter

Also it is worth noticing that there are advanced High Order Unscented Kalman Filters [17] , that can capture higher order moments (Skew, Kurtosis) and dependencies beyond linear correlation, but pay the cost of increased computational complexity. Essentially, we draw more Sigma Points than in the initial UKF ($2n+1 \Rightarrow 4n+1$):

$$\begin{aligned}
&\tilde{x}^0 = \bar{x}, \quad W_0 = \frac{\kappa - n}{\kappa + 3n} \\
&\tilde{x}^i = \bar{x} + \sqrt{n+\kappa} \cdot \sigma_i \cdot v_i, \quad W_i = \frac{1}{2(n+\kappa)} \\
&\tilde{x}^{i+n} = \bar{x} - \sqrt{n+\kappa} \cdot \sigma_i \cdot v_i, \quad W_{i+n} = \frac{1}{2(n+\kappa)} \\
&\tilde{x}^{i+2n} = \bar{x} + \sqrt{3(n+\kappa)} \cdot \sigma_i \cdot v_i, \quad W_{i+2n} = \frac{1}{6(n+\kappa)} \\
&\tilde{x}^{i+3n} = \bar{x} - \sqrt{3(n+\kappa)} \cdot \sigma_i \cdot v_i, \quad W_{i+3n} = \frac{1}{6(n+\kappa)}
\end{aligned}$$

The propagation step remains the same as in the initial UKF, but the update step is modified to capture higher order moments. In the HOC-UKF, the update step takes into account both linear and higher-order (for example, quadratic) correlations between the state and the observations. This is achieved by using an extended form of the Kalman Gain, which consists of two terms: a first-order gain $K^{(1)}$ and a second-order gain $K^{(2)}$. These terms allow the filter to capture dependencies that go beyond simple linear correlations, making it more accurate for nonlinear systems.

Given the propagated sigma points $\tilde{x}^i_{t|t-1}$ and predicted observation sigma points $\tilde{y}^i_{t|t-1}$, the update step proceeds. In addition to P_y and P_{xy}, we also compute P_{y^2} and P_{xy^2}.

$$\begin{cases} P_y = \mathbb{E}\big[\,\big(y_t - \hat{y}_t\big)\big(y_t - \hat{y}_t\big)^T\big], \\ P_{xy} = \mathbb{E}\big[\big(\hat{x}_{t|t-1} - x_t\big)\big(\hat{y}_t - y_t\big)^T\big] \end{cases} + \begin{cases} P_{y^2} = \mathbb{E}\Big[\big(y_t - \hat{y}_t\big)^2\,\big(\big(y_t - \hat{y}_t\big)^2\big)^T\Big], \\ P_{xy^2} = \mathbb{E}\Big[\big(\hat{x}_{t|t-1} - x_t\big)\,\big(\big(\hat{y}_t - y_t\big)^2\big)^T\Big] \end{cases}$$

Then we compute the Kalman Gains:

$$K^{(1)} = P^{(1)}_{xy}\, P_y^{-1}, \quad K^{(2)} = P^{(2)}_{xy^2}\, P_{y^2}^{-1}.$$

[17] Grothe, O., *A higher order correlation unscented Kalman filter*, University of Cologne, Department of Economic and Social Statistics, Albertus-Magnus-Platz, 50923 Köln, Germany.

Now we update the state mean and covariance:

$$\hat{x}_t = \hat{x}_{t|t-1,sample} + K_t^{(1)}\left[y_t - \hat{y}_{t,sample}\right] + K_t^{(2)}\left[y_t - \hat{y}_{t,sample}\right]^2,$$
$$P_{t|t} = P_{t|t-1} - K_t^{(1)} P_y \left(K_t^{(1)}\right)^T - K_t^{(2)} P_{y^2} \left(K_t^{(2)}\right)^T.$$

There also exist recursive formulas for computing the necessary terms, but it would take a long time and much space to derive them here. However, this gives the general idea.

1.3.5 Summary

1) Linearized Kalman Filter is simple, but performs poorly when the actual trajectory deviates from a nominal trajectory. The state transition and measurement are linear approximation, while the true system is nonlinear.

2) Extended Kalman Filter (EKF) tends to underestimate the error covariance, when the system is highly nonlinear. This sometimes causes the divergence of estimate.

3) The distribution of random variables after transformed through nonlinear equations, $f(x,t), h(x,t)$, is no longer Gaussian, although the original distribution was Gaussian. Unscented Kalman Filter approximates this distribution to a Gaussian and characterizes with mean and covariance. Although this approximation is accurate to the 2nd order, the discrepancy from a complete Gaussian may grow, as the process is repeated.

4) Higher Order Correlation Unscented Kalman Filter approximates this distribution to a Gaussian and characterizes with mean and covariance, but also makes sure to capture higher moments like Skew and Kurtosis and higher order correlations. Although this approximation is accurate to the 3nd order, the discrepancy from a complete Gaussian may grow, as the process is repeated.

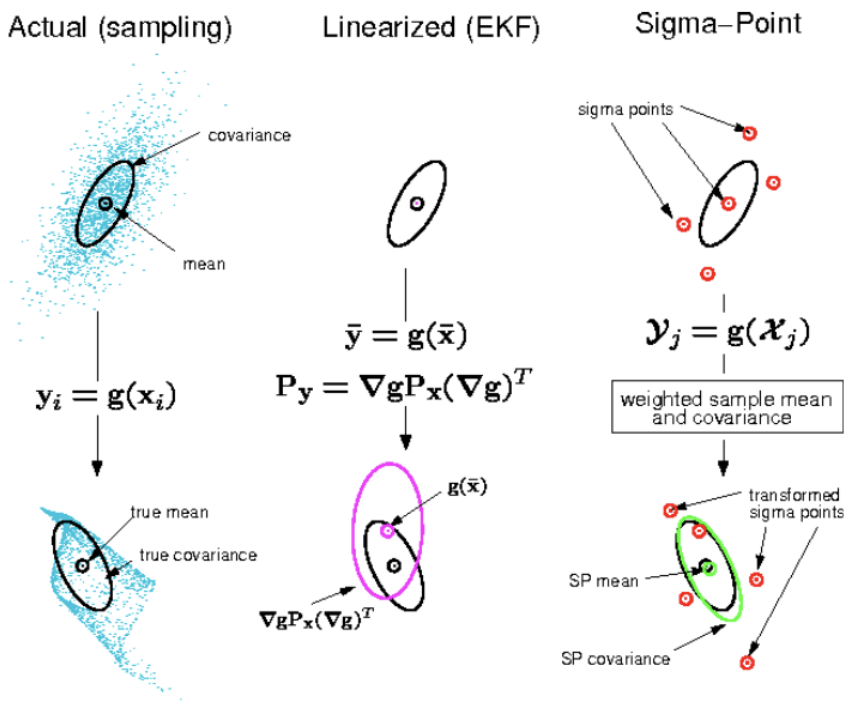

Figure 17: Filters Approximations [18]

