Sensor Fusion: Summary and Outlook

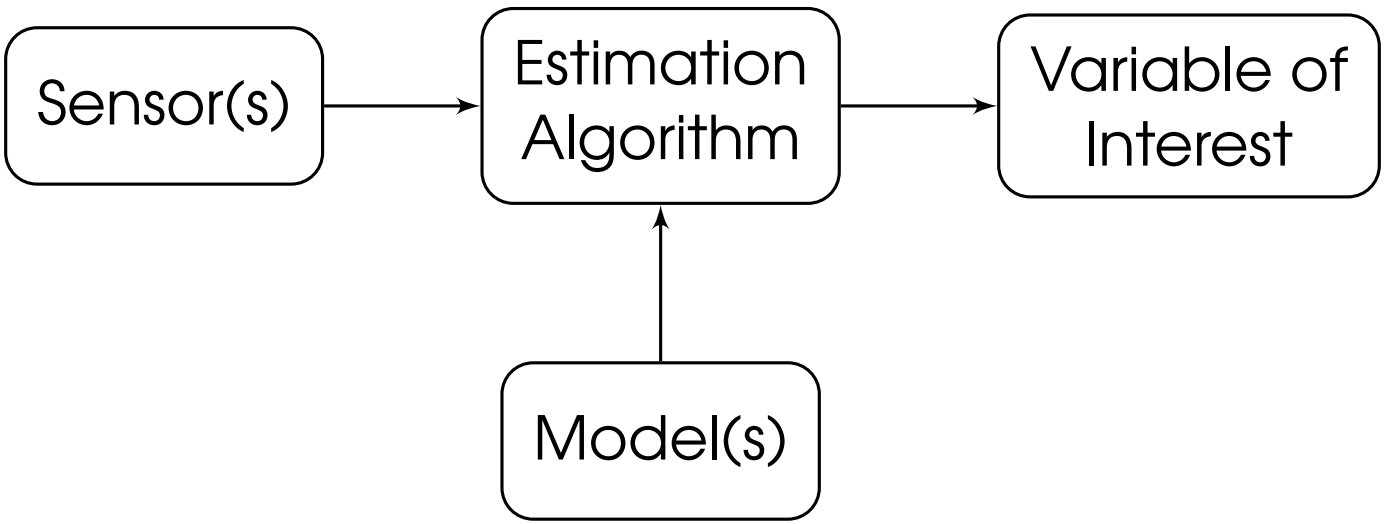
Roland Hostettler

December 5, 2018

Sensor Fusion Problem

Sensor fusion aims at estimating unknown (hidden) parameters and/or the state of a dynamic system using noisy measurements from one or multiple sensors.

(Ideal) Sensor Fusion Process



Steps for Developing a Sensor Fusion System

1. Define the objective (parameters/state) of interest
2. Define the sensors to measure the quantity
3. Model the dynamic behavior of the state
4. Model the relationship between the state and the measurements
5. Discretize the continuous-time dynamic model
6. Apply an appropriate estimation algorithm

Sensors

- Measure a physical or virtual quantity
- The parameter of interest may be the quantity itself or a related quantity (e.g., position vs. range, no. of clicks vs. website popularity)
- Sensor measurements are always noise (e.g., thermal noise, modeling inaccuracies, etc.)
- Typical sensor model for the nth measurement:

$$\boldsymbol{y}_n = g(\boldsymbol{\theta}) + \boldsymbol{r}_n$$

- $\boldsymbol{r}_n$ is the measurement noise

Measurement Noise (1/2)

- The value of the measurement noise $\boldsymbol{r}_n$ is unknown
- We can not invert $g(\boldsymbol{\theta})$ to estimate $\boldsymbol{\theta}$ from $\boldsymbol{y}_n$
- Measurement noise is modeled as a random variable
- Each realization of $\boldsymbol{r}_n$ is different

Measurement Noise (2/2)

- Realizations of $\boldsymbol{r}_n$ follow a probability density function:

$$\boldsymbol{r}_n \sim p(\boldsymbol{r}_n)$$

- The mean of $\boldsymbol{r}_n$ is the average value of all the possible realizations of $\boldsymbol{r}_n$ or the expected value:

$$\mathrm{E}\{\boldsymbol{r}_n\} = \int_{-\infty}^{\infty} \boldsymbol{r}_n p(\boldsymbol{r}_n) \mathrm{d}\boldsymbol{r}_n$$

