Learning Outcomes

1. Principles of Bayesian inference in dynamic systems
2. Construction of probabilistic state space models
3. Bayesian filtering of state space models
4. Bayesian smoothing of state space models
5. Parameter estimation in state space models
Recursive Estimation of Dynamic Processes

- **Dynamic**, that is, time varying phenomenon - e.g., the motion state of a car or a smart phone.

- The phenomenon is measured - for example by a radar or by acceleration and angular velocity sensors.

- The purpose is to compute the state of the phenomenon when only the measurements are observed.

- The solution should be recursive, where the information in new measurements is used for updating the old information.
The laws of physics, biology, epidemiology etc. are typically differential equations.

Uncertainties and unknown sub-phenomena are modeled as stochastic processes:

- **Physical phenomena**: differential equations + uncertainty $\Rightarrow$ stochastic differential equations.
- **Discretized physical phenomena**: Stochastic differential equations $\Rightarrow$ stochastic difference equations.
- **Naturally discrete-time phenomena**: Systems jumping from step to another.

Stochastic differential and difference equations can be represented in stochastic state space form.
Bayesian Modeling of Measurements

- The relationship between measurements and phenomenon is mathematically modeled as a probability distribution.
- The measurements could be (in ideal world) computed if the phenomenon was known (forward model).
- The uncertainties in measurements and model are modeled as random processes.
- The measurement model is the conditional distribution of measurements given the state of the phenomenon.
Consider the linear regression model

\[ y_k = \theta_1 + \theta_2 t_k + \varepsilon_k, \quad k = 1, \ldots, T, \]

with \( \varepsilon_k \sim N(0, \sigma^2) \) and \( \theta = (\theta_1, \theta_2) \sim N(m_0, P_0) \).

In probabilistic notation this is:

\[
p(y_k | \theta) = N(y_k | H_k \theta, \sigma^2) \]
\[
p(\theta) = N(\theta | m_0, P_0),
\]

where \( H_k = (1 \ t_k) \).
The **Bayesian batch solution** by the Bayes’ rule:

\[
p(\theta \mid y_{1:T}) \propto p(\theta) \prod_{k=1}^{T} p(y_k \mid \theta) = N(\theta \mid m_0, P_0) \prod_{k=1}^{T} N(y_k \mid H_k \theta, \sigma^2)
\]

The posterior is Gaussian

\[
p(\theta \mid y_{1:T}) = N(\theta \mid m_T, P_T).
\]

The **mean and covariance** are given as

\[
m_T = \left[ P_0^{-1} + \frac{1}{\sigma^2} H^T H \right]^{-1} \left[ \frac{1}{\sigma^2} H^T y + P_0^{-1} m_0 \right]
\]

\[
P_T = \left[ P_0^{-1} + \frac{1}{\sigma^2} H^T H \right]^{-1},
\]

where \( H_k = (1 \ t_k), \ H = (H_1; H_2; \ldots; H_T), \ y = (y_1; \ldots; y_T). \)
Assume that we have already computed the posterior distribution, which is conditioned on the measurements up to \(k - 1\):

\[
p(\theta \mid y_{1:k-1}) = N(\theta \mid m_{k-1}, P_{k-1}).
\]

Assume that we get the \(k\)th measurement \(y_k\). Using the equations from the previous slide we get

\[
p(\theta \mid y_{1:k}) \propto p(y_k \mid \theta) p(\theta \mid y_{1:k-1})
\]

\[
\propto N(\theta \mid m_k, P_k).
\]

The mean and covariance are given as

\[
m_k = \left[ P_{k-1}^{-1} + \frac{1}{\sigma^2} H_k^T H_k \right]^{-1} \left[ \frac{1}{\sigma^2} H_k^T y_k + P_{k-1}^{-1} m_{k-1} \right]
\]

\[
P_k = \left[ P_{k-1}^{-1} + \frac{1}{\sigma^2} H_k^T H_k \right]^{-1}.
\]
By the **matrix inversion lemma** (or Woodbury identity):

\[
P_k = P_{k-1} - P_{k-1} H_k^T \left[ H_k P_{k-1} H_k^T + \sigma^2 \right]^{-1} H_k P_{k-1}.
\]

Now the equations for the **mean and covariance** reduce to

\[
S_k = H_k P_{k-1} H_k^T + \sigma^2
\]

\[
K_k = P_{k-1} H_k^T S_k^{-1}
\]

\[
m_k = m_{k-1} + K_k [y_k - H_k m_{k-1}]
\]

\[
P_k = P_{k-1} - K_k S_k K_k^T.
\]

Computing these for \( k = 0, \ldots, T \) gives exactly the linear regression solution.

A special case of **Kalman filter**.
Recursive Linear Regression [3/4]
Convergence of the recursive solution to the batch solution – on the last step the solutions are exactly equal:
General batch solution:
- Specify the measurement model:

\[ p(y_{1:T} | \theta) = \prod_{k} p(y_k | \theta). \]

- Specify the prior distribution \( p(\theta) \).
- Compute posterior distribution by the Bayes’ rule:

\[ p(\theta | y_{1:T}) = \frac{1}{Z} p(\theta) \prod_{k} p(y_k | \theta). \]

- Compute point estimates, moments, predictive quantities etc. from the posterior distribution.
**General recursive solution:**

- Specify the **measurement likelihood** \( p(y_k \mid \theta) \).
- Specify the **prior distribution** \( p(\theta) \).
- Process measurements \( y_1, \ldots, y_T \) **one at a time**, starting from the prior:

\[
p(\theta \mid y_1) = \frac{1}{Z_1} p(y_1 \mid \theta)p(\theta)
\]

\[
p(\theta \mid y_{1:2}) = \frac{1}{Z_2} p(y_2 \mid \theta)p(\theta \mid y_1)
\]

\[
p(\theta \mid y_{1:3}) = \frac{1}{Z_3} p(y_3 \mid \theta)p(\theta \mid y_{1:2})
\]

\[\vdots\]

\[
p(\theta \mid y_{1:T}) = \frac{1}{Z_T} p(y_T \mid \theta)p(\theta \mid y_{1:T-1}).
\]

- The result at the last step is the **batch solution**.
Advantages of Recursive Solution

- The recursive solution can be considered as the online learning solution to the Bayesian learning problem.
- Batch Bayesian inference is a special case of recursive Bayesian inference.
- The parameter can be modeled to change between the measurement steps ⇒ basis of filtering theory.
Let assume **Gaussian random walk** between the measurements in the linear regression model:

\[
p(y_k | \theta_k) = \mathcal{N}(y_k | H_k \theta_k, \sigma^2)
\]

\[
p(\theta_k | \theta_{k-1}) = \mathcal{N}(\theta_k | \theta_{k-1}, Q)
\]

\[
p(\theta_0) = \mathcal{N}(\theta_0 | m_0, P_0).
\]

Again, assume that we already know

\[
p(\theta_{k-1} | y_{1:k-1}) = \mathcal{N}(\theta_{k-1} | m_{k-1}, P_{k-1}).
\]

The **joint distribution** of \(\theta_k\) and \(\theta_{k-1}\) is (due to Markovianity of dynamics!):

\[
p(\theta_k, \theta_{k-1} | y_{1:k-1}) = p(\theta_k | \theta_{k-1}) p(\theta_{k-1} | y_{1:k-1}).
\]
Integrating over $\theta_{k-1}$ gives:

$$p(\theta_k \mid y_{1:k-1}) = \int p(\theta_k \mid \theta_{k-1}) p(\theta_{k-1} \mid y_{1:k-1}) d\theta_{k-1}.$$ 

This equation for Markov processes is called the Chapman-Kolmogorov equation.