1.4 Non-Linear Non-Gaussian Filters

Let us consider a distribution that cannot be fully described by only the mean and standard deviation, denoted as $g(x)$. Taking into account that x is a state vector, we still aim to predict it, even though the filters considered up to this point are capable of generating only point estimates. From now on, we will consider a distribution referred to as the "Belief," which we will predict instead of x.

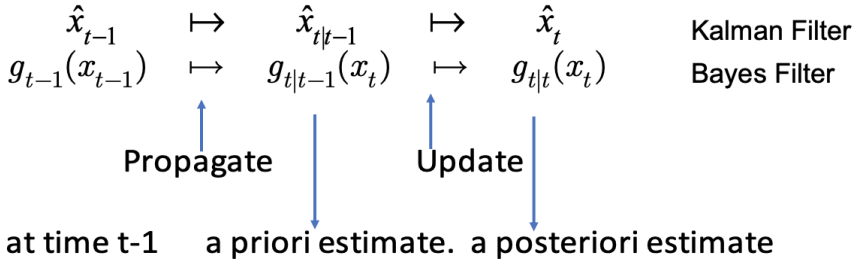

Figure 18: Kalman vs Bayes Filters [19]

[18] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 9: Extended Kalman Filter and Unscented Kalman Filter.
[19] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 10: Bayes Filter.

1.4.1 Bayesian Filter

Theoretically, this filter is very simple, but practically it can be challenging. Using the Chapman-Kolmogorov equation applied to the Markov process, we arrive at the propagation of the Belief:

$$\begin{cases} g_{t|t-1}(x_t) = \int_{-\infty}^{+\infty} \mathbb{P}(x_t \mid x_{t-1}, u_{t-1}) g_{t-1}(x_{t-1}) \mathrm{d}x_{t-1}, \\ x_t = f(x_{t-1}, u_{t-1}) + w_{t-1}, \\ p(w_{t-1}) = f_W(w_{t-1}), \end{cases}$$

which leads to:

$$\boxed{g_{t|t-1}(x_t) = \int_{-\infty}^{+\infty} f_W(x_t - f(x_{t-1}, u_{t-1})) g_{t-1}(x_{t-1}) \mathrm{d}x_{t-1}.}$$

Next, we turn to Bayes' formula for the Belief update:

$$\begin{cases} p(x \mid y) = \frac{p(y|x)p(x)}{p(y)} = \frac{1}{\eta} p(y \mid x) p(x), \\ y_t = h(x_t, t) + v_t, \\ p(v_t) = f_V(v_t), \end{cases}$$

which results in:

$$\boxed{p(x \mid y) = \frac{1}{\eta} f_V(y_t - h(x_t, t)) g_{t|t-1}(x_t).}$$

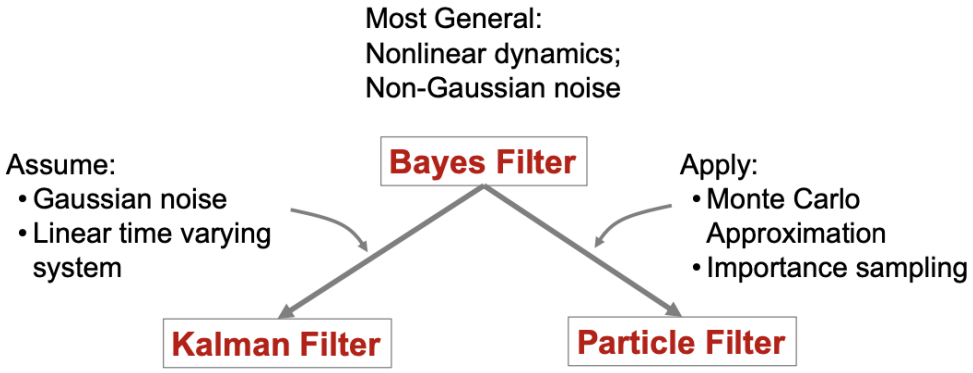

Figure 19: Bayes Hierarchy [20]

[20] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 10: Bayes Filter.

1.4.2 Particle Filter

Now we will dive into the non-parametric representation of probability density, where we draw samples with the probability density $f_X(x)$ and form a data set

$$\tilde{X} = \left\{x^{(1)}, x^{(2)}, \ldots, x^{(M)}\right\} \text{ (Particles)},$$

which estimates the empirical distribution

$$\hat{f}_X(\mathrm{d}x) = \frac{1}{M}\sum_{i=1}^{M} \delta_{x^{(i)}}(dx),$$

where $\delta_{x^{(i)}}$ denotes the Dirac delta mass located at $x^{(i)}$.

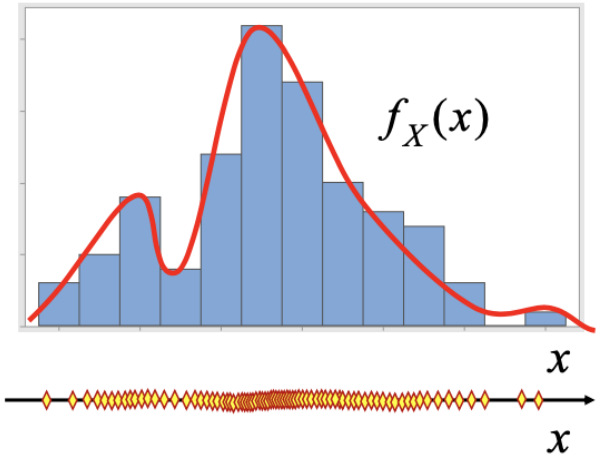

Figure 20: Particle representation of probability density. [21]

An important aspect of this method is the Monte Carlo approximation of integrals:

$$\mathbb{E}[h(x)] = \int_{-\infty}^{+\infty} h(x) f_X(x) \mathrm{d}x \approx \frac{1}{M}\sum_{i=1}^{M} h(x^{(i)}),$$

where $\left\{x^{(1)}, x^{(2)}, \ldots, x^{(M)}\right\}$ are particles drawn from $f_X(x)$.