- The (co-)variance is a measure of the spread of the realizations (i.e., how far from the mean the realizations are to be expected):

$$\begin{aligned}\mathrm{Cov}\{\boldsymbol{r}_n\} &= \mathrm{E}\{(\boldsymbol{r}_n - \mathrm{E}\{\boldsymbol{r}_n\})(\boldsymbol{r}_n - \mathrm{E}\{\boldsymbol{r}_n\})^{\mathsf{T}}\} \\ &= \int_{-\infty}^{\infty} (\boldsymbol{r}_n - \mathrm{E}\{\boldsymbol{r}_n\})(\boldsymbol{r}_n - \mathrm{E}\{\boldsymbol{r}_n\})^{\mathsf{T}} p(\boldsymbol{r}_n) \mathrm{d}\boldsymbol{r}_n\end{aligned}$$

Cost Functions (1/2)

- The estimation aims to minimize the effect of the measurement noise
- An optimality criterion is required to develop an estimation algorithm
- Optimality criterion: Minimize the cost of choosing parameter $\boldsymbol{\theta}$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

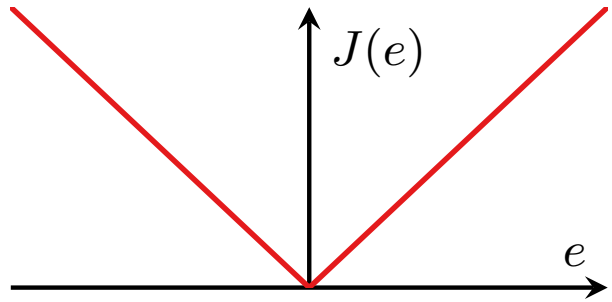
- The error e_n is the difference between the measurements and the output predicted by the model for some value $\boldsymbol{\theta}$

Cost Functions (2/2)

Absolute Error

Penalizes all errors equally:

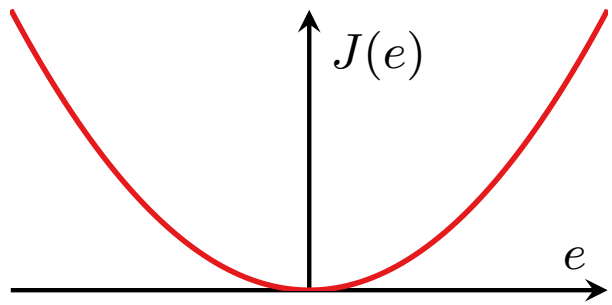
$$J(\boldsymbol{\theta}) = \sum_n |e_n(\boldsymbol{\theta})|$$



Quadratic Error

Penalizes large errors more than small ones:

$$J(\boldsymbol{\theta}) = \sum_n e_n(\boldsymbol{\theta})^2$$



Linear Static Model: Scalar Sensor

- Basic scalar model:

$$y_n = c\theta + r_n$$

  with $\mathrm{E}\{r_n\} = 0$, $\mathrm{var}\{r_n\} = \sigma_r^2$

- Multiple parameters:

$$y_n = \begin{bmatrix} c_1 & \dots & c_K \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_K \end{bmatrix} = \boldsymbol{c}\boldsymbol{\theta} + r_n$$

- Multiple measurements:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \vdots \\ \boldsymbol{c} \end{bmatrix} \boldsymbol{\theta} + \begin{bmatrix} r_1 \\ \vdots \\ r_N \end{bmatrix}, \quad \mathrm{Cov}\{\boldsymbol{r}\} = \sigma_r^2 \boldsymbol{I}_N$$

Linear Static Model: Vector Sensor

- Basic vector model:

$$\boldsymbol{y}_n = \boldsymbol{C}\boldsymbol{\theta} + \boldsymbol{r}_n$$

  with $\mathrm{E}\{\boldsymbol{r}_n\} = 0$, $\mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n$

- Multiple measurements

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} \\ \vdots \\ \boldsymbol{C} \end{bmatrix} \boldsymbol{\theta} + \begin{bmatrix} \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_N \end{bmatrix}, \quad \mathrm{Cov}\{\boldsymbol{r}\} = \begin{bmatrix} \boldsymbol{R}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{R}_N \end{bmatrix}$$

- The scalar sensor model is a special case of the vector sensor model
- Both can be written compactly as