Because the distributions are Gaussian, the result is Gaussian:

$$p(\theta_k \mid y_{1:k-1}) = N(\theta_k \mid m_k^-, P_k^-),$$

where

$$m_k^- = m_{k-1}$$

$$P_k^- = P_{k-1} + Q.$$
As in the pure recursive estimation, we get

\[ p(\theta_k \mid y_{1:k}) \propto p(y_k \mid \theta_k) \ p(\theta_k \mid y_{1:k-1}) \]

\[ \propto \text{N}(\theta_k \mid m_k, P_k). \]

After applying the matrix inversion lemma, mean and covariance can be written as

\[ S_k = H_k P^{-1}_k H^T_k + \sigma^2 \]
\[ K_k = P^{-1}_k H^T_k S_k^{-1} \]
\[ m_k = m^- + K_k [y_k - H_k m^-] \]
\[ P_k = P^- - K_k S_k K^T_k. \]

Again, we have derived a special case of the Kalman filter.

The batch version of this solution would be much more complicated.
State Space Notation

- In the previous slide we formulated the model as

\[ p(\theta_k \mid \theta_{k-1}) = \mathcal{N}(\theta_k \mid \theta_{k-1}, Q) \]
\[ p(y_k \mid \theta_k) = \mathcal{N}(y_k \mid H_k \theta_k, \sigma^2) \]

- But in Kalman filtering and control theory the vector of parameters \( \theta_k \) is usually called “state” and denoted as \( x_k \).

- More standard state space notation:

\[ p(x_k \mid x_{k-1}) = \mathcal{N}(x_k \mid x_{k-1}, Q) \]
\[ p(y_k \mid x_k) = \mathcal{N}(y_k \mid H_k x_k, \sigma^2) \]

- Or equivalently

\[ x_k = x_{k-1} + q_{k-1} \]
\[ y_k = H_k x_k + r_k, \]

where \( q_{k-1} \sim \mathcal{N}(0, Q), \ r_k \sim \mathcal{N}(0, \sigma^2) \).
The canonical Kalman filtering model is

\[ p(x_k | x_{k-1}) = N(x_k | A_{k-1} x_{k-1}, Q_{k-1}) \]
\[ p(y_k | x_k) = N(y_k | H_k x_k, R_k). \]

More often, this model can be seen in the form

\[ x_k = A_{k-1} x_{k-1} + q_{k-1} \]
\[ y_k = H_k x_k + r_k. \]

The Kalman filter actually calculates the following distributions:

\[ p(x_k | y_{1:k-1}) = N(x_k | m_k^-, P_k^-) \]
\[ p(x_k | y_{1:k}) = N(x_k | m_k, P_k). \]
Prediction step of the Kalman filter:

\[ m_k^- = A_{k-1} m_{k-1} \]
\[ P_k^- = A_{k-1} P_{k-1} A_{k-1}^T + Q_{k-1}. \]

Update step of the Kalman filter:

\[ S_k = H_k P_k^- H_k^T + R_k \]
\[ K_k = P_k^- H_k^T S_k^{-1} \]
\[ m_k = m_k^- + K_k [y_k - H_k m_k^-] \]
\[ P_k = P_k^- - K_k S_k K_k^T. \]

These equations can be derived from the general Bayesian filtering equations.
Generic non-linear state space models

\[ x_k = f(x_{k-1}, q_{k-1}) \]
\[ y_k = h(x_k, r_k). \]

Generic Markov models

\[ x_k \sim p(x_k | x_{k-1}) \]
\[ y_k \sim p(y_k | x_k). \]

Continuous-discrete state space models involving stochastic differential equations:

\[ \frac{dx}{dt} = f(x, t) + w(t) \]
\[ y_k \sim p(y_k | x(t_k)). \]
Non-linear state space model with unknown parameters:

\[ x_k = f(x_{k-1}, q_{k-1}, \theta) \]
\[ y_k = h(x_k, r_k, \theta). \]

General Markovian state space model with unknown parameters:

\[ x_k \sim p(x_k | x_{k-1}, \theta) \]
\[ y_k \sim p(y_k | x_k, \theta). \]

Parameter estimation will be considered later – for now, we will attempt to estimate the state.

Why Bayesian filtering and smoothing then?
In principle, we could just use the (batch) Bayes’ rule

\[
p(x_1, \ldots, x_T \mid y_1, \ldots, y_T) = \frac{p(y_1, \ldots, y_T \mid x_1, \ldots, x_T) p(x_1, \ldots, x_T)}{p(y_1, \ldots, y_T)},
\]

Curse of computational complexity: complexity grows more than linearly with number of measurements (typically we have \(O(T^3)\)).

Hence, we concentrate on the following:

- **Filtering distributions:**
  \[p(x_k \mid y_1, \ldots, y_k), \quad k = 1, \ldots, T.\]

- **Prediction distributions:**
  \[p(x_{k+n} \mid y_1, \ldots, y_k), \quad k = 1, \ldots, T, \quad n = 1, 2, \ldots,\]

- **Smoothing distributions:**
  \[p(x_k \mid y_1, \ldots, y_T), \quad k = 1, \ldots, T.\]
Bayesian Filtering, Prediction and Smoothing (cont.)

Measurements | Estimate
---|---
0 | Tk

Prediction:

Filtering:

Smoothing:

Measurements | Estimate
---|---

Simo Särkkä

Tutorial: Bayesian Filtering and Smoothing
Filtering Algorithms

- **Kalman filter** is the classical optimal filter for linear-Gaussian models.
- **Extended Kalman filter (EKF)** is linearization based extension of Kalman filter to non-linear models.
- **Unscented Kalman filter (UKF)** is sigma-point transformation based extension of Kalman filter.
- **Gauss-Hermite and Cubature Kalman filters (GHKF/CKF)** are numerical integration based extensions of Kalman filter.
- **Particle filter** forms a Monte Carlo representation (particle set) to the distribution of the state estimate.
- **Grid based filters** approximate the probability distributions on a finite grid.
- **Mixture Gaussian approximations** are used, for example, in multiple model Kalman filters and Rao-Blackwellized Particle filters.
Rauch-Tung-Striebel (RTS) smoother is the closed form smoother for linear Gaussian models.

Extended, statistically linearized and unscented RTS smoothers are the approximate nonlinear smoothers corresponding to EKF, SLF and UKF.

Gaussian RTS smoothers: cubature RTS smoother, Gauss-Hermite RTS smoothers and various others

Particle smoothing is based on approximating the smoothing solutions via Monte Carlo.