For generating particles, another well-known method is the CDF method. In this method, we draw samples $y^{(i)}$ and, by solving the equation $F(x^{(i)}) = y^{(i)}$, convert uniformly distributed samples into samples from the desired distribution, as long as we know how to invert the CDF or numerically solve the equation.

We are now ready for the propagation step. Instead of directly computing the Chapman-

[21] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 12: Particle Filter.

Kolmogorov equation:

$$g_{t|t-1}(x_t) = \int_{-\infty}^{+\infty} \mathbb{P}(x_t \mid x_{t-1}, u_{t-1}) g_{t-1}(x_{t-1}) \mathrm{d}x_{t-1},$$

we generate M particles representing $g_{t|t-1}(x_t)$ from $g_{t-1}(x_{t-1})$. First, we draw M samples of $w_{t-1}^{(i)}$ for each particle $x_{t-1}^{(i)}$ using the CDF method. We then propagate to a new set of particles:

$$x_{t|t-1}^{(i)} = f(x_{t-1}^{(i)}, u_{t-1}) + w_{t-1}^{(i)}.$$

This serves as an analogue to $\mathbb{P}(x_t \mid x_{t-1}, u_{t-1})$, allowing us to make the propagation step:

$$\tilde{X}_{t-1} = \left\{ x_{t-1}^{(1)}, x_{t-1}^{(2)}, \ldots, x_{t-1}^{(M)} \right\} \Rightarrow \tilde{X}_{t|t-1} = \left\{ x_{t|t-1}^{(1)}, x_{t|t-1}^{(2)}, \ldots, x_{t|t-1}^{(M)} \right\}.$$

For the update step:

$$p(x \mid y) = \frac{1}{\eta} f_V(y_t - h(x_t, t)) g_{t|t-1}(x_t),$$

Monte Carlo approximation cannot be used. Instead, we employ the Importance Sampling technique.

Consider two PDFs $f(x)$ and $g(x)$, where samples can be drawn from $g(x)$ but not from $f(x)$. Assuming the measures corresponding to $f(x)$ and $g(x)$ are equivalent, we can represent:

$$f(x) = \frac{f(x)}{g(x)} g(x),$$

where $W(x) = \frac{f(x)}{g(x)}$ are the Importance Weights and $g(x)$ is the Importance Density.

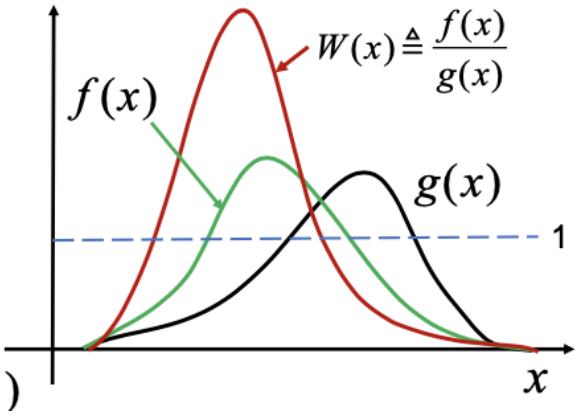

Figure 21: Importance Sampling[22]

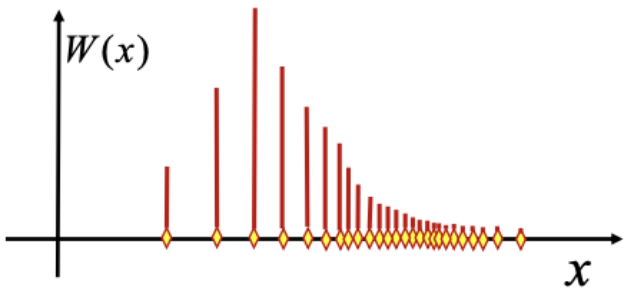

Figure 22: Importance Weights[22]

Applying Importance Sampling to the update step, where samples from $g_{t|t-1}(x_t)$ have al-

[22] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 12: Particle Filter.

ready been generated in the form of particles from the propagation step $\tilde{X}_{t|t-1} = \{x_{t|t-1}^{(1)}, x_{t|t-1}^{(2)}, \ldots, x_{t|t-1}^{(M)}\}$, we observe that the CDF of $g_t(x_t)$ can be expressed as:

$$G_t(x_t) = \int_{-\infty}^{x_t} g_t(x)\mathrm{d}x \approx \frac{1}{W_0}\sum_{i=1}^{M} f_V(y_t - h(x_{t|t-1}^{(i)}))\mathbb{I}_{\{x_{t|t-1}^{(i)} < x_t\}},$$

or equivalently:

$$\boxed{G_t(x_t) = \frac{1}{W_0}\sum_{i=1}^{M} W(x_{t|t-1}^{(i)}; y_t)\mathbb{I}_{\{x_{t|t-1}^{(i)} < x_t\}},}$$

where $W_0 = \sum_{i=1}^{M} f_V(y_t - h(x_{t|t-1}^{(i)}))$.

From G_t, we can sample M new particles $\tilde{X}_t = \{x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(M)}\}$ using the CDF method discussed earlier. Furthermore, weights can be updated recursively (proof omitted), which defines the Sequential Importance Sampling (SIS) algorithm:

$$\boxed{W_t^{(i)} \propto W_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)})p(x_t^{(i)} \mid x_{t-1}^{(i)})}{g_{t-1}(x_t^{(i)} \mid x_{t-1}^{(i)})}.}$$

A well-known issue with the SIS particle filter is the degeneracy phenomenon. After a few iterations, all but one particle will have negligibly small weights. It can be proven that the variance of weights only increases over time. To address this, before the variance of weights becomes too large, particles with small weights are removed, and new M particles are resampled with replacement based on the Importance Weights of the surviving particles. This process is called Sequential Importance Resampling (SIR).