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}, \quad \mathrm{Cov}\{\boldsymbol{r}\} = \boldsymbol{R}$$

Linear Static Model: Estimation (1/2)

- Linear model:

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}$$

  with $\mathrm{E}\{\boldsymbol{r}\} = 0$, $\mathrm{Cov}\{\boldsymbol{r}\} = \boldsymbol{R}$

- Weighted linear least squares problem:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})$$

- Solution steps:
  1. Calculate the gradient w.r.t. $\boldsymbol{\theta}$
  2. Set gradient to zero and solve for $\boldsymbol{\theta}$
- Solution:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G})^{-1} \boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{y}$$

$$\mathrm{Cov}\{\boldsymbol{\theta}\} = (\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G})^{-1}$$

Linear Static Model: Estimation (2/2)

- Linear model:

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}$$

- Prior knowledge:

$$\mathrm{E}\{\boldsymbol{\theta}\} = \boldsymbol{m}, \quad \mathrm{Cov}\{\boldsymbol{\theta}\} = \boldsymbol{P}$$

- Regularized linear least squares cost function:

$$J(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}) + (\boldsymbol{\theta} - \boldsymbol{m})^{\mathsf{T}} \boldsymbol{P}^{-1} (\boldsymbol{\theta} - \boldsymbol{m})$$

- Solution:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G} + \boldsymbol{P}^{-1})^{-1} (\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{y} + \boldsymbol{P}^{-1} \boldsymbol{m})$$

- Alternative form:

$$\hat{\boldsymbol{\theta}} = \boldsymbol{m} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{m})$$

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{G}^{\mathsf{T}} (\boldsymbol{G}\boldsymbol{P}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{R})^{-1}$$

$$\mathrm{Cov}\{\hat{\boldsymbol{\theta}}\} = \boldsymbol{P} - \boldsymbol{K}(\boldsymbol{G}\boldsymbol{P}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{R})\boldsymbol{K}^{\mathsf{T}}$$

Nonlinear Static Model

- General vector sensor model:

$$\boldsymbol{y}_n = g_n(\boldsymbol{\theta}) + \boldsymbol{r}_n$$
$$\mathrm{E}\{\boldsymbol{r}_n\} = 0,\ \mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n$$

- Measurement stacking:

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix} = \begin{bmatrix} g_1(\boldsymbol{\theta}) \\ g_2(\boldsymbol{\theta}) \\ \vdots \\ g_N(\boldsymbol{\theta}) \end{bmatrix} + \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_N \end{bmatrix}$$

- Compact notation:

$$\boldsymbol{y} = g(\boldsymbol{\theta}) + \boldsymbol{r}$$

Nonlinear Least Squares

- Weighted nonlinear least squares cost function:

$$J(\boldsymbol{\theta}) = (\boldsymbol{y} - g(\boldsymbol{\theta}))^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\boldsymbol{\theta}))$$

- Generally closed form solutions can not be found
- Numerical optimization methods iteratively find $\hat{\boldsymbol{\theta}}$ that minimizes the cost
- Iterations:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \Delta\boldsymbol{\theta}^{(i+1)}$$

Gradient Descent

- Search direction is the opposite of the cost function's gradient at the current point:

$$\Delta\boldsymbol{\theta}^{(i+1)} \propto -\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}^{(i)}}$$

- Gradient of nonlinear least squares cost function:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1}(\boldsymbol{y} - g(\boldsymbol{\theta}))$$

- Iteration:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1}(\boldsymbol{y} - g(\boldsymbol{\theta}^{(i)}))$$

- Quick in highly nonlinear areas, poor in flat (small gradient) areas

Gauss–Newton

- The nonlinear function is linearized using a Taylor series expansion around the current point:

$$g(\boldsymbol{\theta}) \approx g(\hat{\boldsymbol{\theta}}^{(i)}) + \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})$$

- The cost function becomes linear, solution:

$$\Delta\boldsymbol{\theta}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$

- Iteration:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$

- Poor approximation in highly nonlinear areas, good in flat (nearly linear) areas

Levenberg–Marquardt

- Regularized Gauss–Newton approach to avoid too large jumps:

$$J(\boldsymbol{\theta}) = (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) + \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}))^{\mathsf{T}} \boldsymbol{R}^{-1}(\dots) + \lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})$$