Rao-Blackwellized particle smoother is a combination of particle smoothing and RTS smoothing.
The dynamics of the car in 2d \((x_1, x_2)\) are given by the Newton's law:

\[ f(t) = ma(t), \]

where \(a(t)\) is the acceleration, \(m\) is the mass of the car, and \(f(t)\) is a vector of (unknown) forces acting the car.

We shall now model \(f(t)/m\) as a 2-dimensional white noise process:

\[
\begin{align*}
\frac{d^2 x_1}{dt^2} &= w_1(t) \\
\frac{d^2 x_2}{dt^2} &= w_2(t).
\end{align*}
\]
If we define \( x_3(t) = dx_1/dt, x_4(t) = dx_2/dt \), then the model can be written as a first order system of differential equations:

\[
\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}.
\]

In shorter matrix form:

\[
\frac{dx}{dt} = Fx + Lw.
\]
If the state of the car is measured (sampled) with sampling period $\Delta t$ it suffices to consider the state of the car only at the time instances $t \in \{0, \Delta t, 2\Delta t, \ldots \}$.

The dynamic model can be discretized, which leads to the linear difference equation model

$$x_k = A x_{k-1} + q_{k-1},$$

where $x_k = x(t_k)$, $A$ is the transition matrix and $q_k$ is a discrete-time Gaussian noise process.
Assume that the position of the car \((x_1, x_2)\) is measured and the measurements are corrupted by Gaussian measurement noise \(e_{1,k}, e_{2,k}\):

\[
y_{1,k} = x_{1,k} + e_{1,k} \\
y_{2,k} = x_{2,k} + e_{2,k}.
\]

The measurement model can be now written as

\[
y_k = H x_k + e_k, \quad H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}
\]
The dynamic and measurement models of the car now form a linear Gaussian filtering model:

\[ x_k = A x_{k-1} + q_{k-1} \]
\[ y_k = H x_k + r_k, \]

where \( q_{k-1} \sim N(0, Q) \) and \( r_k \sim N(0, R) \).

The posterior distribution is Gaussian

\[ p(x_k | y_1, \ldots, y_k) = N(x_k | m_k, P_k). \]

The mean \( m_k \) and covariance \( P_k \) of the posterior distribution can be computed by the Kalman filter.

The whole history of the states can be estimated with the Rauch–Tung–Striebel smoother.
Gravitation law:

\[ f = m \mathbf{a}(t) = -\frac{GM \mathbf{r}(t)}{|\mathbf{r}(t)|^3}. \]

If we also model the friction and uncertainties:

\[ \mathbf{a}(t) = -\frac{GM \mathbf{r}(t)}{|\mathbf{r}(t)|^3} - D(\mathbf{r}(t)) |\mathbf{v}(t)| \mathbf{v}(t) + \mathbf{w}(t). \]
If we define \( \mathbf{x} = (x_1, x_2, \frac{dx_1}{dt}, \frac{dx_2}{dt})^T \), the model is of the form

\[
\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) + \mathbf{L} \mathbf{w}(t).
\]

where \( \mathbf{f}(\cdot) \) is non-linear – do not confuse \( \mathbf{f}(\cdot) \) with the force! – we just ran out of letters.

The radar measurement:

\[
\begin{align*}
    r &= \sqrt{(x_1 - x_r)^2 + (x_2 - y_r)^2 + e_r} \\
    \theta &= \tan^{-1} \left( \frac{x_2 - y_r}{x_1 - x_r} \right) + e_\theta,
\end{align*}
\]

where \( e_r \sim \mathcal{N}(0, \sigma_r^2) \) and \( e_\theta \sim \mathcal{N}(0, \sigma_\theta^2) \).
By suitable numerical integration scheme the model can be approximately written as discrete-time state space model:

\[
x_k = f(x_{k-1}, q_{k-1})
\]
\[
y_k = h(x_k, r_k),
\]

where \(y_k\) is the vector of measurements, and \(q_{k-1} \sim N(0, Q)\) and \(r_k \sim N(0, R)\).

The tracking of the space vehicle can be now implemented by, e.g., extended Kalman filter (EKF), unscented Kalman filter (UKF) or particle filter.

The history of states can be estimated with non-linear smoothers.
General probabilistic state space model:

- Dynamic model: $x_k \sim p(x_k | x_{k-1})$
- Measurement model: $y_k \sim p(y_k | x_k)$

$x_k = (x_{k1}, \ldots, x_{kn})$ is the state and $y_k = (y_{k1}, \ldots, y_{km})$ is the measurement.

Has the form of hidden Markov model (HMM):

- Observed: $y_1 \uparrow y_2 \uparrow y_3 \uparrow y_4$
- Hidden: $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow \ldots$
Example (Gaussian random walk)

Gaussian random walk model can be written as

\[
\begin{align*}
x_k &= x_{k-1} + w_{k-1}, \quad w_{k-1} \sim N(0, q) \\
y_k &= x_k + e_k, \quad e_k \sim N(0, r),
\end{align*}
\]

where \(x_k\) is the hidden state and \(y_k\) is the measurement. In terms of probability densities the model can be written as

\[
\begin{align*}
p(x_k | x_{k-1}) &= \frac{1}{\sqrt{2\pi q}} \exp \left( -\frac{1}{2q} (x_k - x_{k-1})^2 \right) \\
p(y_k | x_k) &= \frac{1}{\sqrt{2\pi r}} \exp \left( -\frac{1}{2r} (y_k - x_k)^2 \right)
\end{align*}
\]

which is a discrete-time state space model.
Example (Gaussian random walk (cont.))
General form of linear Gaussian state space models:

\[ x_k = A x_{k-1} + q_{k-1}, \quad q_{k-1} \sim N(0, Q) \]
\[ y_k = H x_k + r_k, \quad r_k \sim N(0, R) \]

In probabilistic notation the model is:

\[ p(y_k \mid x_k) = N(y_k \mid H x_k, R) \]
\[ p(x_k \mid x_{k-1}) = N(x_k \mid A x_{k-1}, Q). \]

Surprisingly general class of models – linearity is from measurements to estimates, not from time to outputs.
Non-Linear State Space Models

- General form of non-linear Gaussian state space models:
  \[
  x_k = f(x_{k-1}, q_{k-1}) \\
  y_k = h(x_k, r_k).
  \]

- \(q_k\) and \(r_k\) are typically assumed Gaussian.

- Functions \(f(\cdot)\) and \(h(\cdot)\) are non-linear functions modeling the dynamics and measurements of the system.

- Equivalent to the generic probabilistic models of the form
  \[
  x_k \sim p(x_k \mid x_{k-1}) \\
  y_k \sim p(y_k \mid x_k).
  \]
Bayesian optimal filter computes the distribution

\[ p(x_k \mid y_{1:k}) \]

Given the following:

1. Prior distribution \( p(x_0) \).
2. State space model:
   \[
   x_k \sim p(x_k \mid x_{k-1})
   \]
   \[
   y_k \sim p(y_k \mid x_k),
   \]
3. Measurement sequence \( y_{1:k} = y_1, \ldots, y_k \).