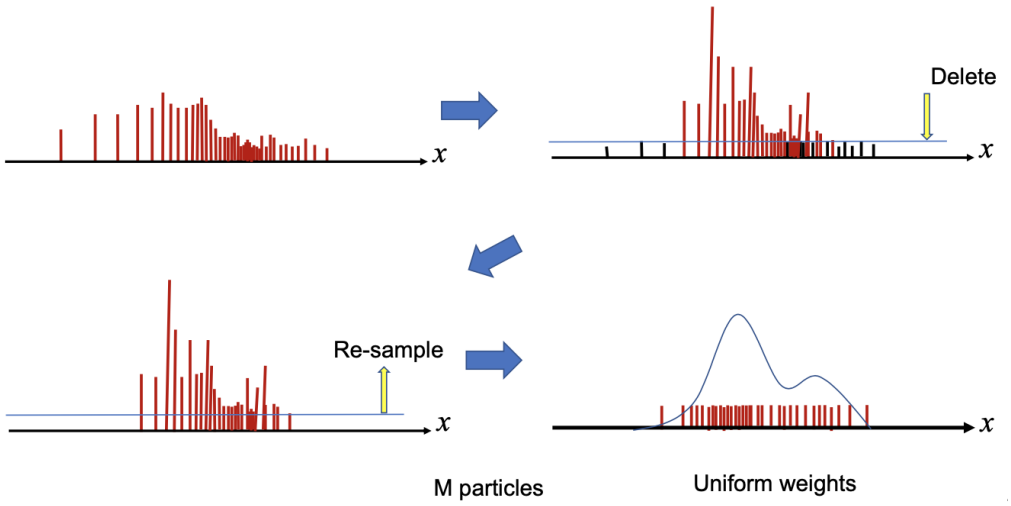

Figure 23: Sequential Importance Resampling (SIR) [23]

In summary, this breakdown describes Sequential Monte Carlo Filters, which are generally

[23] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 12: Particle Filter.

characterized by the following algorithm:

A Sequential Monte Carlo Algorithm

At time $t = 0$,

Step 0: *Initialisation*

- For $i = 1, ..., N$, sample $x_0^{(i)} \sim \pi_0(dx_0)$ and set $t = 1$.

At time $t \geq 1$,

Step 1: *Importance Sampling step*

- For $i = 1, ..., N$, sample $\widetilde{x}_{0:t}^{(i)} \sim q_t\left(d\widetilde{x}_{0:t} | \, x_{0:t-1}^{(i)}\right)$.
- For $i = 1, ..., N$, evaluate the importance weights $w_{0.t}^{(i)}$ using (4.3.14).

Step 2: *Selection step*

- Multiply/Discard particles $\left(\widetilde{x}_{0:t}^{(i)}; i = 1, \ldots, N\right)$ with respect to high/low normalised importance weights $w_{0:t}^{(i)}$ to obtain N equally weighted particles $\left(\ddot{x}_{0:t}^{(i)}; i = 1, \ldots, N\right)$.

Step 3: *Markov transition step*

- For $i = 1, ..., N$, sample $x_{0:t}^{(i)} \sim \Xi_t\left(dx_{0\,t} | \, \ddot{x}_{0:t}^{(i)}\right)$ where $\Xi_t(\cdot | \cdot)$ is a transition kernel of invariant distribution $\pi_t(dx_{0:t})$.
- Set $t \leftarrow t + 1$ and go to **Step 1**.

Figure 24: Sequential Monte Carlo (SMC) Algorithm [24]

(Not sure anbout correctness !!!)

It is worth noting that even though we are now working with non-Gaussian and non-linear systems, we still need to determine the Markov Kernel in the Markov Transition Step. There are many ways to achieve this, one of which is using Markov Chain Monte Carlo (MCMC) methods. Examples include Langevin Dynamics, Hamiltonian Dynamics, and Metropolis-Hastings.

2 Calibration

Usually, we do not know the parameters θ, which means that we need to estimate them along with $p(x_t, \theta \mid y_{1:t})$.

State-of-the-art methods include:

1. Particle Markov chain Monte Carlo (PMCMC),
2. Sequential Monte Carlo squared (SMC^2),

[24] H. Harry Asada, *Identification, Estimation, and Learning* (MIT Postgraduate Course 2.160, 2020–2021), Lecture 12: Particle Filter.

3. Nested Particle Filters (NPF).

It is worth noting that both PMCMC and SMC^2 are batch techniques, while NPF is a recursive method.

Our goal is to compute the joint posterior PDF:

$$p(x_t, \theta \mid y_{1:t}) = \underbrace{p(x_t \mid \theta, y_{1:t})}_{\text{2nd layer}} \underbrace{p(\theta \mid y_{1:t})}_{\text{1st layer}}$$