- Step direction:

$$\Delta\hat{\boldsymbol{\theta}}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}} + \lambda \boldsymbol{I})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$

- λ controls the algorithm behavior:
  - λ small $\Rightarrow$ Gauss–Newton
  - λ large $\Rightarrow$ Gradient descent

Modeling of Dynamic Systems

- (Stochastic/ordinary) differential (difference) equations describe the dynamic behavior of systems
- Lth order SDE:

$$\frac{\mathrm{d}^L z(t)}{\mathrm{d}t^L} = a_0 z(t) + a_1 \frac{\mathrm{d}z(t)}{\mathrm{d}t} + \cdots + a_{L-1} \frac{\mathrm{d}^{L-1} z(t)}{\mathrm{d}t^{L-1}} + b_1 w(t)$$

  with $\mathrm{E}\{w(t)\} = 0$, $R_{ww}(\tau) = \mathrm{E}\{w(t+\tau)w(t)\} = \sigma_w^2 \delta(\tau)$
- State-space/first order vector SDE representation of the Lth order SDE:

$$\begin{bmatrix} \frac{\mathrm{d}z(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}^2 z(t)}{\mathrm{d}t^2} \\ \vdots \\ \frac{\mathrm{d}^L z(t)}{\mathrm{d}t^L} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & 0 \\ a_0 & a_1 & & \dots & a_{L-1} \end{bmatrix} \begin{bmatrix} z(t) \\ \frac{\mathrm{d}z(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{L-1} z(t)}{\mathrm{d}t^{L-1}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_1 \end{bmatrix} w(t)$$

Linear State-Space Models

- Linear state-space model:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_w\boldsymbol{w}(t)$$
$$\boldsymbol{y}_n = \boldsymbol{G}\boldsymbol{x}_n + \boldsymbol{r}_n$$

  with $R_{ww}(\tau) = \boldsymbol{\Sigma}_w\delta(\tau)$, $\mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n$
- Discretization of the dynamic model = integration between t_{n-1} and t_n
- Integrating factor: $e^{-\boldsymbol{A}t}$
- Closed form solution:

$$\boldsymbol{x}_n = \underbrace{e^{\boldsymbol{A}\Delta t}}_{\triangleq \boldsymbol{F}}\boldsymbol{x}_{n-1} + \underbrace{\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\boldsymbol{w}(t)\mathrm{d}t}_{\triangleq \boldsymbol{q}_n}$$

The Process Noise

- Process noise $\boldsymbol{q}_n \triangleq \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)} \boldsymbol{B}_w \boldsymbol{w}(t) \mathrm{d}t$
- Mean:

$$\begin{aligned} \mathrm{E}\{\boldsymbol{q}_n\} &= \mathrm{E}\left\{ \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)} \boldsymbol{B}_w \boldsymbol{w}(t) \mathrm{d}t \right\} \\ &= \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)} \boldsymbol{B}_w \, \mathrm{E}\{\boldsymbol{w}(t)\} \mathrm{d}t = 0 \end{aligned}$$

- Covariance:

$$\begin{aligned} \mathrm{Cov}\{\boldsymbol{q}_n\} &= \mathrm{E}\{\boldsymbol{q}_n \boldsymbol{q}_n^{\mathsf{T}}\} \\ &= \mathrm{E}\left\{ \left(\int e^{\boldsymbol{A}(t_n - t)} \boldsymbol{B}_w \boldsymbol{w}(t) \mathrm{d}t \right) \left(\int e^{\boldsymbol{A}(t_n - \tau)} \boldsymbol{B}_w \boldsymbol{w}(\tau) \mathrm{d}\tau \right)^{\mathsf{T}} \right\} \\ &= \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - \tau)} \boldsymbol{B}_w \boldsymbol{\Sigma}_{ww} \boldsymbol{B}_w^{\mathsf{T}} e^{\boldsymbol{A}^{\mathsf{T}}(t_n - \tau)} \mathrm{d}\tau \end{aligned}$$

Nonlinear Dynamic Models

- For nonlinear dynamic systems, we can write:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_{d_x}(t) \end{bmatrix} = \begin{bmatrix} f_1(x_1(t), x_2(t), \ldots, x_{d_x}(t)) \\ f_2(x_1(t), x_2(t), \ldots, x_{d_x}(t)) \\ \vdots \\ f_{d_x}(x_1(t), x_2(t), \ldots, x_{d_x}(t)) \end{bmatrix} + \boldsymbol{B}_w \boldsymbol{w}(t)$$

