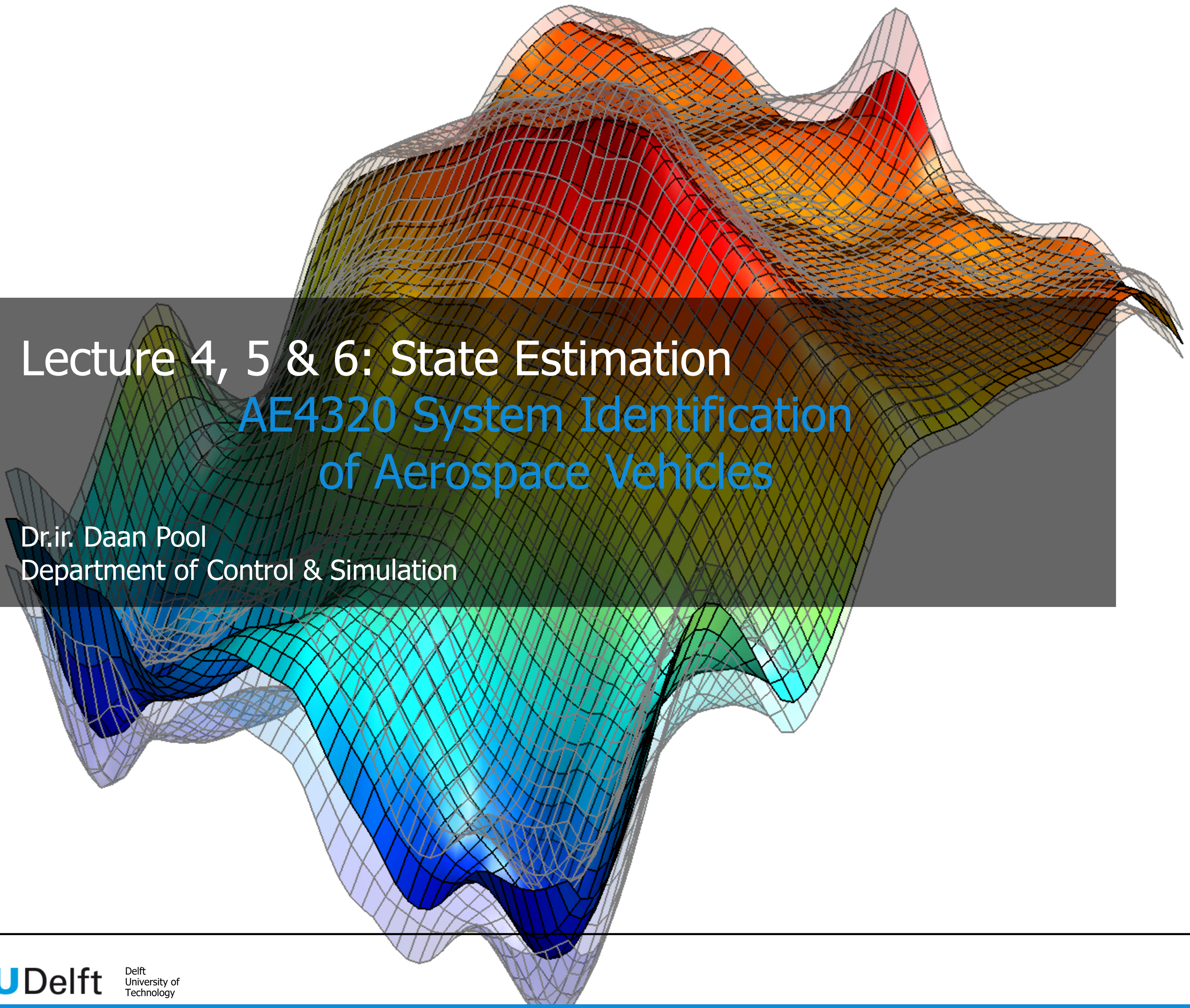


Lecture 4, 5 & 6: State Estimation

AE4320 System Identification of Aerospace Vehicles

Dr.ir. Daan Pool
Department of Control & Simulation

Course Outline

- **Lecture 1: (dr.ir. Coen de Visser)**
  - Course goals and objectives
  - Introduction to System Identification

- **Lecture 2,3: (dr.ir. Daan Pool)**
  - System Identification Experiments

- **Lecture 4,5,6: (dr.ir. Daan Pool)**
  - Kalman filters
  - State estimation & Sensor Fusion

- **Lecture 7,8: (dr.ir. Coen de Visser)**
  - Model structure selection
  - Model parameter estimation

Course Outline

- **Lecture 9: (dr.ir. Coen de Visser)**
    - Advanced identification approach: Neural networks
- **Lecture 10,11: (dr.ir. Coen de Visser)**
    - Advanced identification approach: Multivariate B-Splines
- **Lecture 12: (dr.ir. Coen de Visser)**
    - Model validation, course conclusion

Goals of this Lecture

Questions that will be answered during this lecture:

1. *What is state estimation, and why do we need it?*
2. *What is the relationship between state estimation and sensor fusion?*
3. *What is a Kalman filter?*
4. *What different types of Kalman filters are there and when do we use which type?*
5. *How do we develop a Kalman filter state estimator for a given system?*
6. *What are the limitations of Kalman filter state estimation?*

SysID High Level Overview

Where we are now in the System Identification Cycle:

Experiment phase

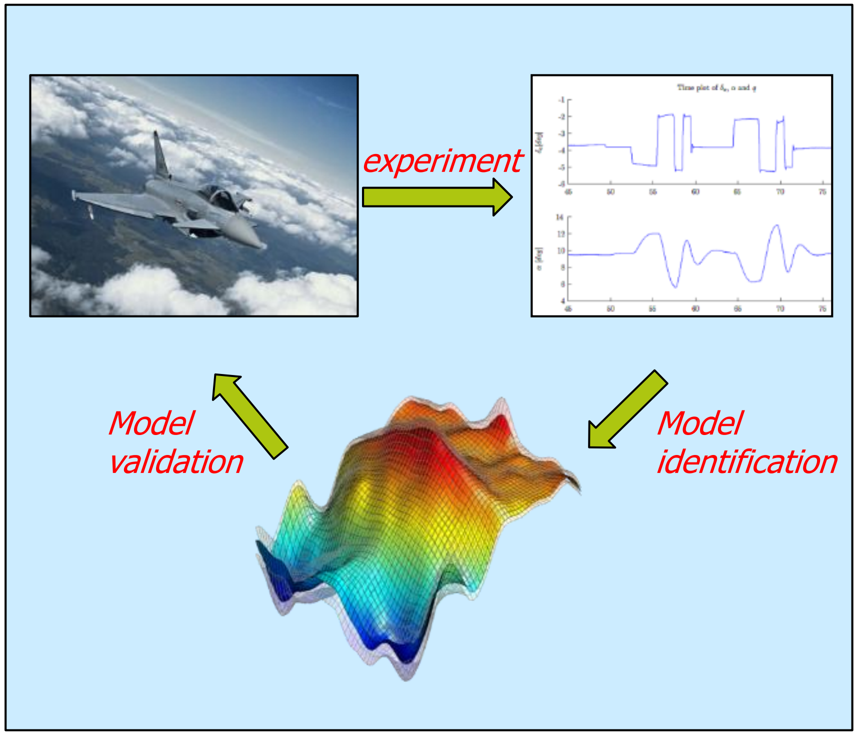
- Plant analysis
- Experiment design and execution
- Data logging and pre-processing

Model identification phase

- State estimation
- Model structure definition
- Parameter estimation

Model validation phase

- Model validation



State Estimation: Motivation

The necessity of state estimation:

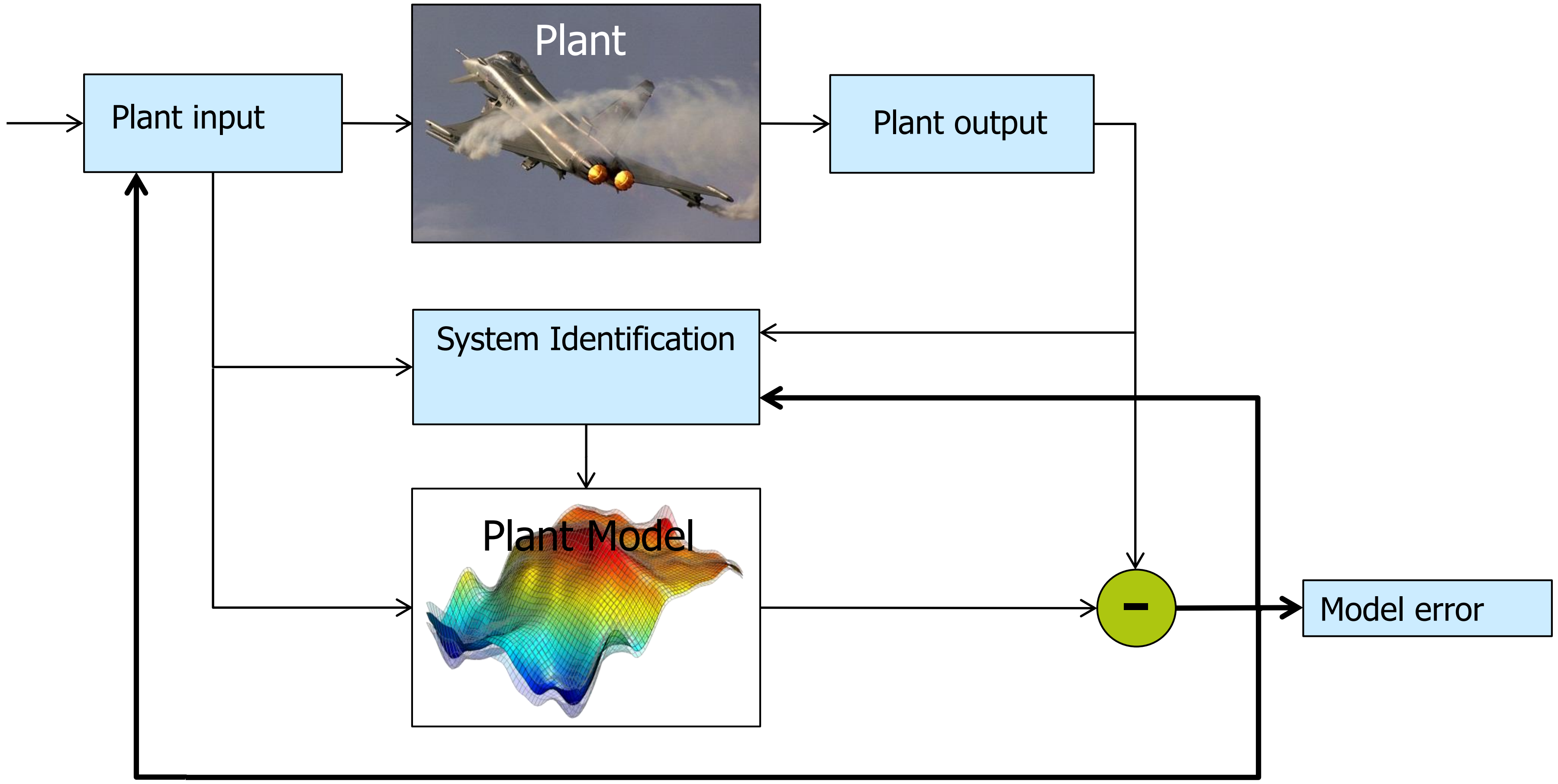
1. Process and measurement noise leads to biased estimates for state variables (e.g. position, velocity, rotational rate, attitude, aerodynamic angles, disturbances, stability and control derivatives, sensor errors and calibration parameters, vehicle mass and inertia properties), which in turn lead to biased estimators, and consequently, reduce the quality of the resulting models (*state reconstruction*).