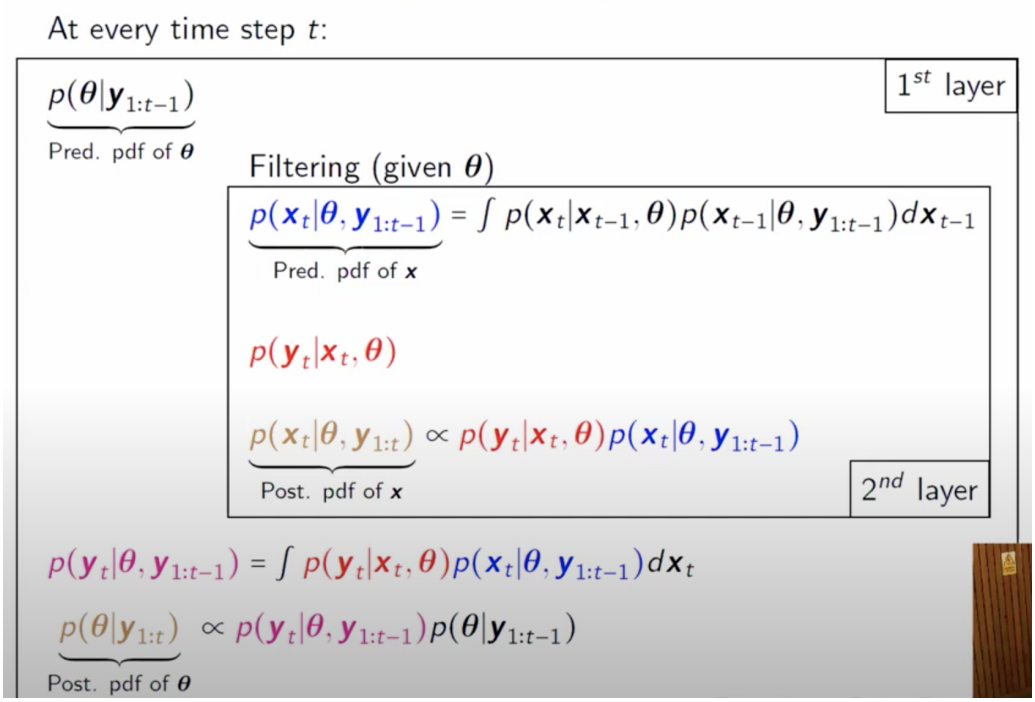

Figure 25: Double-layer structure of the posterior PDF [25]

2.1 Nested Filters

To address the problems that may occur with this layout, we will start with the most basic idea of following this algorithm, which is illustrated as follows:

[25] Sara Perez Vieites, *Learning the number of particles in nested filtering*, presented at the 6th Workshop on Sequential Monte Carlo Methods (SMC 2024).

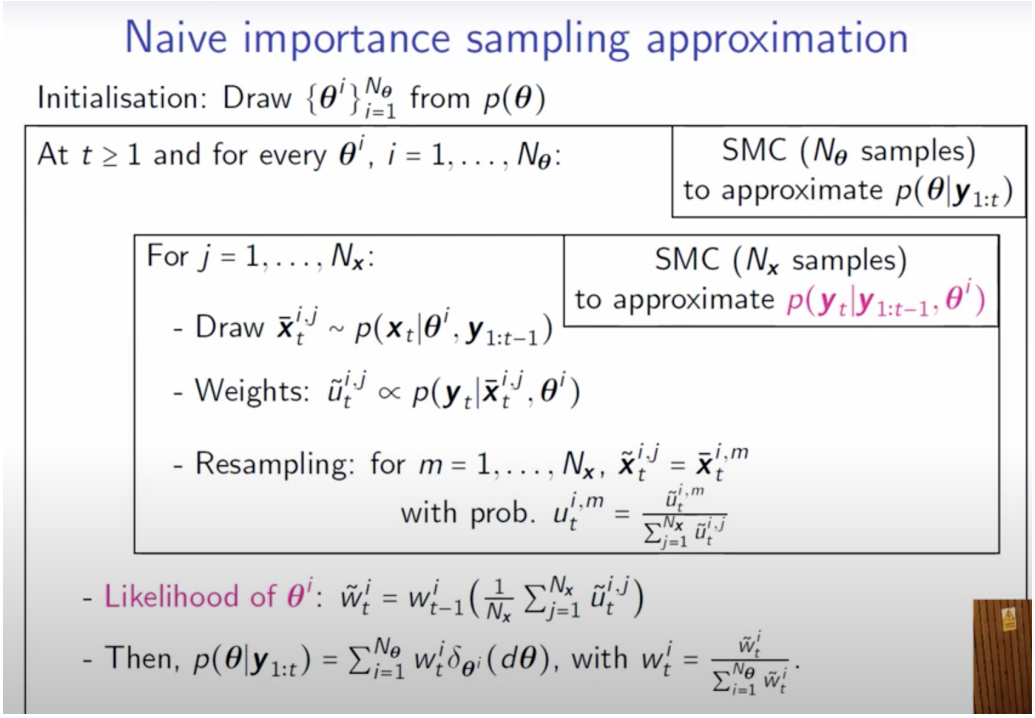

Figure 26: Naive Double-Layer Algorithm [26]

Be careful with $p(\theta)$: after several time steps, the filter degenerates as $p(\theta \mid y_{1:t})$ changes. A possible solution is to draw samples $\theta_t^i \sim p(\theta \mid y_{1:t-1})$ at each time step, re-running the filter for x from scratch (non-recursive approach).

To address the degeneracy issue, we introduce a jittering step in NPF:

$$\tilde{\theta}_t^i \sim \kappa_{N_\theta}(d\theta \mid \theta'),$$

where:

$$\begin{cases} \boxed{\kappa_{N_\theta}(d\theta \mid \theta') = (1 - \epsilon_{N_\theta})\delta_{\theta'}(\theta) + \epsilon_{N_\theta}\kappa_{N_\theta}(d\theta \mid \theta')} \\ 0 < \epsilon_{N_\theta} \leq \frac{1}{\sqrt{N_\theta}}, \\ e.g. \quad \kappa_{N_\theta}(d\theta \mid \theta') = \mathcal{N}(\theta \mid \theta', \sigma^2 I), \end{cases}$$

indicating that the Markov kernel is actually arbitrary.

[26] Sara Perez Vieites, *Learning the number of particles in nested filtering*, presented at the 6th Workshop on Sequential Monte Carlo Methods (SMC 2024).