- Compact notation:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + \boldsymbol{B}_w \boldsymbol{w}(t)$$

- $f(\boldsymbol{x}(t))$ is a vector-valued function (output is a $d_x \times 1$ vector)

General Nonlinear State-Space Model

- Nonlinear state-space model:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + \boldsymbol{B}_w \boldsymbol{w}(t)$$
$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n$$

  with $R_{ww}(\tau) = \boldsymbol{\Sigma}_w \delta(\tau)$, $\mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n$
- Discretization:
  - Linearize the nonlinear, continuous-time model and solve the linear problem
  - Approximate the integral (Euler–Maruyama approximation)

Linearization-Based Discretization

- Nonlinear dynamic model:

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t)) + \boldsymbol{B}_w \boldsymbol{w}(t) \\ &\approx f(\boldsymbol{x}_{n-1}) + \boldsymbol{A}_x(\boldsymbol{x}(t) - \boldsymbol{x}_{n-1}) + \boldsymbol{B}_w \boldsymbol{w}(t) \\ &= \boldsymbol{A}_x \boldsymbol{x}(t) + f(\boldsymbol{x}_{n-1}) - \boldsymbol{A}_x \boldsymbol{x}_{n-1} + \boldsymbol{B}_w \boldsymbol{w}(t)\end{aligned}$$

- Discretization (same approach as for linear systems):

$$\begin{aligned}\boldsymbol{x}_n &= e^{\boldsymbol{A}_x t}\boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}(f(x_{n-1}) - \boldsymbol{A}_x \boldsymbol{x}_{n-1})\mathrm{d}t \\ &\quad + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t \\ &= \boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\mathrm{d}t f(x_{n-1}) + \boldsymbol{q}_n\end{aligned}$$

- Process noise covariance:

$$\boldsymbol{Q}_n = \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - \tau)}\boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}} e^{\boldsymbol{A}_x^{\mathsf{T}}(t_n - \tau)}$$

Euler–Maruyama Discretization (1/2)

- Integral equation:

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} f(\boldsymbol{x}(t))\mathrm{d}t + \int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t$$

- Integral approximation:

$$\boldsymbol{x}_n \approx \boldsymbol{x}_{n-1} + f(\boldsymbol{x}(t))\Delta t + \boldsymbol{q}_n$$

- Mean of the process noise $\boldsymbol{q}_n \triangleq \int \boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t$:

$$\begin{aligned}\mathrm{E}\{\boldsymbol{q}_n\} &= \mathrm{E}\left\{\int \boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t\right\} \\ &= \int \boldsymbol{B}_w \,\mathrm{E}\left\{\boldsymbol{w}(t)\right\}\mathrm{d}t \\ &= 0\end{aligned}$$

Euler–Maruyama Discretization (2/2)

- Covariance of the process noise $\boldsymbol{q}_n \triangleq \int \boldsymbol{B}_w \boldsymbol{w}(t) \mathrm{d}t$:

$$\begin{aligned}
\mathrm{Cov}\{\boldsymbol{q}_n \boldsymbol{q}_n^{\mathsf{T}}\} &= \mathrm{E}\left\{ \left(\int \boldsymbol{B}_w \boldsymbol{w}(t) \mathrm{d}t \right) \left(\int \boldsymbol{B}_w \boldsymbol{w}(\tau) \mathrm{d}\tau \right)^{\mathsf{T}} \right\} \\
&= \int \mathrm{E}\left\{ \boldsymbol{B}_w \boldsymbol{w}(t) \boldsymbol{w}(\tau)^{\mathsf{T}} \boldsymbol{B}_w^{\mathsf{T}} \right\} \mathrm{d}t \mathrm{d}\tau \\
&= \int \boldsymbol{B}_w \, \mathrm{E}\{\boldsymbol{w}(t) \boldsymbol{w}(\tau)^{\mathsf{T}}\} \boldsymbol{B}_w^{\mathsf{T}} \mathrm{d}t \mathrm{d}\tau \\
&= \int \boldsymbol{B}_w \boldsymbol{\Sigma}_w \delta(\tau) \boldsymbol{B}_w^{\mathsf{T}} \mathrm{d}t \mathrm{d}\tau \\
&\approx \Delta t \boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}}
\end{aligned}$$