Computation is based on recursion rule for incorporation of the new measurement \( y_k \) into the posterior:

\[
 p(x_{k-1} \mid y_{1:k-1}) \rightarrow p(x_k \mid y_{1:k})
\]
Optimal filter

- **Initialization**: The recursion starts from the prior distribution $p(x_0)$.
- **Prediction**: by the Chapman-Kolmogorov equation
  \[
  p(x_k | y_{1:k-1}) = \int p(x_k | x_{k-1}) p(x_{k-1} | y_{1:k-1}) \, dx_{k-1}.
  \]
- **Update**: by the Bayes’ rule
  \[
  p(x_k | y_{1:k}) = \frac{1}{Z_k} p(y_k | x_k) p(x_k | y_{1:k-1}).
  \]
- The normalization constant $Z_k = p(y_k | y_{1:k-1})$ is given as
  \[
  Z_k = \int p(y_k | x_k) p(x_k | y_{1:k-1}) \, dx_k.
  \]
On prediction step the distribution of previous step is propagated through the dynamics. Prior distribution from prediction and the likelihood of measurement. The posterior distribution after combining the prior and likelihood by Bayes’ rule.
Kalman Filter: Model

- Gaussian driven linear model, i.e., Gauss-Markov model:
  \[
  x_k = A_{k-1} x_{k-1} + q_{k-1} \\
  y_k = H_k x_k + r_k,
  \]

- \( q_{k-1} \sim \mathcal{N}(0, Q_{k-1}) \) white process noise.
- \( r_k \sim \mathcal{N}(0, R_k) \) white measurement noise.
- \( A_{k-1} \) is the transition matrix of the dynamic model.
- \( H_k \) is the measurement model matrix.
- In probabilistic terms the model is
  \[
  p(x_k \mid x_{k-1}) = \mathcal{N}(x_k \mid A_{k-1} x_{k-1}, Q_{k-1}) \\
  p(y_k \mid x_k) = \mathcal{N}(y_k \mid H_k x_k, R_k).
  \]

- Kalman filter computes
  \[
  p(x_k \mid y_{1:k}) = \mathcal{N}(x_k \mid m_k, P_k)
  \]
Kalman Filter: Equations

**Initialization:** $x_0 \sim N(m_0, P_0)$

**Prediction step:**

$$m_k^- = A_{k-1} m_{k-1}$$
$$P_k^- = A_{k-1} P_{k-1} A_{k-1}^T + Q_{k-1}.$$

**Update step:**

$$v_k = y_k - H_k m_k^-$$
$$S_k = H_k P_k^- H_k^T + R_k$$
$$K_k = P_k^- H_k^T S_k^{-1}$$
$$m_k = m_k^- + K_k v_k$$
$$P_k = P_k^- - K_k S_k K_k^T.$$
Basic **Non-Linear Gaussian State Space Model** is of the form:

\[
\begin{align*}
x_k &= f(x_{k-1}) + q_{k-1} \\
y_k &= h(x_k) + r_k
\end{align*}
\]

- \(x_k \in \mathbb{R}^n\) is the state
- \(y_k \in \mathbb{R}^m\) is the measurement
- \(q_{k-1} \sim N(0, Q_{k-1})\) is the Gaussian process noise
- \(r_k \sim N(0, R_k)\) is the Gaussian measurement noise
- \(f(\cdot)\) is the dynamic model function
- \(h(\cdot)\) is the measurement model function
In EKF, the non-linear functions are linearized as follows:

\[
\begin{align*}
    f(x) &\approx f(m) + F_x(m) (x - m) \\
    h(x) &\approx h(m) + H_x(m) (x - m)
\end{align*}
\]

where \( x \sim N(m, P) \), and \( F_x, H_x \) are the Jacobian matrices of \( f \) and \( h \), respectively.

- Only the first terms in linearization contribute to the approximate means of the functions \( f \) and \( h \).
- The second term has zero mean and defines the approximate covariances of the functions.
- Can be generalized into approximation of a non-linear transform.
The linear Gaussian approximation to the joint distribution of $x$ and $y = g(x) + q$, where $x \sim N(m, P)$ and $q \sim N(0, Q)$ is

$$
\begin{pmatrix} x \\ y \end{pmatrix} \sim N \left( \begin{pmatrix} m \\ \mu_\mathcal{L} \end{pmatrix}, \begin{pmatrix} P & C_L \\ C_L^T & S_L \end{pmatrix} \right),
$$

where

$$
\begin{align*}
\mu_\mathcal{L} &= g(m) \\
S_\mathcal{L} &= G_x(m) P G_x^T(m) + Q \\
C_L &= P G_x^T(m).
\end{align*}
$$
EKF Equations

Extended Kalman filter

- **Prediction:**
  \[
  \begin{align*}
  \mathbf{m}_k^- &= f(\mathbf{m}_{k-1}) \\
  \mathbf{P}_k^- &= \mathbf{F}_x(\mathbf{m}_{k-1}) \mathbf{P}_{k-1} \mathbf{F}^T_x(\mathbf{m}_{k-1}) + \mathbf{Q}_{k-1}.
  \end{align*}
  \]

- **Update:**
  \[
  \begin{align*}
  \mathbf{v}_k &= \mathbf{y}_k - h(\mathbf{m}_k^-) \\
  \mathbf{S}_k &= \mathbf{H}_x(\mathbf{m}_k^-) \mathbf{P}_k^- \mathbf{H}_x^T(\mathbf{m}_k^-) + \mathbf{R}_k \\
  \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_x^T(\mathbf{m}_k^-) \mathbf{S}_k^{-1} \\
  \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k \mathbf{v}_k \\
  \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T.
  \end{align*}
  \]
Problem: Determine the mean and covariance of $y$:

$$x \sim N(\mu, \sigma^2)$$

$$y = \sin(x)$$

Recall the linearization based approximation:

$$y = \sin(\mu) + \frac{\partial \sin(\mu)}{\partial \mu}(x - \mu) + \ldots$$

which gives

$$E[y] \approx E[\sin(\mu) + \cos(\mu)(x - \mu)] = \sin(\mu)$$

$$\text{Cov}[y] \approx E[(\sin(\mu) + \cos(\mu)(x - \mu) - \sin(\mu))^2] = \cos^2(\mu) \sigma^2.$$
Form 3 sigma points as follows:

\[
\mathcal{X}^{(0)} = \mu \\
\mathcal{X}^{(1)} = \mu + \sigma \\
\mathcal{X}^{(2)} = \mu - \sigma.
\]