2. Certain states cannot be measured directly, but must be reconstructed from other sensor sources (*state reconstruction*).

3. Accuracy of state estimates can be improved by combining different sensors (*sensor fusion*).

State Estimation: Motivation (1)

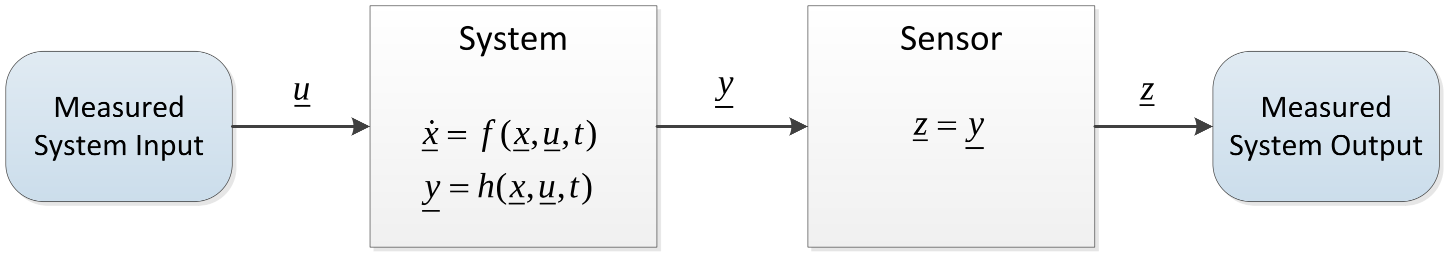
Sensor measurements are biased, resulting biased SysID results...



State Estimation: Motivation (1)

The hypothetical "perfect" **deterministic** system:

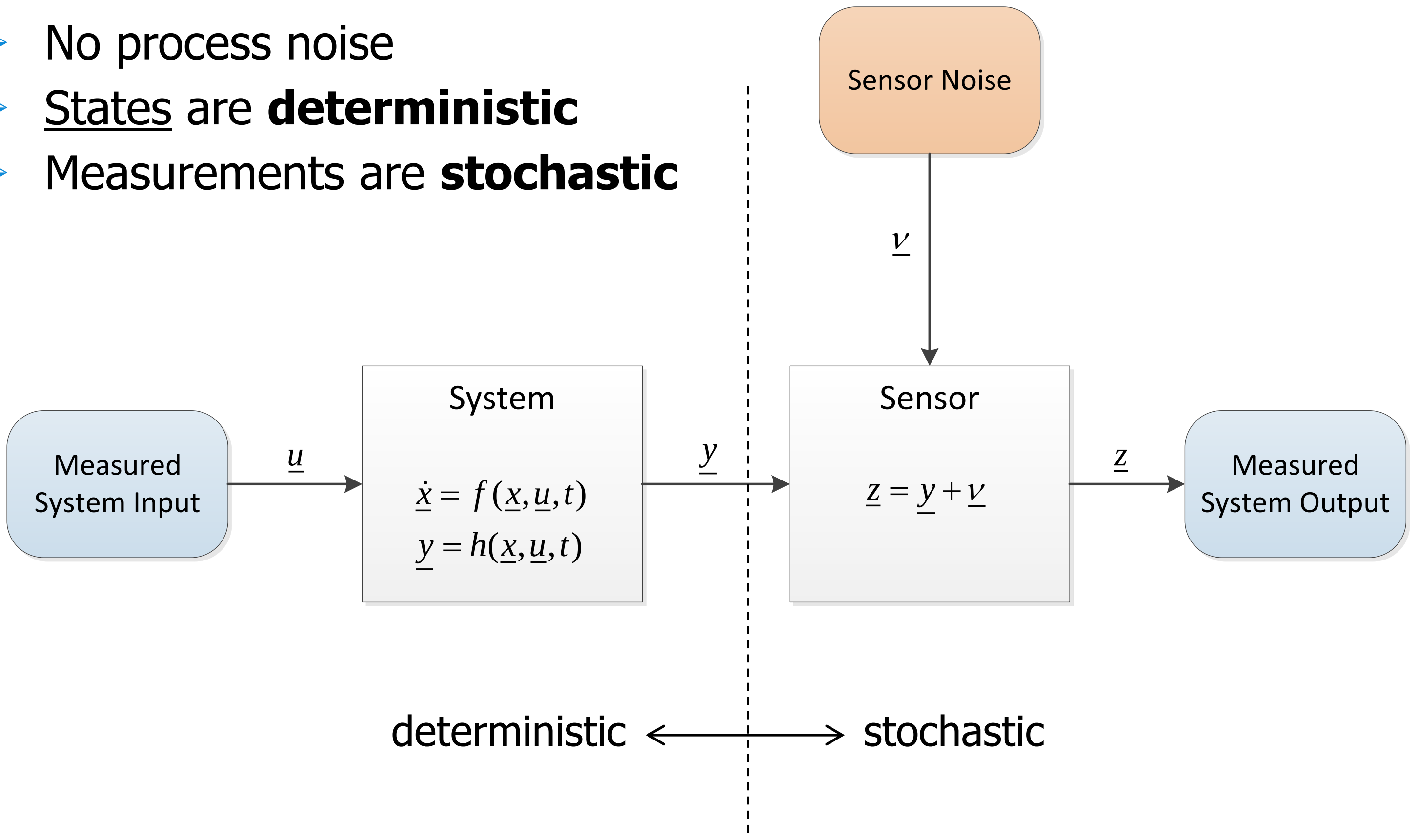
- No sensor noise
- No process noise
- <u>Measurements and states</u> are **deterministic**


Measured System Input
$\underline{u}$
System
$\dot{\underline{x}} = f(\underline{x},\underline{u},t)$
$\underline{y} = h(\underline{x},\underline{u},t)$
$\underline{y}$
Sensor
$\underline{z} = \underline{y}$
$\underline{z}$
Measured System Output

State Estimation: Motivation (1)

The process-noise free system:

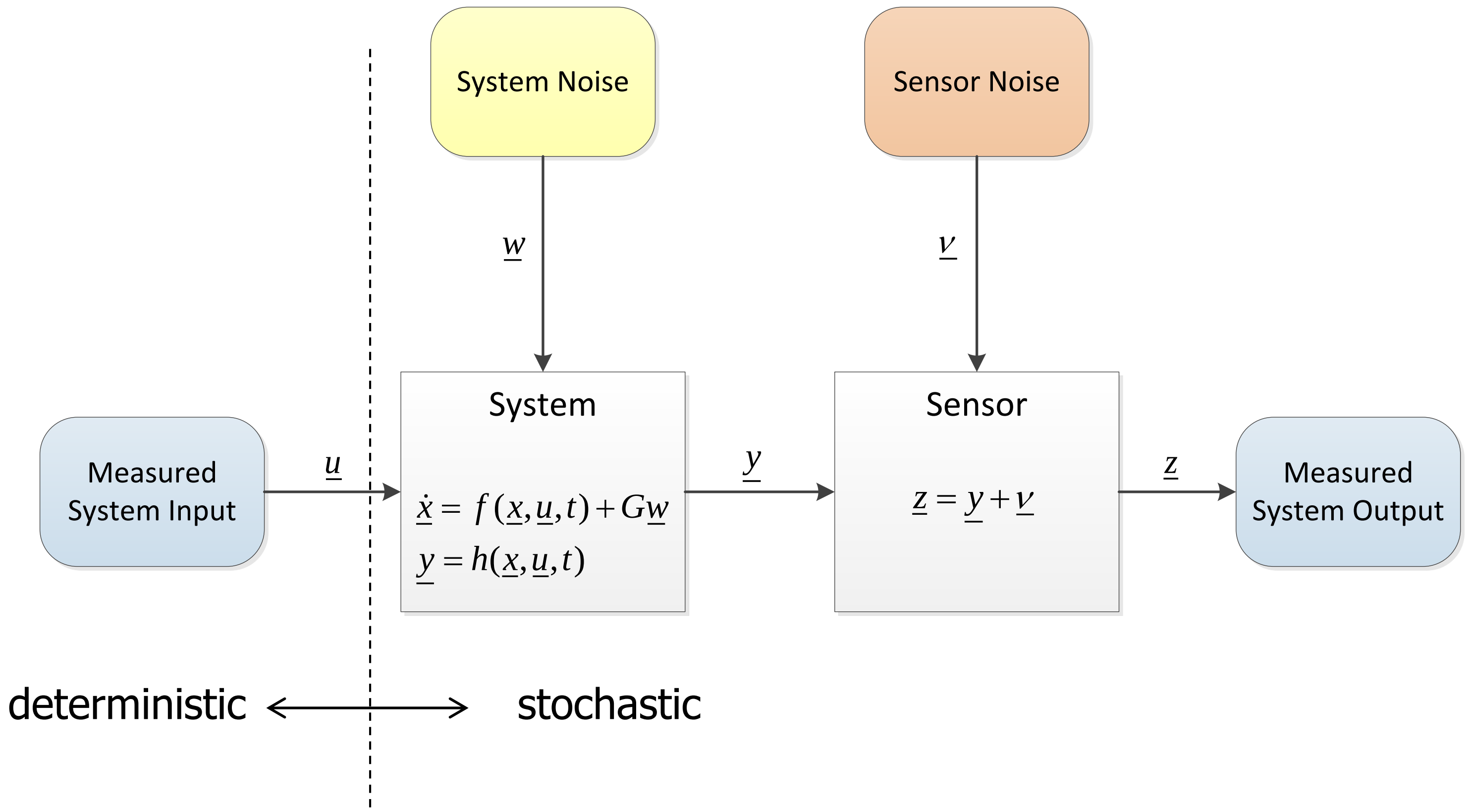
- No process noise
- States are **deterministic**
- Measurements are **stochastic**


Sensor Noise
$\underline{v}$
Measured System Input
$\underline{u}$
System
$\dot{\underline{x}} = f(\underline{x},\underline{u},t)$
$\underline{y} = h(\underline{x},\underline{u},t)$
$\underline{y}$
Sensor
$\underline{z} = \underline{y} + \underline{v}$
$\underline{z}$
Measured System Output
deterministic
stochastic

State Estimation: Motivation (1)