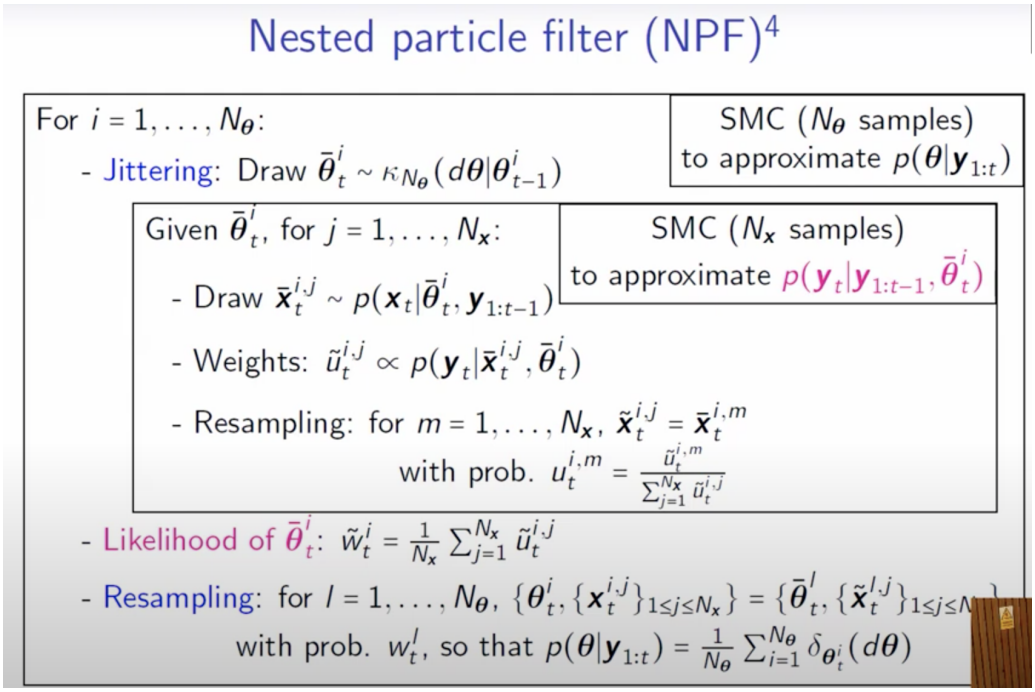

Figure 27: Nested Particle Filter (NPF) [27]

Here are three nested filtering approaches:

1. Nested Particle Filters (NPFs):
   - Both layers → Sequential Monte Carlo (SMC) methods.
   - High computational complexity: $N_\theta \times N_x$.
2. Nested Hybrid Filters (NHFs):
   - θ-layer → Monte Carlo-based methods (e.g., SMC or SQMC).
   - x-layer → Gaussian techniques (e.g., EKFs or UKFs).
3. Nested Gaussian Filters (NGFs):
   - θ-layer → Deterministic sampling methods (e.g., UKF).
   - x-layer → Gaussian techniques (e.g., EKFs or UKFs).

2.2 Particle Markov Chain Monte Carlo

Let us once again consider the full probabilistic model describing the joint distribution of observation variables, latent variables, and parameters:

[27] Sara Perez Vieites, *Learning the number of particles in nested filtering*, presented at the 6th Workshop on Sequential Monte Carlo Methods (SMC 2024).

$$p(\theta, x_{0:T}, y_{1:T}) = \underbrace{p(y_{1:T} \mid x_{0:T}, \theta)}_{\text{data distribution}} \underbrace{p(x_{0:T} \mid \theta) p(\theta)}_{\text{prior}} = \underbrace{\prod_{t=1}^{T} \underbrace{p(y_t \mid x_t, \theta)}_{\text{observation}}}_{\text{data distribution}} \underbrace{\prod_{t=1}^{T} \underbrace{p(x_t \mid x_{t-1}, \theta)}_{\text{dynamics}} \underbrace{p(x_0 \mid \theta)}_{\text{state}} \underbrace{p(\theta)}_{\text{params}}}_{\text{prior}}$$

Bayesian inference involves approximating the target distribution $\pi(x)$, which could represent only the parameters, $x = \theta$, with $\pi(x) = p(\theta \mid y_{1:T})$, or the joint distribution of parameters and latent variables, $\pi(x, \theta) = p(x_{0:T}, \theta \mid y_{1:T})$.

Two approaches can address this problem: 1. Using a parametric approach to fit the chosen distribution for parameters. 2. Constructing a Markov kernel such that its limiting distribution coincides with the target distribution.

Here, we focus on the second approach.

The main requirements for the Markov kernel $\kappa(x^* \mid x)$, a conditional distribution for the next state x^* given the current state x, are:

1. Stationarity: The kernel should admit π as a stationary distribution:

$$\int \pi(x) \kappa(x^* \mid x) \mathrm{d}x = \pi(x^*).$$

2. Ergodicity: The kernel should ensure that the chain reaches the stationary distribution regardless of the initial distribution:

$$\forall (x^* \mid x) : \ \kappa(x^* \mid x) > 0.$$

The properties of MCMC approximations, using the MCMC sample $x[m+1] \sim \kappa(x \mid x[m])$, for $m = \overline{1, M-1}$, include:

1. Ergodic Theorem: For an ergodic chain, the MCMC estimator is consistent:

$$\frac{1}{M} \sum_{m=1}^{M} \varphi(x[m]) \underset{M \to \infty}{\overset{a.s.}{\to}} \int \varphi(x) \pi(x) \mathrm{d}x.$$

2. Geometric Ergodicity: For a geometrically ergodic chain with state space S and stationary distribution π, there exist constants $C > 0$ and $r \in (0, 1)$ such that:

$$\sup_{x \in S} \| P^n(x, \cdot) - \pi(\cdot) \|_{\mathrm{TV}} \leq C r^n,$$

where: - $P^n(x, \cdot)$ is the n-step transition probability starting from state x, - $\pi(\cdot)$ is the stationary distribution, - $\| \cdot \|_{\mathrm{TV}}$ denotes the total variation norm.

Intuitively, this means that the chain exponentially forgets its initial distribution. For such

Markov chains:

$$\sqrt{M}\left(\frac{1}{M}\sum_{m=1}^{M}\varphi(x[m]) - \mathbb{E}_\pi[\varphi(X)]\right) \underset{M\to\infty}{\overset{d}{\to}} \mathcal{N}\left(0, \sigma_\infty^2(\varphi)\right),$$

where:

$$\sigma_\infty^2(\varphi) = \mathrm{Var}_\pi[\varphi(X)]\left(1 + 2\sum_{\ell=1}^{\infty}\mathrm{Corr}(\varphi(X_\ell), \varphi(X_0))\right).$$

The term IACT (Integrated Autocorrelation Time) is commonly used in practice to represent this summation.