Let's select some weights \( W^{(0)}, W^{(1)}, W^{(2)} \) such that the original mean and variance can be recovered by

\[
\mu = \sum_i W^{(i)} \mathcal{X}^{(i)} \\
\sigma^2 = \sum_i W^{(i)} (\mathcal{X}^{(i)} - \mu)^2.
\]
We use the same formula for approximating the moments of $y = \sin(x)$ as follows:

$$
\mu = \sum_i W(i) \sin(\chi(i))
$$

$$
\sigma^2 = \sum_i W(i) (\sin(\chi(i)) - \mu)^2.
$$

For vectors $x \sim N(m, P)$ the generalization of standard deviation $\sigma$ is the Cholesky factor $L = \sqrt{P}$:

$$
P = LL^T.
$$

The sigma points can be formed using columns of $L$ (here $c$ is a suitable positive constant):

$$
\chi^{(0)} = m
$$

$$
\chi^{(i)} = m + c L_i
$$

$$
\chi^{(n+i)} = m - c L_i
$$
For transformation \( y = g(x) \) the approximation is:

\[
\mu_y = \sum_i W^{(i)} g(x^{(i)})
\]

\[
\sum_y = \sum_i W^{(i)} (g(x^{(i)}) - \mu_y) (g(x^{(i)}) - \mu_y)^T.
\]

It is convenient to define transformed sigma points:

\[ \gamma^{(i)} = g(x^{(i)}) \]

Joint moments of \( x \) and \( y = g(x) + q \) are then approximated as:

\[
E \left[ \begin{pmatrix} x \\ g(x) + q \end{pmatrix} \right] \approx \sum_i W^{(i)} \begin{pmatrix} x^{(i)} \\ \gamma^{(i)} \end{pmatrix} = \begin{pmatrix} \mathbf{m} \\ \mu_y \end{pmatrix}
\]

\[
\text{Cov} \left[ \begin{pmatrix} x \\ g(x) + q \end{pmatrix} \right] \approx \sum_i W^{(i)} \begin{pmatrix} (x^{(i)} - \mathbf{m}) (x^{(i)} - \mathbf{m})^T & (x^{(i)} - \mathbf{m}) (\gamma^{(i)} - \mu_y)^T \\ (\gamma^{(i)} - \mu_y) (x^{(i)} - \mathbf{m})^T & (\gamma^{(i)} - \mu_y) (\gamma^{(i)} - \mu_y)^T + \mathbf{Q} \end{pmatrix}
\]

Simo Särkkä
Tutorial: Bayesian Filtering and Smoothing
Unscented transform

The unscented transform approximation to the joint distribution of \( \mathbf{x} \) and \( \mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{q} \) where \( \mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{P}) \) and \( \mathbf{q} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \) is

\[
\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \mathbf{m} \\ \mu_U \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_U \\ \mathbf{C}_U^T & \mathbf{S}_U \end{pmatrix} \right),
\]

where the sub-matrices are formed as follows:

1. Form the sigma points as

\[
\begin{align*}
\chi^{(0)} & = \mathbf{m} \\
\chi^{(i)} & = \mathbf{m} + \sqrt{n + \lambda} \left[ \sqrt{\mathbf{P}} \right]_i, \\
\chi^{(i+n)} & = \mathbf{m} - \sqrt{n + \lambda} \left[ \sqrt{\mathbf{P}} \right]_i, \quad i = 1, \ldots, n
\end{align*}
\]
Propagate the sigma points through $g(\cdot)$:

$$\gamma^{(i)} = g(\chi^{(i)}), \quad i = 0, \ldots, 2n.$$ 

The sub-matrices are then given as:

$$\mu_U = \sum_{i=0}^{2n} W_i^{(m)} \gamma^{(i)}$$

$$S_U = \sum_{i=0}^{2n} W_i^{(c)} (\gamma^{(i)} - \mu_U) (\gamma^{(i)} - \mu_U)^T + Q$$

$$C_U = \sum_{i=0}^{2n} W_i^{(c)} (\chi^{(i)} - m) (\gamma^{(i)} - \mu_U)^T.$$
λ is a scaling parameter defined as $\lambda = \alpha^2 (n + \kappa) - n$. 

α and κ determine the spread of the sigma points.

Weights $W_i^{(m)}$ and $W_i^{(c)}$ are given as follows:

$$
W_0^{(m)} = \lambda / (n + \lambda) \\
W_0^{(c)} = \lambda / (n + \lambda) + (1 - \alpha^2 + \beta) \\
W_i^{(m)} = 1 / \{2(n + \lambda)\}, \quad i = 1, \ldots, 2n \\
W_i^{(c)} = 1 / \{2(n + \lambda)\}, \quad i = 1, \ldots, 2n,
$$

β can be used for incorporating prior information on the (non-Gaussian) distribution of $x$. 
Unscented Kalman filter: Prediction step

1. Form the sigma points:

\[
\begin{align*}
\chi_{k-1}^{(0)} &= \mathbf{m}_{k-1}, \\
\chi_{k-1}^{(i)} &= \mathbf{m}_{k-1} + \sqrt{n + \lambda} \left[ \sqrt{\mathbf{P}_{k-1}} \right]_i, \\
\chi_{k-1}^{(i+n)} &= \mathbf{m}_{k-1} - \sqrt{n + \lambda} \left[ \sqrt{\mathbf{P}_{k-1}} \right]_i, \quad i = 1, \ldots, n.
\end{align*}
\]

2. Propagate the sigma points through the dynamic model:

\[
\hat{\chi}_k^{(i)} = \mathbf{f}(\chi_{k-1}^{(i)}). \quad i = 0, \ldots, 2n.
\]
Unscented Kalman filter: Prediction step (cont.)

3 Compute the predicted mean and covariance:

\[
\begin{align*}
\mathbf{m}_k^- &= \sum_{i=0}^{2n} W_i^{(m)} \hat{X}_k^{(i)} \\
\mathbf{P}_k^- &= \sum_{i=0}^{2n} W_i^{(c)} (\hat{X}_k^{(i)} - \mathbf{m}_k^-) (\hat{X}_k^{(i)} - \mathbf{m}_k^-)^T + \mathbf{Q}_{k-1}.
\end{align*}
\]
Unscented Kalman filter: Update step

1. Form the sigma points:

\[
\chi_{k_{(0)}} = m_k^-,
\]

\[
\chi_{k_{(i)}} = m_k^- + \sqrt{n + \lambda} \left[ \sqrt{P_k^-} \right]_i, \quad i = 1, \ldots, n.
\]

\[
\chi_{k_{(i+n)}} = m_k^- - \sqrt{n + \lambda} \left[ \sqrt{P_k^-} \right]_i, \quad i = 1, \ldots, n.
\]

2. Propagate sigma points through the measurement model:

\[
\hat{y}_{k_i} = h(\chi_{k_{(i)}}), \quad i = 0, \ldots, 2n.
\]
Compute the following:

\[ \mu_k = \sum_{i=0}^{2n} W_i^{(m)} \hat{y}_k^{(i)} \]

\[ S_k = \sum_{i=0}^{2n} W_i^{(c)} (\hat{y}_k^{(i)} - \mu_k) (\hat{y}_k^{(i)} - \mu_k)^T + R_k \]

\[ C_k = \sum_{i=0}^{2n} W_i^{(c)} (x_k^{-(i)} - m_k^-) (\hat{y}_k^{(i)} - \mu_k)^T \]

\[ K_k = C_k S_k^{-1} \]

\[ m_k = m_k^- + K_k [y_k - \mu_k] \]

\[ P_k = P_k^- - K_k S_k K_k^T. \]
Consider the transformation of $x$ into $y$:

$$x \sim \mathcal{N}(m, P)$$

$$y = g(x).$$

Form Gaussian approximation to $(x, y)$ by directly approximating the integrals:

$$\mu_M = \int g(x) \mathcal{N}(x \mid m, P) \, dx$$

$$S_M = \int (g(x) - \mu_M)(g(x) - \mu_M)^T \mathcal{N}(x \mid m, P) \, dx$$

$$C_M = \int (x - m)(g(x) - \mu_M)^T \mathcal{N}(x \mid m, P) \, dx.$$
Gaussian Moment Matching