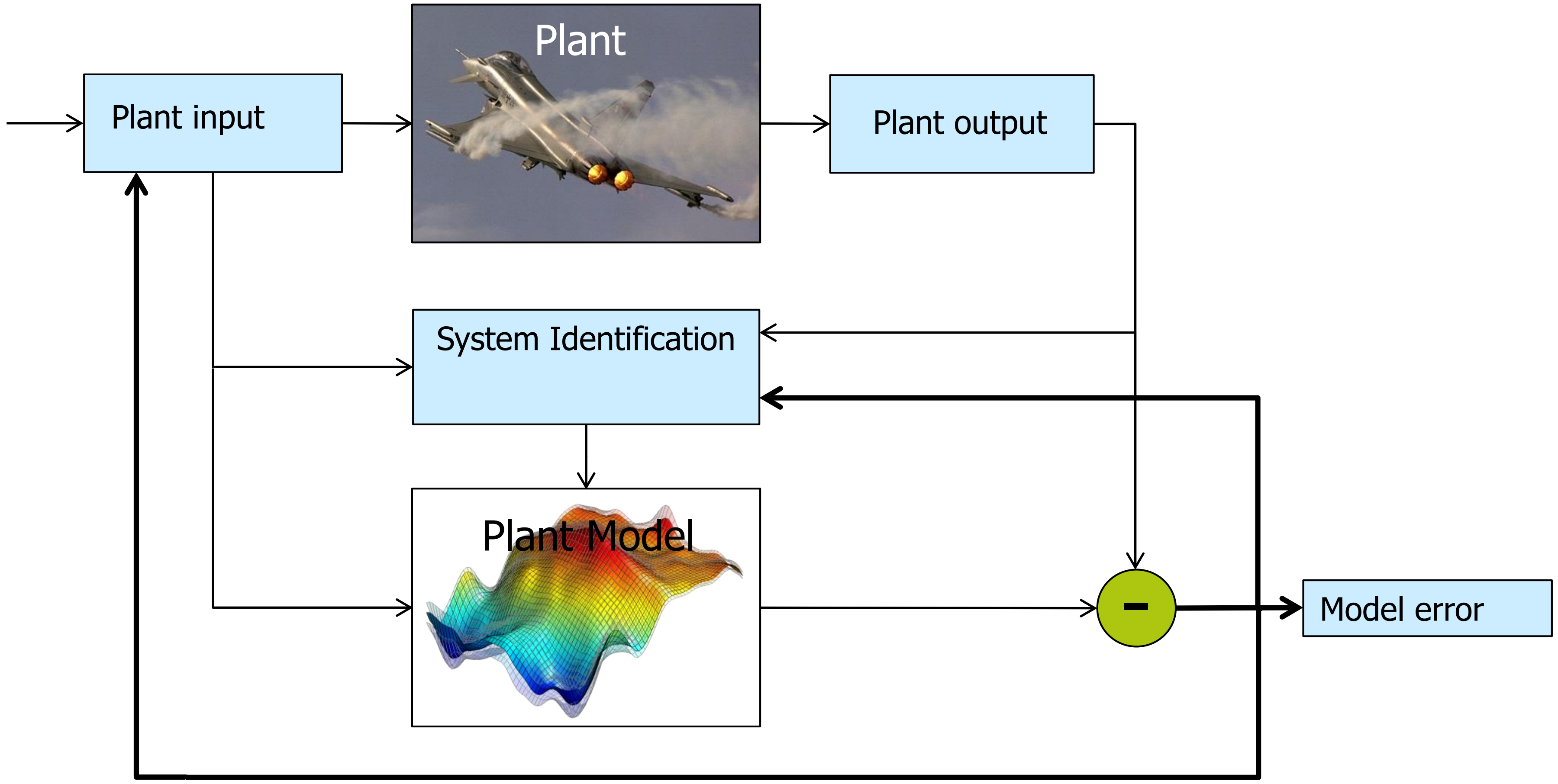
The realistic system:

- Measurements and states are **stochastic**



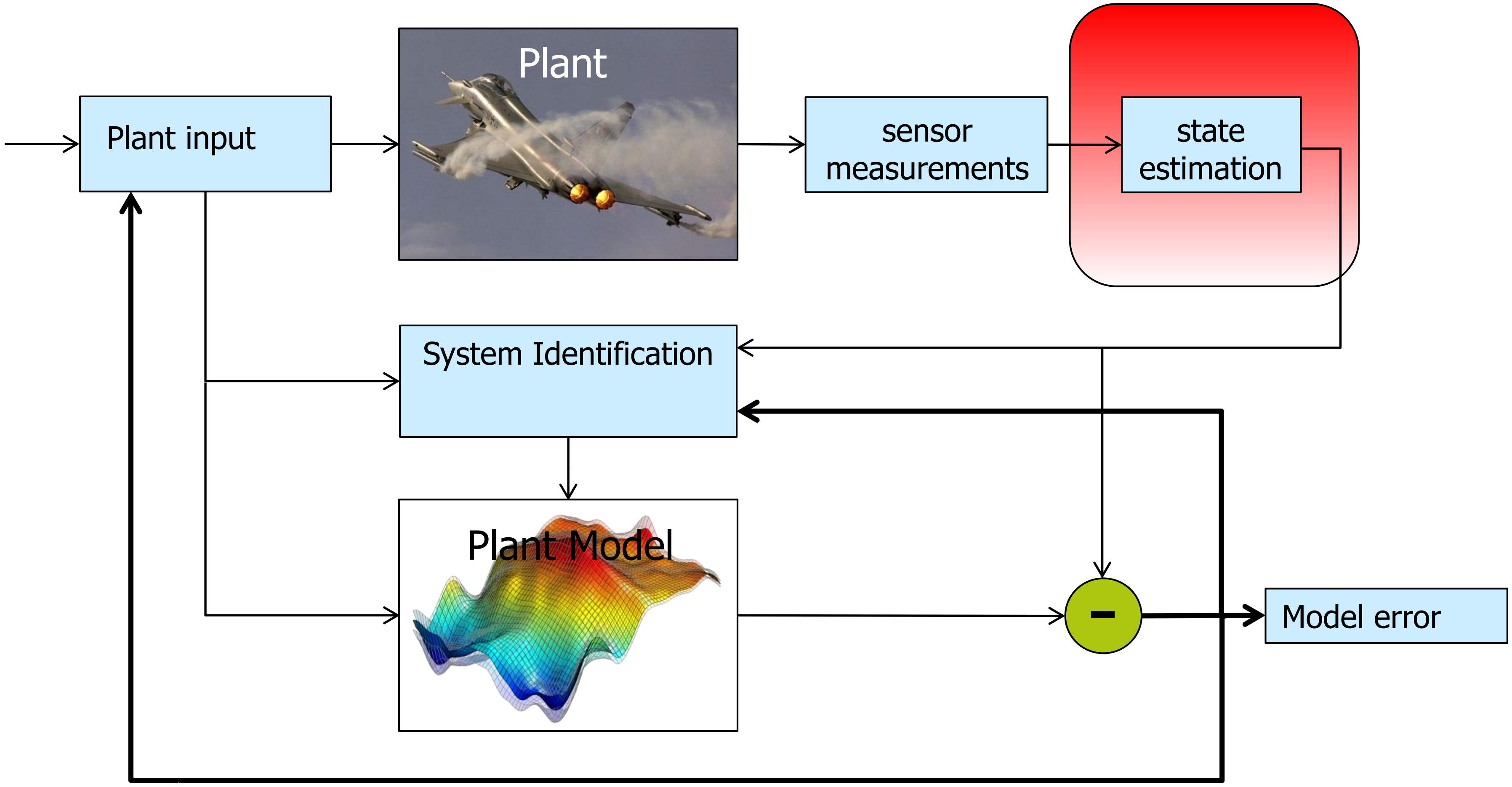
State Estimation: Motivation (1)

Sensor biases are present in the 'Plant output' block



State Estimation: Motivation (1)

Sensor biases can be estimated using state estimator...



State Estimation: Motivation (2)

Some states cannot be measured directly

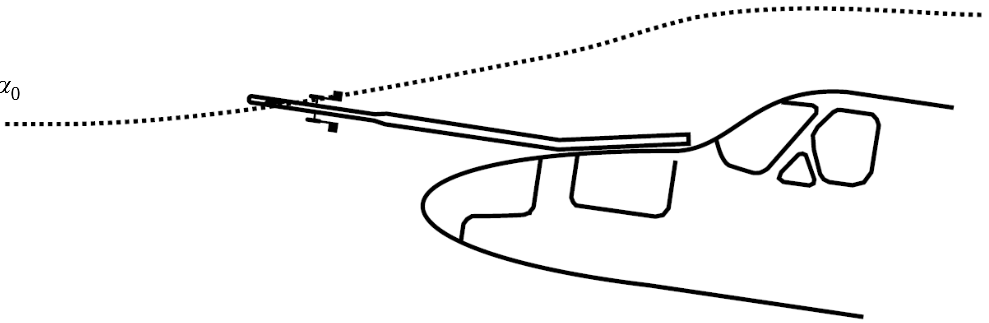
- States like the true (geometric) angle of attack and angle of sideslip are not measurable directly.

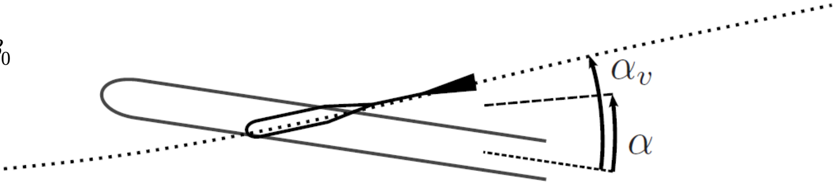
$$\alpha_v = (1 + C_{\alpha_{up}})\alpha + \frac{(q+\lambda_q)x_{v_\alpha}}{V} + C_{\alpha_0}$$

$$\Rightarrow \quad \alpha = \arctan\left(\frac{w}{u}\right)$$

$$\beta_v = (1 + C_{\beta_{si}})\beta - \frac{(r+\lambda_r)x_{v_\beta}}{V} + \frac{(p+\lambda_p)z_{v_\beta}}{V} + C_{\beta_0}$$

$$\Rightarrow \quad \beta = \arctan\left(\frac{v^2}{\sqrt{u^2+w^2}}\right)$$





State Estimation: Motivation (3)

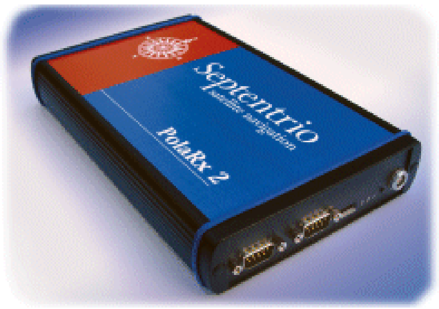
State estimates can be improved by combining sensors

- The accuracy of state estimates can often be improved by combining (**fusing**) measurements made by different sensors.

- **For example:**
  Velocities obtained using GPS sensors can be used to improve the accuracy of the velocities obtained from the pitot static tube.

pitot static tube

+



GPS

= **???**

State Estimation Definitions

Definition of system representation

A system model has the following general form:

$$\dot{\underline{x}}(t) = f(\underline{\theta}(t), \underline{x}(t), \underline{u}(t), t) + G(\underline{\theta}(t), t)\underline{w}(t), \qquad \underline{x}(t_0) = \underline{x}_0$$

$$\underline{z}(t) = h(\underline{\theta}, \underline{x}(t), \underline{u}(t), t) + \underline{v}(t)$$

where $\underline{x}$ is the state vector and $\underline{\theta}$ is the parameter vector.

process noise statistics : $E\{\underline{w}(t)\} = 0; \quad E\left\{\underline{w}(t)\underline{w}^T(\tau)\right\} = Q(\underline{\theta}, t)$

sensor noise statistics : $E\{\underline{v}(t_i)\} = 0; \quad E\left\{\underline{v}(t_i)\underline{v}^T(t_j)\right\} = R(\underline{\theta}, t_i)$

joint sensor-process noise statistics : $E\left\{\underline{w}(t)\underline{v}^T(t_i)\right\} = 0,$

State Estimation Definitions

Definitions of state estimation and parameter estimation

State estimation problem:
searching for the best estimate of $\underline{x}$ while $\underline{\theta}$ is known.