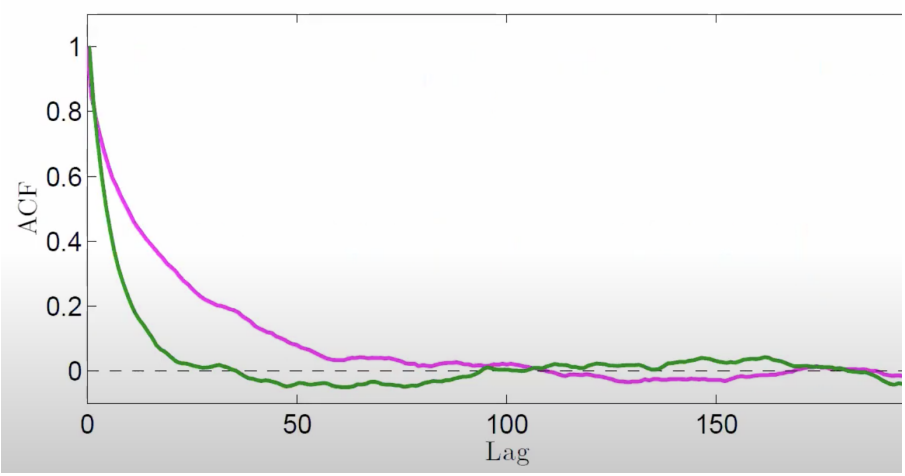

Figure 28: Integrated Autocorrelation Time (IACT) [28]

We now build a pipeline for approximating the target distribution and solving the encountered challenges:

1. Using Metropolis-Hastings (MH) Markov Kernel.

[28] Fredrik Lindsten, *Particle Markov chain Monte Carlo*, presented at the Summer School on Bayesian Filtering: Fundamental Theory and Numerical Methods (SSBF 2024).

Algorithm 1 Metropolis Hastings (MH)

1. **Initialize:** Set the initial state of the Markov chain $x[1]$.
2. **For** $m = 1$ **to** M**, iterate:**
   a. Sample $x' \sim q(x \mid x[m])$.
   b. Sample $u \sim \mathcal{U}[0, 1]$.
   c. Compute the acceptance probability
   $$\alpha = \min\left(1, \frac{\pi(x')}{\pi(x[m])} \frac{q(x[m] \mid x')}{q(x' \mid x[m])}\right)$$
   d. Set the next state $x[m+1]$ of the Markov chain according to
   $$x[m+1] = \begin{cases} x' & u \leq \alpha \\ x[m] & \text{otherwise} \end{cases}$$

Figure 29: Metropolis-Hastings Algorithm [29]

2. Considering the likelihood in acceptance ratio for Linear Gaussian dynamical systems:

$$p(y_{1:T} \mid \theta) = \prod_{t=1}^{T} \mathcal{N}(y_t \mid \hat{y}_t(\theta), S_t(\theta)).$$

3. Introducing storage for computational efficiency:

Algorithm 3 Metropolis–Hastings for LG-SSM

1. **Initialize** $(m = 1)$**:** Set $\theta[1]$ and run a Kalman filter to compute $\hat{z}[1] = p(y_{1:T} \mid \theta[1])$.
2. **For** $m = 2$ **to** M**, iterate:**
   a. Sample $\theta' \sim q(\theta \mid \theta[m])$.
   b. Compute $\hat{z}' = p(y_{1:T} \mid \theta')$ (run a Kalman filter and compute likelihood)
   c. With probability
   $$\alpha = \min\left(1, \frac{\hat{z}'}{\hat{z}[m]} \frac{p(\theta')}{p(\theta[m])} \frac{q(\theta[m] \mid \theta')}{q(\theta' \mid \theta[m])}\right)$$
   set $\{\theta[m+1], \hat{z}[m+1]\} \leftarrow \{\theta', \hat{z}'\}$ (accept candidate sample) and with prob. $1 - \alpha$ set $\{\theta[m+1], \hat{z}[m+1]\} \leftarrow \{\theta[m], \hat{z}[m]\}$ (reject candidate sample).

Figure 30: Modified Metropolis-Hastings Algorithm with Storage [29]

4. Generalizing to Non-Linear State Space Models with Particle Filters:

[29] Fredrik Lindsten, *Particle Markov chain Monte Carlo*, presented at the Summer School on Bayesian Filtering: Fundamental Theory and Numerical Methods (SSBF 2024).

Algorithm 4 Bootstrap particle filter (for $i = 1, \ldots, N$)

1. **Initialization** ($t = 0$):
   (a) Sample $x_0^i \sim p(x_0 \mid \theta)$.
   (b) Set initial weights: $w_0^i = 1/N$.
2. **for** $t = 1$ **to** T **do**
   (a) Resample: sample ancestor indices $a_t^i \sim \mathcal{C}(\{w_{t-1}^j\}_{j=1}^N)$.
   (b) Propagate: sample $x_t^i \sim p(x_t \mid x_{t-1}^{a_t^i}, \theta)$.
   (c) Weight: compute $\tilde{w}_t^i = p(y_t \mid x_t^i, \theta)$ and normalize $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$.

Figure 31: Particle Filter for Non-Linear State Space Models [30]

An estimate of the marginal likelihood obtained as a byproduct is

$$p(y_{1:T} \mid \theta) \approx \hat{z} = \prod_{t=1}^{T} \Big(\tfrac{1}{N} \sum_{i=1}^{N} \tilde{w}_t^i \Big),$$

whose distribution can actually be explicitly written by considering the steps of the algorithm.

A particle filter that runs for time steps $t = 0, \ldots, T$ samples the random variables:

$$X_t = \{X_t^i\}_{i=1}^N, \quad t = 0, \ldots, T, \quad A_t = \{A_t^i\}_{i=1}^N, \quad t = 1, \ldots, T,$$

with the following distributions for the bootstrap particle filter:

1) Initialization:

$$X_0 \sim \prod_{i=1}^{N} p(x_0^i \mid \theta).$$

2) Resampling:

$$A_t \mid (X_{t-1} = x_{t-1}) \sim \prod_{i=1}^{N} W_{t-1}^{a_t^i}.$$

3) Propagation:

$$X_t \mid (X_{t-1} = x_{t-1},\, A_t = a_t) \sim \prod_{i=1}^{N} p\big(x_t^i \mid x_{t-1}^{a_t^i}, \theta\big).$$

Let $X_{0:T} = (X_0, \ldots, X_T)$ and $A_{1:T} = (A_1, \ldots, A_T)$.