- Euler–Maruyama discretization:

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + f(\boldsymbol{x}_{n-1}) \Delta t + \boldsymbol{q}_n, \; \mathrm{Cov}\{\boldsymbol{Q}_n\} \approx \Delta t \boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}}$$

Estimation in the Dynamic Case: Filtering

- State changes continuously $\Rightarrow$ Estimation methods for static parameters do not work
- Sequential estimation algorithms $\Rightarrow$ Filtering
- Two-step algorithms:
  1. Prediction: Use the dynamic model to predict the state at time t_n based on the estimate at t_{n-1}
  2. Measurement update: Use the measurements to estimate the current state at t_n

Kalman Filter

- Exactly solves the filtering problem for linear models:

$$\boldsymbol{x}_n = \boldsymbol{F}\boldsymbol{x}_{n-1} + \boldsymbol{q}_n$$
$$\boldsymbol{y}_n = \boldsymbol{G}\boldsymbol{x}_n + \boldsymbol{r}_n$$

- Prediction:

$$\hat{\boldsymbol{x}}_{n|n-1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n-1|n-1}$$
$$\boldsymbol{P}_{n|n-1} = \boldsymbol{F}\boldsymbol{P}_{n-1|n-1}\boldsymbol{F}^{\mathsf{T}} + \boldsymbol{Q}_n$$

- Measurement update:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1}\boldsymbol{G}^{\mathsf{T}}(\boldsymbol{G}\boldsymbol{P}_{n|n-1}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{R}_n)^{-1}$$
$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n(\boldsymbol{y}_n - \boldsymbol{G}\hat{\boldsymbol{x}}_{n|n-1})$$
$$\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n(\boldsymbol{G}\boldsymbol{P}_{n|n-1}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{R}_n)\boldsymbol{K}_n^{\mathsf{T}}$$

Extended and Unscented Kalman Filter

- Both solve the filtering problem for nonlinear models:

$$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n$$
$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n$$

- The EKF linearizes the dynamic and measurement models using a Taylor series expansion
- The UKF uses a set of deterministic sigma-points based on the current estimation (prediction) to calculate the updated mean and covariances
- Both operate within the Kalman filtering framework (propagating the mean and covariance)

Bootstrap Particle Filter

- Propagates a weighted set of randomly drawn samples (particles)
- Uses the dynamic model to propagate the samples:

$$\boldsymbol{x}_n^j = f(\boldsymbol{x}_{n-1}^j) + \boldsymbol{q}_n^j$$

- Uses the likelihood $p(\boldsymbol{y}_n \mid \boldsymbol{x}_n)$ to evaluate the importance weight of each particle

$$\tilde{w}_n^j = p(\boldsymbol{y}_n \mid \boldsymbol{x}_n^j)$$

$$w_n^j = \frac{\tilde{w}_n^j}{\sum_{k=1}^J \tilde{w}_n^k}$$

- Requires resampling to avoid particle degeneracy
- Suitable for arbitrary models (linear or nonlinear)

Practical Issues

- Sensor calibration
- Sensor synchronization
- Choice of process noise spectral densities and measurement noise covariance
- Communication and storage
- Scaling, numerical stability, round-off errors
- ...

What's Next?

- Filtering, smoothing, and estimation in dynamic systems:
  - ELEC-E8105: Non-linear Filtering and Parameter Estimation
  - 5 credits, period III–IV (January 9, 2019 – April 10, 2019)
- Sensor types, construction, etc.:
  - ELEC-E5710: Sensors and Measurement Methods
  - 5 credits, period IV–V 2019
- Stochastic dynamic systems:
  - EEA-EV: Applied Stochastic Differential Equations
  - 3 credits, preliminarily 2020
- Possibilities for projects and theses

Announcements

- Project:
  - No experiments after December 11, 2018, plan your time!
  - Final report deadline: December 14, 2018, 23:55 EET
  - Final report = Revised intermediate report + part II
  - Review deadline: December 21, 23:55 EET
- Exam:
  - Monday, December 10, 2018, 14.00–17.00 in TU6
  - Allowed aids: 1 handwritten A4 sheet (not written by computer, not photocopied, etc.)
- Course feedback:
  - Please participate!
  - What did you like?
  - What should be improved?
  - What did you miss?
  - Were your expectations met?