Parameter estimation problem:
searching for the best estimate of $\underline{\theta}$ while $\underline{x}$ is known.

Joint State and Parameter estimation problem:
searching for the best estimate of both $\underline{\theta}$ and $\underline{x}$ which are both unknown.

$$\dot{\underline{x}}(t) = f(\underline{\theta}(t), \underline{x}(t), \underline{u}(t), t) + G(\underline{\theta}(t), t)\underline{w}(t)$$

$$\underline{z}(t) = h(\underline{\theta}, \underline{x}(t), \underline{u}(t), t) + \underline{v}(t)$$

with $\underline{x}$ the state vector and $\underline{\theta}$ the parameter vector.

The Kalman Filter

The Kalman filter is perhaps the most important, most influential element of systems theory ever invented…

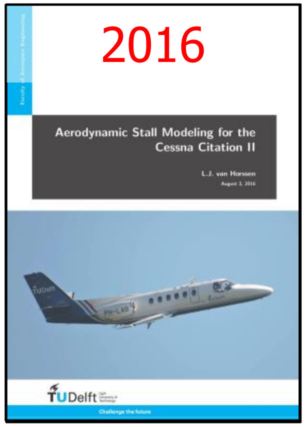
$$\begin{tikzcd}
\boxed{\text{1. One-step ahead prediction}} \arrow[d] \\
\boxed{\text{2. Covariance matrix of state prediction error}} \arrow[d] \\
\boxed{\text{3. Kalman gain calculation}} \arrow[d] \\
\boxed{\text{4. Measurement update}} \arrow[d] \\
\boxed{\text{5. Covariance matrix of state estimation error}}
\end{tikzcd}$$

The Kalman Filter

Many recent MSc/PhD projects at C&S make use of Kalman filters!

Citation/stall modeling

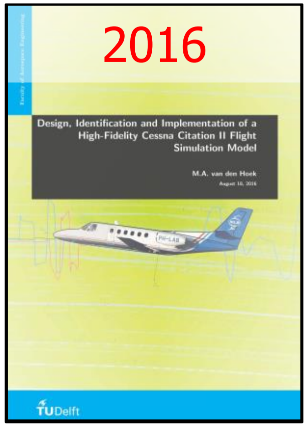
Fault detection


2016

Aerodynamic Stall Modeling for the Cessna Citation II

L.J. van Horssen

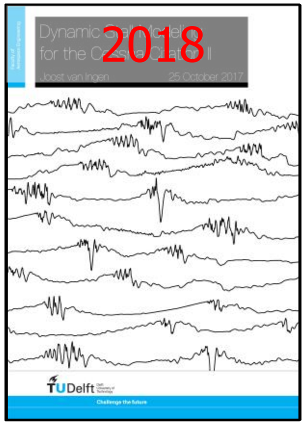
August 3, 2016


2016

Design, Identification and Implementation of a High-Fidelity Cessna Citation II Flight Simulation Model

M.A. van den Hoek

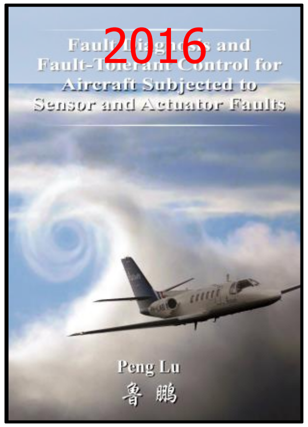
August 16, 2016


2018

Dynamic Stall Model for the Cessna Citation II

Joost van Ingen

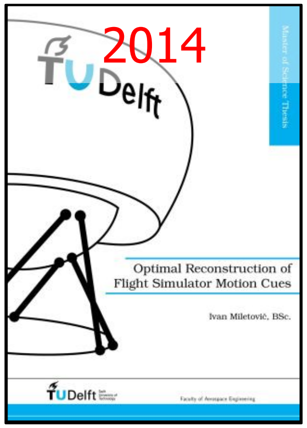
25 October 2017


2016

Fault Diagnosis and Fault-Tolerant Control for Aircraft Subjected to Sensor and Actuator Faults

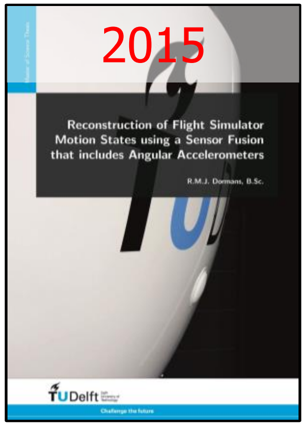
Peng Lu

鲁 鹏


2014

Optimal Reconstruction of Flight Simulator Motion Cues

Ivan Miletović, BSc.


2015

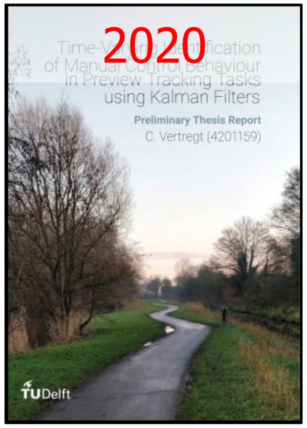
Reconstruction of Flight Simulator Motion States using a Sensor Fusion that includes Angular Accelerometers

R.M.J. Dormans, B.Sc.

2016

Towards Understanding Human Adaptation: Time-Varying Parameter Estimation Utilizing the Unscented Kalman Filter

J. Rojer


2020

Time-Varying Identification of Manual Control Behaviour in Preview Tracking Tasks using Kalman Filters

Preliminary Thesis Report

C. Vertregt (4201159)

SIMONA state reconstruction

Time-varying pilot identification

The Kalman Filter

Some facts on the Kalman filter

- The Kalman filter, in one form or the other, is instrumental in virtually every signal processing related application, from the phase-locked-loop in FM radio systems to cruise missile navigation systems.

- The working principle of the Kalman filter is the calculation of a weighted average between the measured and the predicted state, where the weight (Kalman gain!) depends on the uncertainty in the measurement; the higher this uncertainty, the lower the weight.

- The Kalman filter was (co-)invented by Rudolf Kalman in 1958 when he was working on the Apollo project navigational computer.

Rudolf Kalman

The Kalman Filter

- The **Kalman Filter** (KF) is an optimal linear filter, but it works only for linear systems of the form:

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} + G\underline{w}$$
$$\underline{z} = C\underline{x} + D\underline{u} + \underline{v}$$

- The **Extended Kalman Filter** (EKF) is a non-optimal extension of the Kalman filter to nonlinear systems of the form:

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}) + G\underline{w}$$
$$\underline{z} = h(\underline{x}, \underline{u}) + \underline{v}$$

- The **Iterated Extended Kalman Filter** (IEKF) is an semi-optimal iterative extension of the EKF for nonlinear systems.

- **Many, many more** Kalman filters exist: *Unscented, Hybrid, Steady-State, Central-Difference, Frequency-Weighted, Ensemble, Fast, Schmidt, Dual-State…*

The Kalman Filter

Working principle of the Kalman filter

- The working principle of the Kalman filter is the calculation of a **<u>weighted average between the measured and the predicted state</u>**:

$$x_{estimated} = x_{predicted} + K \cdot \left(z_{measured} - z_{predicted} \right)$$

- This weight (*K* = the Kalman gain!) depends on the uncertainties in the <u>prediction</u> and in the <u>measurement</u>; the higher the measurement uncertainty and the lower the prediction uncertainty, the lower this weight.

- So, the big question is: what is the <u>optimal</u> *K*, and how do we find it?

Intermezzo: Cost Functions

Intermezzo: What is a cost function?

Lets say we have a measurement y, and a model $f(x,\theta)$ then the error between the measurement and the model is given by:

$$\varepsilon = y - f(\underset{\boxed{\text{states}}}{\underset{\nearrow}{x}},\underset{\boxed{\text{parameters}}}{\underset{\nwarrow}{\theta}})$$

The goal is to find values for θ which make ε as small as possible...

How do we proceed?

We want to **minimize** ε by changing θ; this implies that we have an **optimization problem**!

Intermezzo: Cost Functions

So how do we penalize the error $\varepsilon = y - f(x,\theta)$?

We sum up all values of ε for all values of x for a given set of parameters θ in what we call a **cost function**:

$$J(x,\theta) = \sum_{i=1}^{N} y_i - f(x_i,\theta)$$

The example above is a linear cost function; it just sums up the all errors between the objective function (data) and the model.

The optimization problem then has the following notation:

$$\hat{\theta} = \arg\min \; J(x,\theta)$$

with $\hat{\theta}$ a **parameter estimator** for θ!

Intermezzo: Cost Functions

Some more examples of cost functions:

$$J(x,\theta) = \sum_{i=1}^{N} y_i - f(x_i,\theta), \qquad \text{(linear)}$$

$$J(x,\theta) = \sum_{i=1}^{N} \left(y_i - f(x_i,\theta)\right)^2, \qquad \text{(quadratic)}$$

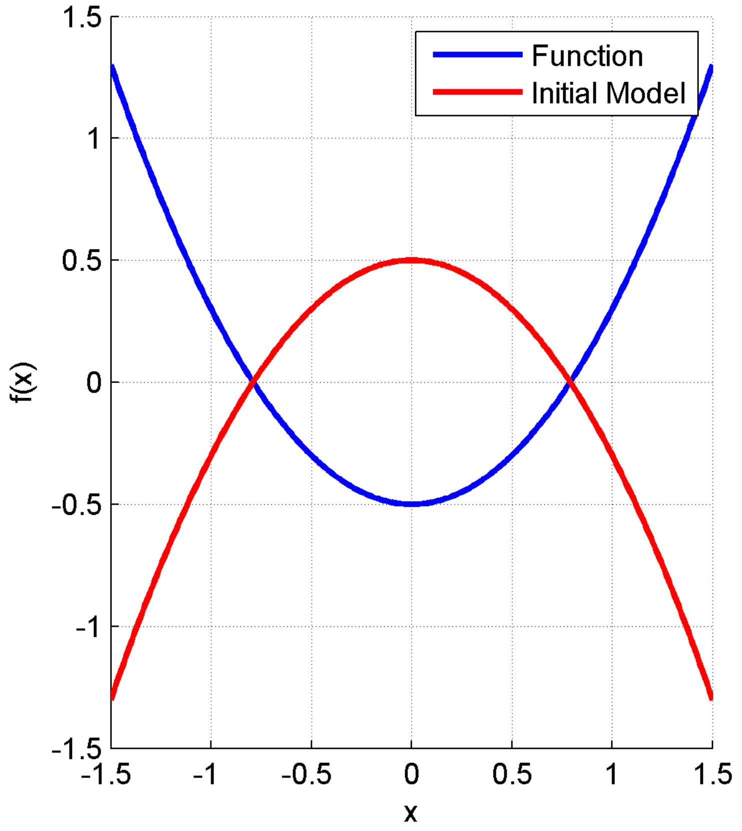
$$J(x,\theta) = \sum_{i=1}^{N} \left|y_i - f(x_i,\theta)\right|, \qquad \text{(absolute value)}$$