The moment matching based Gaussian approximation to the joint distribution of $x$ and the transformed random variable $y = g(x) + q$ where $x \sim N(m, P)$ and $q \sim N(0, Q)$ is given as

$$
\begin{pmatrix} x \\ y \end{pmatrix} \sim N \left( \begin{pmatrix} m \\ \mu_M \end{pmatrix}, \begin{pmatrix} P & C_M \\ C_M^T & S_M \end{pmatrix} \right),
$$

where

$$
\mu_M = \int g(x) \ N(x \mid m, P) \, dx
$$

$$
S_M = \int (g(x) - \mu_M)(g(x) - \mu_M)^T \ N(x \mid m, P) \, dx + Q
$$

$$
C_M = \int (x - m)(g(x) - \mu_M)^T \ N(x \mid m, P) \, dx.
$$
Gaussian Filter [1/3]

Gaussian filter prediction

Compute the following Gaussian integrals:

\[ \mathbf{m}_k^- = \mathbf{m}_{k-1} = \int f(x_{k-1}) \mathcal{N}(x_{k-1} | \mathbf{m}_{k-1}, \mathbf{P}_{k-1}) \, dx_{k-1} \]

\[ \mathbf{P}_k^- = \int (f(x_{k-1}) - \mathbf{m}_k^-)(f(x_{k-1}) - \mathbf{m}_k^-)^T \times \mathcal{N}(x_{k-1} | \mathbf{m}_{k-1}, \mathbf{P}_{k-1}) \, dx_{k-1} + \mathbf{Q}_{k-1}. \]
1. Compute the following Gaussian integrals:

\[ \mu_k = \int h(x_k) \ N(x_k | m_k^-, P_k^-) \, dx_k \]

\[ S_k = \int (h(x_k) - \mu_k) (h(x_k) - \mu_k)^T \ N(x_k | m_k^-, P_k^-) \, dx_k + R_k \]

\[ C_k = \int (x_k - m_k^-) (h(x_k) - \mu_k)^T \ N(x_k | m_k^-, P_k^-) \, dx_k. \]

2. Then compute the following:

\[ K_k = C_k S_k^{-1} \]

\[ m_k = m_k^- + K_k (y_k - \mu_k) \]

\[ P_k = P_k^- - K_k S_k K_k^T. \]
Special case of assumed density filtering (ADF).

Multidimensional Gauss-Hermite quadrature \(\Rightarrow\) Gauss-Hermite Kalman filter (GHKF).

Cubature integration \(\Rightarrow\) Cubature Kalman filter (CKF).

Monte Carlo integration \(\Rightarrow\) Monte Carlo Kalman filter (MCKF).

Gaussian process / Bayes-Hermite Kalman filter: Form Gaussian process regression model from set of sample points and integrate the approximation.

Linearization (EKF), unscented transform (UKF), central differences, divided differences can be considered as special cases.

Note that all of these lead to Gaussian approximations

\[
p(x_k \mid y_{1:k}) \approx N(x_k \mid m_k, P_k)
\]
The spherical cubature rule is exact up to third degree:

\[
\int g(x) \ N(x \mid m, P) \ dx = \int g(m + \sqrt{P} \xi) \ N(\xi \mid 0, I) \ d\xi 
\approx \frac{1}{2n} \sum_{i=1}^{2n} g(m + \sqrt{P} \xi^{(i)}),
\]

where

\[\xi^{(i)} = \begin{cases} \sqrt{n} e_i, & i = 1, \ldots, n \\ -\sqrt{n} e_{i-n}, & i = n + 1, \ldots, 2n, \end{cases}\]

where \(e_i\) denotes a unit vector to the direction of coordinate axis \(i\).

A special case of unscented transform!
Cartesian product of classical Gauss–Hermite quadratures gives

\[
\int g(x) \ N(x \mid m, P) \, dx = \int g(m + \sqrt{P} \xi) \ N(\xi \mid 0, I) \, d\xi = \int \cdots \int g(m + \sqrt{P} \xi_1) \ N(\xi_1 \mid 0, 1) \, d\xi_1 \times \cdots \times N(\xi_n \mid 0, 1) \, d\xi_n \approx \sum_{i_1, \ldots, i_n} W(i_1) \times \cdots \times W(i_n) g(m + \sqrt{P} \xi(i_1, \ldots, i_n)).
\]

\(\xi(i_1, \ldots, i_n)\) are formed from the roots of Hermite polynomials.

\(W(i_j)\) are the weights of one-dimensional Gauss–Hermite rules.
Animation: Kalman vs. Particle Filtering:

- Kalman filter animation
- Particle filter animation
Sequential Importance Resampling (SIR) (= particle filtering) is concerned with models

\[ x_k \sim p(x_k \mid x_{k-1}) \]
\[ y_k \sim p(y_k \mid x_k) \]

The SIS algorithm uses a weighted set of particles \( \{(w_k^{(i)}, x_k^{(i)}) : i = 1, \ldots, N\} \) such that

\[ \mathbb{E}[g(x_k) \mid y_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} g(x_k^{(i)}). \]

Or equivalently

\[ p(x_k \mid y_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(x_k - x_k^{(i)}), \]

where \( \delta(\cdot) \) is the Dirac delta function.

Uses importance sampling sequentially.
Sequential Importance Resampling

- Draw point $x_k^{(i)}$ from the importance distribution:
  $$x_k^{(i)} \sim \pi(x_k \mid x_{0:k-1}^{(i)}, y_{1:k}), \quad i = 1, \ldots, N.$$

- Calculate new weights
  $$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(y_k \mid x_k^{(i)}) p(x_k^{(i)} \mid x_{k-1}^{(i)})}{\pi(x_k^{(i)} \mid x_{0:k-1}^{(i)}, y_{1:k})}, \quad i = 1, \ldots, N,$$
  and normalize them to sum to unity.