The distribution of all the random variables sampled by the bootstrap particle filter is thus:

$$\boxed{\psi(X_{0:T}, A_{1:T} \mid \theta) = \Big\{ \prod_{i=1}^{N} p(x_0^i \mid \theta) \Big\} \prod_{t=1}^{T} \Big\{ \prod_{i=1}^{N} w_{t-1}^{a_t^i}\, p\big(x_t^i \mid x_{t-1}^{a_t^i}, \theta\big) \Big\}}$$

[30] Fredrik Lindsten, *Particle Markov chain Monte Carlo*, presented at the Summer School on Bayesian Filtering: Fundamental Theory and Numerical Methods (SSBF 2024).

with domain $\mathcal{X}^{N(T+1)} \times \{1, \ldots, N\}^{NT}$.

Therefore, executing a particle filter can be viewed as drawing one sample from this distribution. Let us consider the properties of that sample with respect to the initial likelihood $p(y_{1:T} \mid \theta)$ that we are trying to estimate. We denote the distribution of $\hat{Z}$ by $\psi(\hat{z} \mid \theta)$:

1) The likelihood estimator $\hat{Z}$ is unbiased for any number of particles $N \geq 1$ for all auxiliary particle filters (which we will discuss later), not only the bootstrap particle filter:

$$\mathbb{E}_{\psi(\hat{z}|\theta)}[\hat{Z}] = p\big(y_{1:T} \mid \theta\big).$$

Algorithm 6 Particle Metropolis–Hastings

1. **Initialize** ($m = 1$)**:** Set $\theta[1]$ and run a particle filter to compute $\hat{z}[1]$.
2. **For** $m = 2$ **to** M**, iterate:**
   a. Sample $\theta' \sim q(\theta \mid \theta[m])$.
   b. Sample $\hat{z}' \sim \psi(\hat{z} \mid \theta')$ (run a particle filter and compute likelihood estimate)
   c. With probability

$$\alpha = \min\left(1, \frac{\hat{z}'}{\hat{z}[m]} \frac{p(\theta')}{p(\theta[m])} \frac{q(\theta[m] \mid \theta')}{q(\theta' \mid \theta[m])}\right)$$

   set $\{\theta[m+1], \hat{z}[m+1]\} \leftarrow \{\theta', \hat{z}'\}$ (accept candidate sample) and with prob. $1 - \alpha$ set $\{\theta[m+1], \hat{z}[m+1]\} \leftarrow \{\theta[m], \hat{z}[m]\}$ (reject candidate sample).

Figure 32: Modified Metropolis-Hastings Algorithm with Particles [31]

Now the important difference from the Kalman filter is that the likelihood is not a deterministic function of the parameters θ, but rather a random variable. Therefore, we cannot use exactly the same approach as in the Kalman filter, where we would recompute the likelihood for each proposal and store it for acceptance. Here we jointly accept or reject the pair $\{\theta, \hat{z}\}$.

For this to be valid, the estimate of the likelihood must be non-negative and unbiased. Using such an estimate within the Metropolis-Hastings algorithm is called the pseudo-marginal approach.

To further illustrate the concept of auxiliary variables, consider a simple case:

Target distribution: $\pi(x)$, which is difficult to sample from.

Idea: Introduce another variable U with conditional distribution $\pi(u \mid x)$.

The joint distribution $\pi(x, u) = \pi(u \mid x)\,\pi(x)$ admits $\pi(x)$ as a marginal by construction, that is, $\int \pi(x, u)\,\mathrm{d}u = \pi(x)$.

Sampling from the joint $\pi(x, u)$ may be easier than directly sampling from the marginal $\pi(x)$.

The variable U is an auxiliary variable. It may have some physical interpretation; in our case

[31] Fredrik Lindsten, *Particle Markov chain Monte Carlo*, presented at the Summer School on Bayesian Filtering: Fundamental Theory and Numerical Methods (SSBF 2024).

it is the latent particles X that help to estimate the posterior distribution of the parameters θ, going from $p(\theta \mid y_{1:T})$ to $p(\theta, x_{1:T} \mid y_{1:T})$. Usually, after the sample pair $\{\theta, \hat{z}\}$ is drawn, the auxiliary variable is dropped (or kept until the entire time-spread sample is drawn), but in our case we may also be interested in the filtration results as well as in the calibration results.

Why do we need the auxiliary variable to be non-negative and unbiased? Because:

Target $\pi(\theta) = \dfrac{p(\theta \mid y_{1:T})\, p(\theta)}{p(y_{1:T})}$. The likelihood is intractable, but $p(y_{1:T} \mid \theta) = \mathbb{E}_{\psi(\hat{z}|\theta)}[\hat{Z}]$ (unbiased). We can substitute it into Bayes' formula for the target distribution:

$$\pi(\theta) = \int \frac{\hat{Z}\, \psi(\hat{z} \mid \theta)\, p(\theta)}{p(y_{1:T})}\, \mathrm{d}\hat{z} \quad \stackrel{\text{def}}{=} \quad \pi(\theta, \hat{z}),$$

provided $\hat{z}$ is non-negative.

We can explicitly see why we proposed this in Algorithm 6. Given a proposal $q(\theta' \mid \theta[m])$, we construct a joint proposal for $\{\theta, \hat{z}\}$ as:

$$\begin{aligned}
&q(\theta', \hat{z}' \mid \theta[m]) = \psi(\hat{z}' \mid \theta')\, q(\theta' \mid \theta[m]) \implies \\
&\alpha = \min\Big(1,\ \frac{\pi(\theta', \hat{z}')}{\pi(\theta[m], \hat{z}[m])}\, \frac{q(\theta[m], \hat{z}[m] \mid \theta')}{q(\theta', \hat{z}' \mid \theta[m])}\Big) \\
&\quad = \min\Big(1,\ \frac{\hat{z}'\, \psi(\hat{z}' \mid \theta')\, p(\theta')}{p(y_{1:T})}\, \frac{p(y_{1:T})\, \hat{z}[m]\, \psi(\hat{z}[m] \mid \theta[m])\, p(\theta[m])}{\hat{z}[m]\, \psi(\hat{z}[m] \mid \theta[m])\, q(\theta[m] \mid \theta')\, \psi(\hat{z}' \mid \theta')\, q(\theta' \mid \theta[m])}\Big) \\
&\quad = \min\Big(1,\ \frac{\hat{z}'}{\hat{z}[m]}\, \frac{p(\theta')}{p(\theta[m])}\, \frac{q(\theta[m] \mid \theta')}{q(\theta' \mid \theta[m])}\Big).
\end{aligned}$$