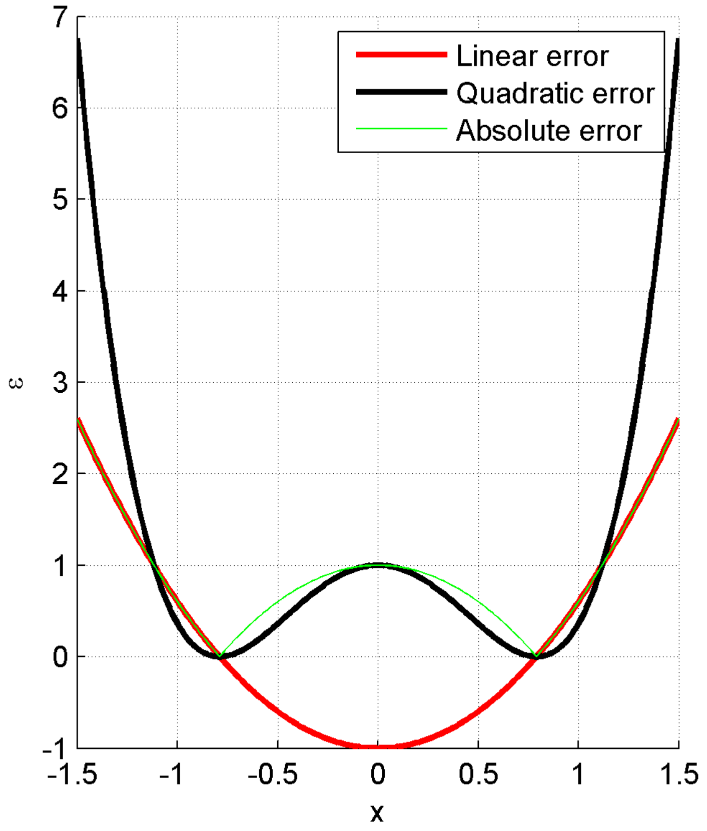
So how do we choose a cost function? Which ones will work, and which ones won’t? How do we find the optimum value for θ?

Intermezzo: Cost Functions

Cost function comparison:

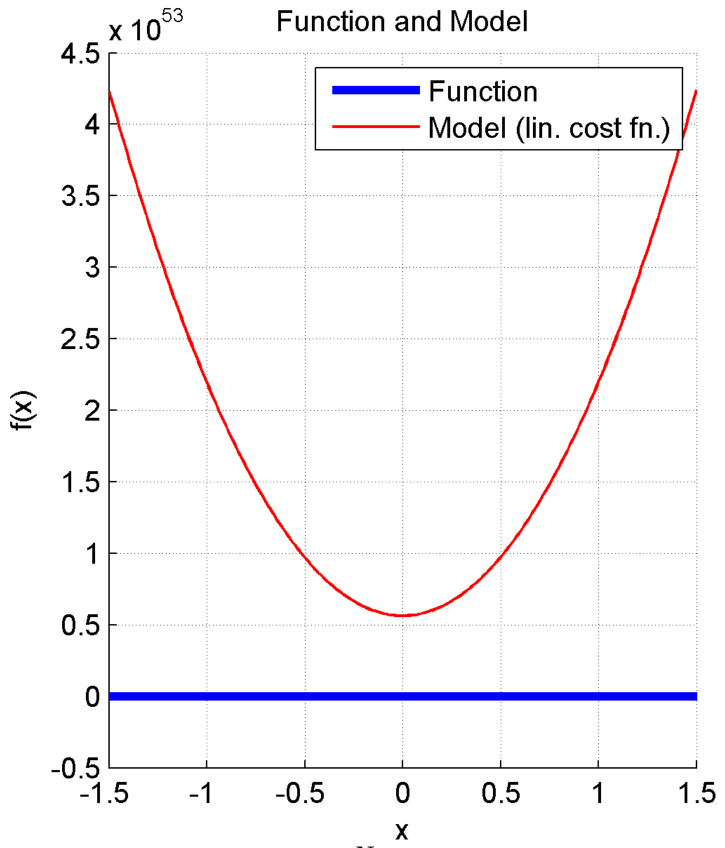
$$J(x,\theta)=\sum_{i=1}^{N} y_i - f(x_i,\theta), \qquad J(x,\theta)=\sum_{i=1}^{N}\left(y_i - f(x_i,\theta)\right)^2, \quad J(x,\theta)=\sum_{i=1}^{N}\left|y_i - f(x_i,\theta)\right|$$

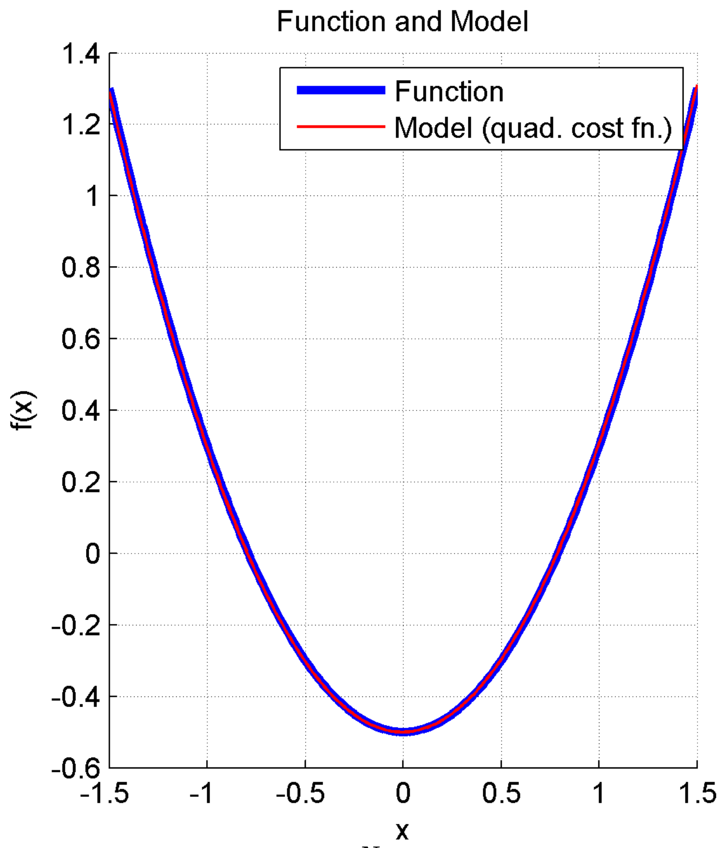




Intermezzo: Cost Functions

Resulting models:





$$\cancel{J(x \ldots \sum_{i=1}^{N} y_i \ldots x_i,\theta)}$$

$$J(x,\theta)=\sum_{i=1}^{N}\left(y_i - f(x_i,\theta)\right)^2$$

Intermezzo: Cost Functions

So given a cost function $J(x,\theta)$ *how do we find the optimal value for* θ*?*

In fact, there are many ways to minimize a cost function. For example, first derivative, Newton, Newton-Gauss, gradient descent, conjugate gradient, Lagrange multiplier, Nelder-Mead Simplex, etc.

One of the most used methods is the **single-step first-derivative method**:

$$\frac{\partial J(x,\theta)}{\partial\theta} = 0, \;\; \theta = \hat{\theta}$$

For the quadratic cost function we then get:

$$\frac{\partial \sum_{i=1}^{N}\left(y_i - f(x_i,\theta)\right)^2}{\partial\theta} = \sum_{i=1}^{N} 2\left(y_i - f(x_i,\theta)\right)\frac{\partial f(x_i,\theta)}{\partial\theta} = 0, \quad \theta = \hat{\theta}$$

Derivation of the linear Kalman Filter

The general linear, input affine, time-varying state space model is:

$$\dot{\underline{x}}(t) = F(t)\underline{x}(t) + B(t)\underline{u}(t) + G(t)\underline{w}(t), \qquad \underline{x}(t_0) = \underline{x}_0$$

$$\underline{z}(t) = H(t)\underline{x}(t) + D(t)\underline{u}(t) + \underline{v}(t) \qquad t = t_i, \quad i = 1, 2, \ldots$$

with:

$\underline{x}(t)$	$=$ state vector of dimension n	$\underline{F}(t)$	$=$ System matrix, dimension $n \times n$
$\underline{x}_0$	$=$ initial condition of the state	$\underline{B}(t)$	$=$ Input matrix, dimension $n \times l$
$\underline{u}(t)$	$=$ control input, dimension l	$\underline{G}(t)$	$=$ System noise input matrix, dimension $n \times m$
$\underline{w}(t)$	$=$ system noise, dimension m	$\underline{H}(t)$	$=$ Observation matrix, dimension on $p \times n$
$\underline{v}(t)$	$=$ measurement noise, dimension p	$\underline{D}(t)$	$=$ Feedforward matrix, dimension $p \times l$
$\underline{z}(t)$	$=$ measurement vector, dimension p		

Derivation of the linear Kalman Filter

Additionally, $\underline{w}(t)$ is a continuous time white noise process, $\underline{v}(t_i)$ is a discrete time white noise sequence, and $\underline{w}(t)$ and $\underline{v}(t_i)$ are mutually uncorrelated at all $t=t_i$, $i= 1,2,\dots$.

We then have the following noise statistics:

system noise mean:	$E\left\{\underline{w}(t)\right\} = 0$
system noise covariance:	$E\left\{\underline{w}(t)\underline{w}^T(\tau)\right\} = Q(t)$
sensor noise mean:	$E\left\{\underline{v}(t_i)\right\} = 0$
sensor noise covariance:	$E\left\{\underline{v}(t_i)\underline{v}^T(t_j)\right\} = R(t_i)$
system-sensor cross-covariance:	$E\left\{\underline{w}(t)\underline{v}^T(t_i)\right\} = 0,$

The initial state $\underline{x}_0$ is a multivariate Gaussian with mean $\underline{\mu}_x(0)$ and covariance $P_x(0)$:

$$\underline{x}_0 \sim N\left(\underline{x}(0); \underline{\mu}_x(0), P_x(0)\right)$$

Derivation of the linear Kalman Filter

Some definitions:

Real states:

$\underline{x}_k$: Current real system state

$\underline{x}_{k+1}$: One-step-ahead real system state

Predicted/Estimated states:

$\hat{\underline{x}}_{k,k}$: Current optimal estimated system state

$\hat{\underline{x}}_{k+1,k}$: One-step-ahead state prediction

$\hat{\underline{x}}_{k+1,k+1}$: One-step-ahead optimal state estimation

Prediction/estimation errors

$\hat{\underline{\varepsilon}}_{k+1,k} = \hat{\underline{x}}_{k+1,k} - \underline{x}_{k+1}$: State prediction error

$\hat{\underline{\varepsilon}}_{k+1,k+1} = \hat{\underline{x}}_{k+1,k+1} - \underline{x}_{k+1}$: State estimation error

Derivation of the linear Kalman Filter

Before continuing, we have to discretize the continuous time system:

$$\underline{\dot{x}}(t) = F(t)\underline{x}(t) + B(t)\underline{u}(t) + G(t)\underline{w}(t),$$

$$\underline{z}(t) = H(t)\underline{x}(t) + D(t)\underline{u}(t) + \underline{v}(t)$$

For this, we can use the built-in Matlab function `c2d(F, B, deltaT)`:

```
[Phi, Psi] = c2d(F, B, deltaT);
[Phi, Gamma] = c2d(F, G, deltaT);
```

Matlab tip!