- If the effective number of particles is too low, perform resampling.
In bootstrap filter we use the dynamic model as the importance distribution
\[
\pi(x_k^{(i)} \mid x_{0:k-1}^{(i)}, y_{1:k}) = p(x_k^{(i)} \mid x_{k-1}^{(i)})
\]
and resample at every step:

**Bootstrap Filter**

- Draw point \( x_k^{(i)} \) from the dynamic model:
  \[
x_k^{(i)} \sim p(x_k \mid x_{k-1}^{(i)}), \quad i = 1, \ldots, N.
  \]
- Calculate new weights
  \[
w_k^{(i)} \propto p(y_k \mid x_k^{(i)}), \quad i = 1, \ldots, N,
  \]
  and normalize them to sum to unity.
- Perform resampling.
The optimal importance distribution is

$$\pi(x_k^{(i)} \mid x_{0:k-1}^{(i)}, y_{1:k}) = p(x_k^{(i)} \mid x_{k-1}^{(i)}, y_k)$$

Then the weight update reduces to

$$w_k^{(i)} \propto w_{k-1}^{(i)} p(y_k \mid x_{k-1}^{(i)}), \quad i = 1, \ldots, N.$$ 

The optimal importance distribution can be used, for example, when the state space is finite.
Sequential Importance Resampling: Importance Distribution via Kalman Filtering

- We can also form a Gaussian approximation to the optimal importance distribution:

$$p(x_k^{(i)} \mid x_{k-1}^{(i)}, y_k) \approx N(x_k^{(i)} \mid \tilde{m}_k^{(i)}, \tilde{P}_k^{(i)})$$

by using a single prediction and update steps of a Gaussian filter starting from a singular distribution at $x_{k-1}^{(i)}$.

- We can also replace above with the result of a Gaussian filter $N(m_{k-1}^{(i)}, P_{k-1}^{(i)})$ started from a random initial mean.

- A very common way seems to be to use the previous sample as the mean: $N(x_{k-1}^{(i)}, P_{k-1}^{(i)})$.

- A particle filter with UKF proposal has been given name unscented particle filter (UPF) – you can invent new PFs easily this way.
Rao-Blackwellized Particle Filter: Idea

- Rao-Blackwellized particle filtering (RBPF) is concerned with conditionally Gaussian models:

\[
p(x_k | x_{k-1}, u_{k-1}) = \mathcal{N}(x_k | A_{k-1}(u_{k-1}) x_{k-1}, Q_{k-1}(u_{k-1}))
\]
\[
p(y_k | x_k, u_k) = \mathcal{N}(y_k | H_k(u_k) x_k, R_k(u_k))
\]
\[
p(u_k | u_{k-1}) = \text{(any given form)},
\]

where

- \( x_k \) is the state
- \( y_k \) is the measurement
- \( u_k \) is an arbitrary latent variable

- Given the latent variables \( u_{1:T} \) the model is Gaussian.
- The RBPF uses SIR for the latent variables and computes the conditionally Gaussian part in closed form with Kalman filter.
Bayesian Smoothing Problem

- Probabilistic state space model:

  measurement model: \( y_k \sim p(y_k | x_k) \)

  dynamic model: \( x_k \sim p(x_k | x_{k-1}) \)

- Assume that the filtering distributions \( p(x_k | y_{1:k}) \) have already been computed for all \( k = 0, \ldots, T \).

- We want recursive equations of computing the smoothing distribution for all \( k < T \):

  \[ p(x_k | y_{1:T}). \]

- The recursion will go backwards in time, because on the last step, the filtering and smoothing distributions coincide:

  \[ p(x_T | y_{1:T}). \]
The Bayesian optimal smoothing equations consist of prediction step and backward update step:

\[
p(x_{k+1} | y_{1:k}) = \int p(x_{k+1} | x_k) p(x_k | y_{1:k}) \, dx_k
\]

\[
p(x_k | y_{1:T}) = p(x_k | y_{1:k}) \int \left[ \frac{p(x_{k+1} | x_k) p(x_{k+1} | y_{1:T})}{p(x_{k+1} | y_{1:k})} \right] \, dx_{k+1}
\]

The recursion is started from the filtering (and smoothing) distribution of the last time step \(p(x_T | y_{1:T})\).
Rauch-Tung-Striebel Smoother

Backward recursion equations for the smoothed means $m^s_k$ and covariances $P^s_k$:

\[
\begin{align*}
m^-_{k+1} &= A_k m_k \\
P^-_{k+1} &= A_k P_k A_k^T + Q_k \\
G_k &= P_k A_k^T [P_{k+1}^-]^{-1} \\
m^s_k &= m_k + G_k [m^s_{k+1} - m^-_{k+1}] \\
P^s_k &= P_k + G_k [P^s_{k+1} - P^-_{k+1}] G_k^T,
\end{align*}
\]

- $m_k$ and $P_k$ are the mean and covariance computed by the Kalman filter.
- The recursion is started from the last time step $T$, with $m^s_T = m_T$ and $P^s_T = P_T$. 

Simo Särkkä
Tutorial: Bayesian Filtering and Smoothing
The equations for the extended RTS smoother are

\[
\begin{align*}
\mathbf{m}^-_{k+1} &= \mathbf{f}(\mathbf{m}_k) \\
\mathbf{P}^-_{k+1} &= \mathbf{F}_x(\mathbf{m}_k) \mathbf{P}_k \mathbf{F}_x^T(\mathbf{m}_k) + \mathbf{Q}_k \\
\mathbf{G}_k &= \mathbf{P}_k \mathbf{F}_x^T(\mathbf{m}_k) \left[ \mathbf{P}^-_{k+1} \right]^{-1} \\
\mathbf{m}^s_k &= \mathbf{m}_k + \mathbf{G}_k \left[ \mathbf{m}^s_{k+1} - \mathbf{m}^-_{k+1} \right] \\
\mathbf{P}^s_k &= \mathbf{P}_k + \mathbf{G}_k \left[ \mathbf{P}^s_{k+1} - \mathbf{P}^-_{k+1} \right] \mathbf{G}_k^T,
\end{align*}
\]

where the matrix \( \mathbf{F}_x(\mathbf{m}_k) \) is the Jacobian matrix of \( \mathbf{f}(\mathbf{x}) \) evaluated at \( \mathbf{m}_k \).
The equations for the Gaussian RTS smoother are

\[ \mathbf{m}_{k+1}^- = \int f(\mathbf{x}_k) \mathcal{N}(\mathbf{x}_k | \mathbf{m}_k, \mathbf{P}_k) \, d\mathbf{x}_k \]

\[ \mathbf{P}_{k+1}^- = \int [f(\mathbf{x}_k) - \mathbf{m}_{k+1}^-] [f(\mathbf{x}_k) - \mathbf{m}_{k+1}^-]^T \times \mathcal{N}(\mathbf{x}_k | \mathbf{m}_k, \mathbf{P}_k) \, d\mathbf{x}_k + \mathbf{Q}_k \]

\[ \mathbf{D}_{k+1} = \int [\mathbf{x}_k - \mathbf{m}_k] [f(\mathbf{x}_k) - \mathbf{m}_{k+1}^-]^T \mathcal{N}(\mathbf{x}_k | \mathbf{m}_k, \mathbf{P}_k) \, d\mathbf{x}_k \]

\[ \mathbf{G}_k = \mathbf{D}_{k+1} [\mathbf{P}_{k+1}^-]^{-1} \]

\[ \mathbf{m}_k^s = \mathbf{m}_k + \mathbf{G}_k (\mathbf{m}_{k+1}^s - \mathbf{m}_{k+1}^-) \]

\[ \mathbf{P}_k^s = \mathbf{P}_k + \mathbf{G}_k (\mathbf{P}_{k+1}^s - \mathbf{P}_{k+1}^-) \mathbf{G}_k^T. \]
The smoothing solution can be obtained from SIR by storing the whole state histories into the particles.