Which results in the discretized linear system:

$$\underline{x}_{k+1} = \boxed{\Phi_{k+1,k}}\underline{x}_k + \Psi_{k+1,k}\underline{u}_k + \Gamma_{k+1,k}\underline{w}_{d,k}$$

$$\underline{z}_{k+1} = H_{k+1}\underline{x}_{k+1} + D_{k+1}\underline{u}_{k+1} + \underline{v}_{k+1}$$

Confusing notation! $\hat{\underline{x}}_{k+1,k}$

In which $\Phi_{k+1,k}$ is the system transition matrix, $\Psi_{k+1,k}$ the input distribution matrix, and $\Gamma_{k+1,k}$ the noise input matrix.

Derivation of the linear Kalman Filter

After discretization the following basic linear, (possibly time-varying), discrete time dynamic system, which is characterized by an nx1 state vector and an mx1 measurement vector, will be found:

$$\underline{x}_{k+1} = \Phi_{k+1,k}\underline{x}_k + \Psi_{k+1,k}\underline{u}_k + \Gamma_{k+1,k}\underline{w}_{d,k}$$

$$\underline{z}_{k+1} = H_{k+1}\underline{x}_{k+1} + \cancel{D_{k+1}\underline{u}_{k+1}} + \underline{v}_{k+1}$$

We assume no feedthrough of inputs to outputs, which is true for most realistic systems!

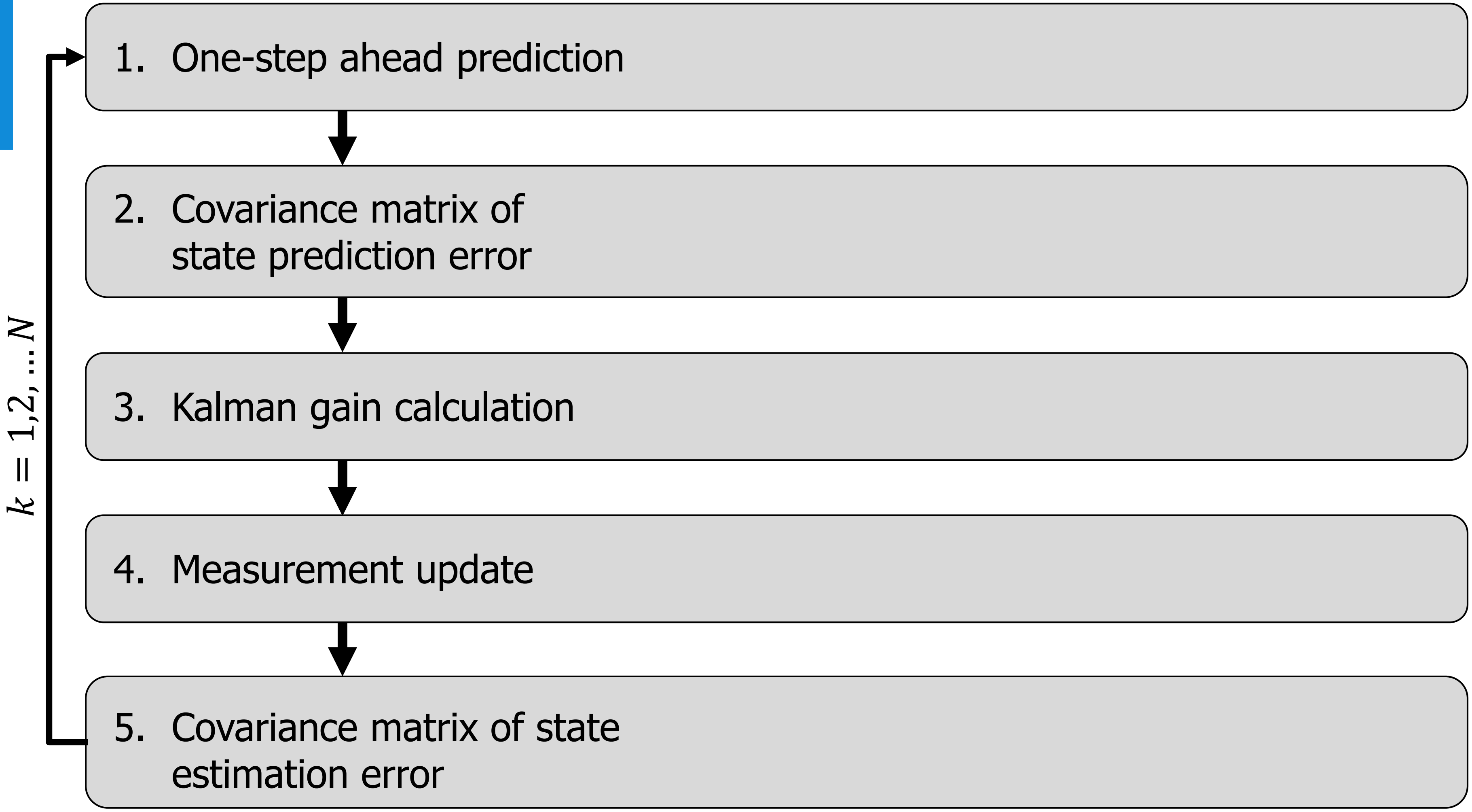
The discretized noise characteristics are as follows:

$$E\left\{\underline{w}_{d,i}\right\}=0; \quad E\left\{\underline{w}_{d,i}\underline{w}_{d,j}^T\right\} = Q_{d,i}$$

$$E\left\{\underline{v}_i\right\}=0; \quad E\left\{\underline{v}_i\underline{v}_j^T\right\} = R_i$$

$$E\left\{\underline{w}_{d,i}\underline{v}_j^T\right\} = 0$$

The Linear Kalman Filter


1. One-step ahead prediction
2. Covariance matrix of state prediction error
3. Kalman gain calculation
4. Measurement update
5. Covariance matrix of state estimation error
$k = 1,2,\ldots N$

Derivation of the linear Kalman Filter

We can now derive the expected value and associated error covariance matrix of the state $\underline{x}_{k+1}$ as follows:

$$E\left\{\underline{x}_{k+1}\right\} = \underline{\hat{x}}_{k+1,k} = \Phi_{k+1,k} E\left\{\underline{x}_k\right\} + \Psi_{k+1,k}\underline{u}_k + E\left\{\Gamma_{k+1,k}\underline{w}_{d,k}\right\}$$

consider the following statistics:

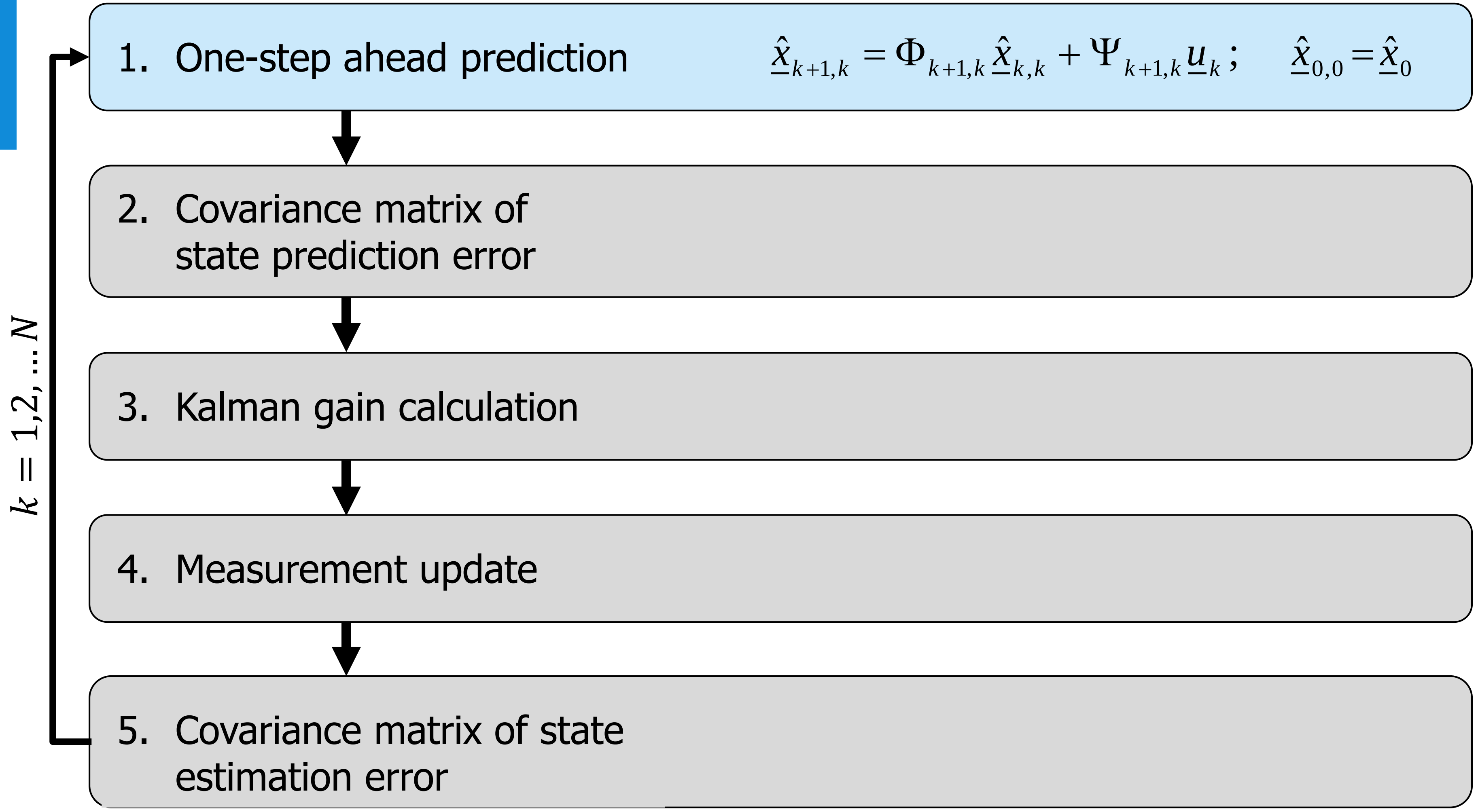
$$E\left\{\underline{x}_0\right\} = \underline{\hat{x}}_{0,0}; \quad E\left\{\left(\underline{\hat{x}}_{0,0} - \underline{x}_0\right)\left(\underline{\hat{x}}_{0,0} - \underline{x}_0\right)^T\right\} = P_{0,0}$$

remember that the process noise was considered to be white: $E\{\underline{w}_{d,i}\} = 0$

Therefore, we have for one-step ahead prediction of the state:

$$\boxed{\underline{\hat{x}}_{k+1,k} = \Phi_{k+1,k}\underline{\hat{x}}_{k,k} + \Psi_{k+1,k}\underline{u}_k}$$

The Linear Kalman Filter


1. One-step ahead prediction $\underline{\hat{x}}_{k+1,k} = \Phi_{k+1,k}\underline{\hat{x}}_{k,k} + \Psi_{k+1,k}\underline{u}_k\,; \quad \underline{\hat{x}}_{0,0} = \underline{\hat{x}}_0$
2. Covariance matrix of state prediction error
3. Kalman gain calculation
4. Measurement update
5. Covariance matrix of state estimation error
$k = 1,2,\ldots N$

Derivation of the linear Kalman Filter

The substitution of

$$\boxed{\underline{\hat{x}}_{k+1,k} = \Phi_{k+1,k}\underline{\hat{x}}_{k,k} + \Psi_{k+1,k}\underline{u}_k}$$

and

$$\underline{x}_{k+1} = \Phi_{k+1,k}\underline{x}_k + \Psi_{k+1,k}\underline{u}_k + \Gamma_{k+1,k}\underline{w}_{d,k}$$

in

$$\underline{\hat{\varepsilon}}_{k+1,k} = \underline{\hat{x}}_{k+1,k} - \underline{x}_{k+1}$$

Results in:

$$\begin{aligned}\underline{\hat{\varepsilon}}_{k+1,k} &= \left(\Phi_{k+1,k}\underline{\hat{x}}_{k,k} + \Psi_{k+1,k}\underline{u}_k\right) - \left(\Phi_{k+1,k}\underline{x}_k + \Psi_{k+1,k}\underline{u}_k + \Gamma_{k+1,k}\underline{w}_{d,k}\right)\\ &= \Phi_{k+1,k}\left(\underline{\hat{x}}_{k,k} - \underline{x}_k\right) - \Gamma_{k+1,k}\underline{w}_{d,k}\\ &= \Phi_{k+1,k}\underline{\hat{\varepsilon}}_{k,k} - \Gamma_{k+1,k}\underline{w}_{d,k}\end{aligned}$$

Derivation of the linear Kalman Filter

Now we substitute the expression we found for the state estimation error:

$$\underline{\hat{\varepsilon}}_{k+1,k} = \Phi_{k+1,k}\,\underline{\hat{\varepsilon}}_{k,k} - \Gamma_{k+1,k}\,\underline{w}_{d,k}$$

in the expression for the state estimation error covariance matrix:

$$P_{k+1,k} = E\left\{\left(\underline{x}_{k+1} - \underline{\hat{x}}_{k+1,k}\right)\left(\underline{x}_{k+1} - \underline{\hat{x}}_{k+1,k}\right)^T\right\}$$

the result of the substitution is:

$$\begin{aligned} P_{k+1,k} &= E\left\{\underline{\hat{\varepsilon}}_{k+1,k}\,\underline{\hat{\varepsilon}}_{k+1,k}^T\right\} = E\left\{\left(\Phi_{k+1,k}\underline{\hat{\varepsilon}}_{k,k} - \Gamma_{k+1,k}\underline{w}_{d,k}\right)\left(\Phi_{k+1,k}\underline{\hat{\varepsilon}}_{k,k} - \Gamma_{k+1,k}\underline{w}_{d,k}\right)^T\right\} \\ &= \Phi_{k+1,k}E\left\{\underline{\hat{\varepsilon}}_{k,k}\underline{\hat{\varepsilon}}_{k,k}^T\right\}\Phi_{k+1,k}^T - \Phi_{k+1,k}E\left\{\underline{\hat{\varepsilon}}_{k,k}\underline{w}_{d,k}^T\right\}\Gamma_{k+1,k}^T - \Gamma_{k+1,k}E\left\{\underline{w}_{d,k}\underline{\hat{\varepsilon}}_{k,k}^T\right\}\Phi_{k+1,k}^T \\ &\quad + \Gamma_{k+1,k}E\left\{\underline{w}_{d,k}\underline{w}_{d,k}^T\right\}\Gamma_{k+1,k}^T \end{aligned}$$

Derivation of the linear Kalman Filter

We now have:

$$P_{k+1,k} = \Phi_{k+1,k} E\left\{\underline{\hat{\varepsilon}}_{k,k}\underline{\hat{\varepsilon}}_{k,k}^T\right\}\Phi_{k+1,k}^T - \Phi_{k+1,k} E\left\{\underline{\hat{\varepsilon}}_{k,k}\underline{w}_{d,k}^T\right\}\Gamma_{k+1,k}^T - \Gamma_{k+1,k} E\left\{\underline{w}_{d,k}\underline{\hat{\varepsilon}}_{k,k}^T\right\}\Phi_{k+1,k}^T + \Gamma_{k+1,k} E\left\{\underline{w}_{d,k}\underline{w}_{d,k}^T\right\}\Gamma_{k+1,k}^T$$

which can be simplified further by making use of the statistics

$$E\left\{\underline{\hat{\varepsilon}}_{k,k}\underline{w}_{d,k}^T\right\} = 0, \qquad E\left\{\underline{w}_{d,k}\underline{\hat{\varepsilon}}_{k,k}^T\right\} = 0,$$

$$E\left\{\underline{\hat{\varepsilon}}_{k,k}\underline{\hat{\varepsilon}}_{k,k}^T\right\} = P_{k,k}, \quad E\left\{\underline{w}_{d,k}\underline{w}_{d,k}^T\right\} = Q_{d,k}$$

Finally, we get for the state prediction covariance matrix update:

$$\boxed{P_{k+1,k} = \Phi_k P_{k,k}\Phi_k^T + \Gamma_{k+1,k} Q_{d,k}\Gamma_{k+1,k}^T}$$

The Linear Kalman Filter

$k = 1,2,\dots N$

1. One-step ahead prediction $\quad \underline{\hat{x}}_{k+1,k} = \Phi_{k+1,k}\,\underline{\hat{x}}_{k,k} + \Psi_{k+1,k}\,\underline{u}_k\,; \quad \underline{\hat{x}}_{0,0} = \underline{\hat{x}}_0$
2. Covariance matrix of state prediction error $\quad P_{k+1,k} = \Phi_{k+1,k} P_{k,k} \Phi^T_{k+1,k} + \Gamma_{k+1,k} Q_{d,k} \Gamma^T_{k+1,k}\,; \quad P_{0,0} = P_0$
3. Kalman gain calculation
4. Measurement update
5. Covariance matrix of state estimation error

Derivation of the linear Kalman Filter

Define the expectation and the associated error covariance matrix of the state $\underline{x}_{k+1}$ as:

$$E\left\{\underline{x}_{k+1}\right\} = \hat{\underline{x}}_{k+1,k} \quad \Longrightarrow \quad \text{unbiased estimator for the state } \underline{x}_{k+1}$$

$$E\left\{\left(\underline{x}_{k+1} - \hat{\underline{x}}_{k+1,k}\right)\left(\underline{x}_{k+1} - \hat{\underline{x}}_{k+1,k}\right)^T\right\} = P_{k+1,k} \quad \Longrightarrow \quad \text{state prediction covariance matrix}$$

We must now define a **Cost Function** $J(\hat{\underline{x}}_{k+1,k})$ which must be minimized to find an optimal estimate for the predicted state $\hat{\underline{x}}_{k+1,k}$which we indicate as $\hat{\underline{x}}_{k+1,k+1}$:

$$\hat{\underline{x}}_{k+1,k+1} = \arg\min J(\hat{\underline{x}}_{k+1,k})$$

Derivation of the linear Kalman Filter

Cost function definition is **not trivial!!!**

We will use a **weighted least squares** cost function to determine an optimal estimate for the state $\underline{\hat{x}}_{k+1,k}$:

$$J = \underbrace{\tfrac{1}{2}\left(\underline{x}_{k+1} - \underline{\hat{x}}_{k+1,k}\right)^T P_{k+1,k}^{-1}\left(\underline{x}_{k+1} - \underline{\hat{x}}_{k+1,k}\right)}_{\text{penalizes state prediction error}} + \underbrace{\tfrac{1}{2}\left(\underline{z}_{k+1} - H_{k+1}\underline{x}_{k+1}\right)^T R_{k+1}^{-1}\left(\underline{z}_{k+1} - H_{k+1}\underline{x}_{k+1}\right)}_{\text{penalizes measurement error}}$$

Derivation of the linear Kalman Filter

The Weighted Least Squares cost function is:

$$J = \tfrac{1}{2}\left(\underline{x}_{k+1} - \hat{\underline{x}}_{k+1,k}\right)^T P_{k+1,k}^{-1}\left(\underline{x}_{k+1} - \hat{\underline{x}}_{k+1,k}\right) + \tfrac{1}{2}\left(\underline{z}_{k+1} - H_{k+1}\underline{x}_{k+1}\right)^T R_{k+1}^{-1}\left(\underline{z}_{k+1} - H_{k+1}\underline{x}_{k+1}\right)$$

We want to minimize the Weighted Least Squares cost function as a function of the real one-step-ahead state $\underline{x}_{k+1}$.

Therefore, differentiate J with respect to $\underline{x}_{k+1}$:

$$\frac{\partial J}{\partial \underline{x}_{k+1}} = \left(\underline{x}_{k+1} - \hat{\underline{x}}_{k+1,k}\right)^T P_{k+1,k}^{-1} - \left(\underline{z}_{k+1} - H_{k+1}\underline{x}_{k+1}\right)^T R_{k+1}^{-1} H_{k+1}$$

Derivation of the linear Kalman Filter

The best estimate of $\underline{x}_{k+1}$, indicated as $\hat{\underline{x}}_{k+1,k+1}$ can be found by letting:

$$\frac{dJ}{d\underline{x}_{k+1}} = \left(\hat{\underline{x}}_{k+1,k+1} - \hat{\underline{x}}_{k+1,k}\right)^T P_{k+1,k}^{-1} - \left(\underline{z}_{k+1} - H_{k+1}\hat{\underline{x}}_{k+1,k+1}\right)^T R_{k+1}^{-1} H_{k+1} = 0$$

Now transpose this equation as follows:

$$P_{k+1,k}^{-1}\left(\hat{\underline{x}}_{k+1,k+1} - \hat{\underline{x}}_{k+1,k}\right) - H_{k+1}^T R_{k+1}^{-1}\left(\underline{z}_{k+1} - H_{k+1}\hat{\underline{x}}_{k+1,k+1}\right) = 0$$

Collect terms with $\hat{\underline{x}}_{k+1,k+1}$ on the left, terms with $\hat{\underline{x}}_{k+1,k}$ on the right:

$$\left(P_{k+1,k}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)\hat{\underline{x}}_{k+1,k+1} = P_{k+1,k}^{-1}\hat{\underline{x}}_{k+1,k} + H_{k+1}^T R_{k+1}^{-1}\underline{z}_{k+1}$$

Derivation of the linear Kalman Filter

Adding and subtracting $H_{k+1}^T R_{k+1}^{-1} H_{k+1} \underline{\hat{x}}_{k+1,k}$ to the right hand side of

Trick!