Special care is needed on the resampling step.

The smoothed distribution approximation is then of the form

$$p(x_k | y_{1:T}) \approx \sum_{i=1}^{N} w_T^{(i)} \delta(x_k - x_k^{(i)}),$$

where $x_k^{(i)}$ is the $k$th component in $x_{1:T}$.

Unfortunately, the approximation is often quite degenerate.
Particle Smoothing: Backward Simulation

Backward simulation particle smoother

Given the weighted set of particles \( \{ w_k^{(i)}, x_k^{(i)} \} \) representing the filtering distributions:

- Choose \( \tilde{x}_T = x_T^{(i)} \) with probability \( w_T^{(i)} \).
- For \( k = T - 1, \ldots, 0 \):
  1. Compute new weights by
     \[
     w_{k|k+1}^{(i)} \propto w_k^{(i)} p(\tilde{x}_{k+1} | x_k^{(i)})
     \]
  2. Choose \( \tilde{x}_k = x_k^{(i)} \) with probability \( w_{k|k+1}^{(i)} \)

Given \( S \) iterations resulting in \( \tilde{x}_1^{(j)} : T \) for \( j = 1, \ldots, S \) the smoothing distribution approximation is

\[
p(x_1:T | y_1:T) \approx \frac{1}{S} \sum_j \delta(x_1:T - \tilde{x}_1^{(j)}:T).
\]
Particle Smoothing: Reweighting

Reweighting Particle Smoother

Given the weighted set of particles \( \{ w_k^{(i)}, x_k^{(i)} \} \) representing the filtering distribution, we can form approximations to the marginal smoothing distributions as follows:

- Start by setting \( w_T^{(i)} = w_T \) for \( i = 1, \ldots, n \).
- For each \( k = T - 1, \ldots, 0 \) do the following:
  - Compute new importance weights by
    \[
    w_k^{(i)} \propto \sum_j w_{k+1}^{(j)} \frac{w_k^{(i)} p(x_{k+1}^{(j)} \mid x_k^{(i)})}{\sum_l w_k^{(l)} p(x_{k+1}^{(l)} \mid x_k^{(l)})}.
    \]

At each step \( k \) the marginal smoothing distribution can be approximated as

\[
p(x_k \mid y_{1:T}) \approx \sum_i w_{k|T}^{(i)} \delta(x_k - x_k^{(i)}).
\]
Bayesian estimation of parameters

- State space model with unknown parameters $\theta \in \mathbb{R}^d$:
  \[
  \begin{align*}
  \theta & \sim p(\theta) \\
  x_0 & \sim p(x_0 \mid \theta) \\
  x_k & \sim p(x_k \mid x_{k-1}, \theta) \\
  y_k & \sim p(y_k \mid x_k, \theta).
  \end{align*}
  \]

- We approximate the marginal posterior distribution:
  \[
  p(\theta \mid y_{1:T}) \propto p(y_{1:T} \mid \theta) p(\theta)
  \]

- The key is the prediction error decomposition:
  \[
  p(y_{1:T} \mid \theta) = \prod_{k=1}^{T} p(y_k \mid y_{1:k-1}, \theta)
  \]

- Luckily, the Bayesian filtering equations allow us to compute $p(y_k \mid y_{1:k-1}, \theta)$ efficiently.
Recursion for marginal likelihood of parameters

The marginal likelihood of parameters is given by

\[
p(y_{1:T} \mid \theta) = \prod_{k=1}^{T} p(y_k \mid y_{1:k-1}, \theta)
\]

where the terms can be solved via the recursion

\[
p(x_k \mid y_{1:k-1}, \theta) = \int p(x_k \mid x_{k-1}, \theta) p(x_{k-1} \mid y_{1:k-1}, \theta) \, dx_{k-1}
\]

\[
p(y_k \mid y_{1:k-1}, \theta) = \int p(y_k \mid x_k, \theta) p(x_k \mid y_{1:k-1}, \theta) \, dx_k
\]

\[
p(x_k \mid y_{1:k}, \theta) = \frac{p(y_k \mid x_k, \theta) p(x_k \mid y_{1:k-1}, \theta)}{p(y_k \mid y_{1:k-1}, \theta)}.
\]
The energy function:

\[ \varphi_T(\theta) = -\log p(y_{1:T} | \theta) - \log p(\theta). \]

The posterior distribution can be recovered via

\[ p(\theta | y_{1:T}) \propto \exp(-\varphi_T(\theta)). \]

The energy function can be evaluated recursively as follows:

- Start from \( \varphi_0(\theta) = -\log p(\theta). \)
- At each step \( k = 1, 2, \ldots, T \) compute the following:

\[ \varphi_k(\theta) = \varphi_{k-1}(\theta) - \log p(y_k | y_{1:k-1}, \theta) \]

For linear models, we can evaluate the energy function exactly with help of Kalman filter.

In non-linear models we can use Gaussian filters or particle filters for approximating the energy function.
MAP and ML-estimates can be computed by direct optimization of the energy function (or posterior).

Derivatives of the energy function can be computed via sensitivity equations or Fisher’s identity.

Markov chain Monte Carlo (MCMC) methods can be used to sample from the posterior once the energy function is known.

When particle filter approximation and MCMC is combined we get the exact particle Markov chain Monte Carlo (PMCMC) method.

EM-algorithm can be used for computing MAP or ML-estimates when energy function is not available.
Probabilistic state space models can be used to model various dynamic phenomena, e.g., dynamics of a car or re-entry vehicle.

Bayesian filtering and smoothing methods solve Bayesian inference problems on state space models recursively.

Kalman filter is the closed form linear Gaussian filtering solution.

Extended Kalman filter (EKF) is linearization based extension of Kalman filter to non-linear models.

Unscented Kalman filter (UKF) is sigma-point transformation based extension of Kalman filter.

Gauss-Hermite and Cubature Kalman filters (GHKF/CKF) are numerical integration based extensions of Kalman filter.

Particle filter forms a Monte Carlo representation (particle set) to the distribution of the state estimate.
Summary (cont.)

- **Rauch-Tung-Striebel (RTS) smoother** is the closed form smoother for **linear Gaussian models**.
- **Extended, unscented, cubature, and related RTS smoothers** are the approximate nonlinear smoothers for **non-linear models**.
- **Particle smoothing** is based on approximating the smoothing solutions via **Monte Carlo**.
- The **marginal posterior distribution of state-space model parameters** can be computed from the results of Bayesian filter.
- Given the marginal posterior, we can, e.g., compute **MAP/ML estimates** or use **MCMC methods** (or even **EM-algorithms**).
- For **non-linear/non-Gaussian models** the parameter posterior can be approximated with **non-linear Kalman filters and particle filters**.

✓ Also freely available ONLINE at becs.aalto.fi/~ssarkka/