$$\left(P_{k+1,k}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)\underline{\hat{x}}_{k+1,k+1} = P_{k+1,k}^{-1}\underline{\hat{x}}_{k+1,k} + H_{k+1}^T R_{k+1}^{-1}\underline{z}_{k+1}$$

leads to:

$$\begin{aligned}\left(P_{k+1,k}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)\underline{\hat{x}}_{k+1,k+1} = P_{k+1,k}^{-1}\underline{\hat{x}}_{k+1,k} + H_{k+1}^T R_{k+1}^{-1}\underline{z}_{k+1} + \\ + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\underline{\hat{x}}_{k+1,k} - H_{k+1}^T R_{k+1}^{-1} H_{k+1}\underline{\hat{x}}_{k+1,k}\end{aligned}$$

which can be rewritten as follows:

$$\begin{aligned}\left(P_{k+1}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)\underline{\hat{x}}_{k+1,k+1} = \left(P_{k+1}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)\underline{\hat{x}}_{k+1,k} + \\ + H_{k+1}^T R_{k+1}^{-1}\left[\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right]\end{aligned}$$

Derivation of the linear Kalman Filter

We now define the following matrices:

$$P_{k+1,k+1} = \left(P_{k+1,k}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)^{-1} \Longrightarrow \text{estimate covariance matrix}$$

$$K_{k+1} = P_{k+1,k+1} H_{k+1}^T R_{k+1}^{-1} \Longrightarrow \textbf{Kalman} \text{ gain matrix}$$

Substitution of $P_{k+1,k+1}$ and K_{k+1} in

$$\left(P_{k+1}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)\underline{\hat{x}}_{k+1,k+1} = \left(P_{k+1}^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}\right)\underline{\hat{x}}_{k+1,k} + $$
$$+ H_{k+1}^T R_{k+1}^{-1}\left[\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right]$$

Results in:

$$P_{k+1,k+1}^{-1}\underline{\hat{x}}_{k+1,k+1} = P_{k+1,k+1}^{-1}\underline{\hat{x}}_{k+1,k} + P_{k+1,k+1}^{-1} K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right)$$

Derivation of the linear Kalman Filter

Now multiply

$$P^{-1}_{k+1,k+1}\underline{\hat{x}}_{k+1,k+1} = P^{-1}_{k+1,k+1}\underline{\hat{x}}_{k+1,k} + P^{-1}_{k+1,k+1}K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right)$$

with $P_{k+1,k+1}$, which results in:

$$\boxed{\underline{\hat{x}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right)}$$

In words: the optimal state estimate $\underline{\hat{x}}_{k+1,k+1}$ is a combination of the biased state estimate $\underline{\hat{x}}_{k+1,k}$ and the estimated measurement error $\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}$ which is scaled by the **Kalman** gain K_{k+1}.

The Linear Kalman Filter

$k = 1, 2, \ldots N$

1. One-step ahead prediction $\quad \hat{\underline{x}}_{k+1,k} = \Phi_{k+1,k}\hat{\underline{x}}_{k,k} + \Psi_{k+1,k}\underline{u}_k; \quad \hat{\underline{x}}_{0,0} = \hat{\underline{x}}_0$
2. Covariance matrix of state prediction error $\quad P_{k+1,k} = \Phi_{k+1,k}P_{k,k}\Phi^T_{k+1,k} + \Gamma_{k+1,k}Q_{d,k}\Gamma^T_{k+1,k}; \quad P_{0,0} = P_0$
3. Kalman gain calculation
4. Measurement update $\quad \hat{\underline{x}}_{k+1,k+1} = \hat{\underline{x}}_{k+1,k} + K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\hat{\underline{x}}_{k+1,k}\right)$
5. Covariance matrix of state estimation error

Derivation of the linear Kalman Filter

Since $\underline{\hat{x}}_{k+1,k+1}$ is only the best **estimate** of the real state vector, the estimation error has to be analyzed.

Therefore, we define the error vectors as follows:

$\underline{\hat{\varepsilon}}_{k+1,k} = \underline{\hat{x}}_{k+1,k} - \underline{x}_{k+1}$ ⇒ state prediction error

$\underline{\hat{\varepsilon}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k+1} - \underline{x}_{k+1}$ ⇒ optimal state estimation error

Substitution of $\boxed{\underline{\hat{x}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right)}$ in the equation

for $\underline{\hat{\varepsilon}}_{k+1,k+1}$ leads to:

$$\underline{\hat{\varepsilon}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right) - \underline{x}_{k+1}$$

Derivation of the linear Kalman Filter

Now substitute $\underline{z}_{k+1} = H_{k+1}\underline{x}_{k+1} + \underline{v}_{k+1}$

in the optimal state estimator error:

$$\underline{\hat{\varepsilon}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right) - \underline{x}_{k+1}$$

which leads to:

$$\underline{\hat{\varepsilon}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1}\left(H_{k+1}\underline{x}_{k+1} + \underline{v}_{k+1} - H_{k+1}\underline{\hat{x}}_{k+1,k}\right) - \underline{x}_{k+1}$$

$$= \left(\underline{\hat{x}}_{k+1,k} - \underline{x}_{k+1}\right) - K_{k+1}H_{k+1}\left(\underline{\hat{x}}_{k+1,k} - \underline{x}_{k+1}\right) + K_{k+1}\underline{v}_{k+1}$$

$$= \left(I - K_{k+1}H_{k+1}\right)\underline{\hat{\varepsilon}}_{k+1,k} + K_{k+1}\underline{v}_{k+1}$$

$$\boxed{\underline{\hat{\varepsilon}}_{k+1,k} = \underline{\hat{x}}_{k+1,k} - \underline{x}_{k+1}}$$

Derivation of the linear Kalman Filter

The optimal state estimator error is:

$$\underline{\hat{\varepsilon}}_{k+1,k+1} = \left(I - K_{k+1}H_{k+1}\right)\underline{\hat{\varepsilon}}_{k+1,k} + K_{k+1}\underline{v}_{k+1}$$

The quadratic form $\underline{\hat{\varepsilon}}_{k+1,k+1}\underline{\hat{\varepsilon}}^T_{k+1,k+1}$ is given by:

$$\begin{aligned}
\underline{\hat{\varepsilon}}_{k+1,k+1}\underline{\hat{\varepsilon}}^T_{k+1,k+1} &= \left[\left(I - K_{k+1}H_{k+1}\right)\underline{\hat{\varepsilon}}_{k+1,k} + K_{k+1}\underline{v}_{k+1}\right]\left[\left(I - K_{k+1}H_{k+1}\right)\underline{\hat{\varepsilon}}_{k+1,k} + K_{k+1}\underline{v}_{k+1}\right]^T \\
&= \left(I - K_{k+1}H_{k+1}\right)\underline{\hat{\varepsilon}}_{k+1,k}\underline{\hat{\varepsilon}}^T_{k+1,k}\left(I - K_{k+1}H_{k+1}\right)^T + \\
&\quad + \left(I - K_{k+1}H_{k+1}\right)\underline{\hat{\varepsilon}}_{k+1,k}\underline{v}^T_{k+1}K^T_{k+1} + \\
&\quad + K_{k+1}\underline{v}_{k+1}\underline{\hat{\varepsilon}}^T_{k+1,k}\left(I - K_{k+1}H_{k+1}\right)^T + K_{k+1}\underline{v}_{k+1}\underline{v}^T_{k+1}K^T_{k+1}
\end{aligned}$$

Derivation of the linear Kalman Filter

The quadratic form $\hat{\underline{\varepsilon}}_{k+1,k+1}\hat{\underline{\varepsilon}}^T_{k+1,k+1}$ is given by:

$$\begin{aligned}\hat{\underline{\varepsilon}}_{k+1,k+1}\hat{\underline{\varepsilon}}^T_{k+1,k+1} = &\left(I - K_{k+1}H_{k+1}\right)\hat{\underline{\varepsilon}}_{k+1,k}\hat{\underline{\varepsilon}}^T_{k+1,k}\left(I - K_{k+1}H_{k+1}\right)^T + \\ &+\left(I - K_{k+1}H_{k+1}\right)\hat{\underline{\varepsilon}}_{k+1,k}\underline{v}^T_{k+1}K^T_{k+1} + \\ &+K_{k+1}\underline{v}_{k+1}\hat{\underline{\varepsilon}}^T_{k+1,k}\left(I - K_{k+1}H_{k+1}\right)^T + K_{k+1}\underline{v}_{k+1}\underline{v}^T_{k+1}K^T_{k+1}\end{aligned}$$

The expectation of $\hat{\underline{\varepsilon}}_{k+1,k+1}\hat{\underline{\varepsilon}}^T_{k+1,k+1}$ is the covariance matrix given by:

$$\begin{aligned}E\left\{\hat{\underline{\varepsilon}}_{k+1,k+1}\hat{\underline{\varepsilon}}^T_{k+1,k+1}\right\} = &\left(I - K_{k+1}H_{k+1}\right)E\left\{\hat{\underline{\varepsilon}}_{k+1,k}\hat{\underline{\varepsilon}}^T_{k+1,k}\right\}\left(I - K_{k+1}H_{k+1}\right)^T + \\ &+\left(I - K_{k+1}H_{k+1}\right)E\left\{\hat{\underline{\varepsilon}}_{k+1,k}\underline{v}^T_{k+1}\right\}K^T_{k+1} + \\ &+K_{k+1}E\left\{\underline{v}_{k+1}\hat{\underline{\varepsilon}}^T_{k+1,k}\right\}\left(I - K_{k+1}H_{k+1}\right)^T + K_{k+1}E\left\{\underline{v}_{k+1}\underline{v}^T_{k+1}\right\}K^T_{k+1}\end{aligned}$$

Derivation of the linear Kalman Filter

If we consider the noise statistics given earlier as:

$$E\left\{\hat{\underline{\varepsilon}}_{k+1,k}\hat{\underline{\varepsilon}}^{T}_{k+1,k}\right\} = E\left\{\left(\underline{x}_{k+1}-\hat{\underline{x}}_{k+1,k}\right)\left(\underline{x}_{k+1}-\hat{\underline{x}}_{k+1,k}\right)^{T}\right\} = P_{k+1,k}$$

$$E\left\{\underline{v}_{k+1}\underline{v}^{T}_{k+1}\right\} = R_{k+1}$$

$$E\left\{\hat{\underline{\varepsilon}}_{k+1,k}\underline{v}^{T}_{k+1}\right\} = E\left\{(\hat{\underline{x}}_{k+1,k}-\underline{x}_{k+1})\underline{v}^{T}_{k+1}\right\} = E\left\{\underline{w}_{d,k}\underline{v}^{T}_{k+1}\right\} = 0$$

And substitute this in in the expression for $\hat{\underline{\varepsilon}}_{k+1,k+1}\hat{\underline{\varepsilon}}^{T}_{k+1,k+1}$ we get:

$$\begin{aligned}E\left\{\hat{\underline{\varepsilon}}_{k+1,k+1}\hat{\underline{\varepsilon}}^{T}_{k+1,k+1}\right\} &= \left(I-K_{k+1}H_{k+1}\right)E\left\{\hat{\underline{\varepsilon}}_{k+1,k}\hat{\underline{\varepsilon}}^{T}_{k+1,k}\right\}\left(I-K_{k+1}H_{k+1}\right)^{T} + \\ &+\cancel{\left(I-K_{k+1}H_{k+1}\right)E\left\{\hat{\underline{\varepsilon}}_{k+1,k}\underline{v}^{T}_{k+1}\right\}K^{T}_{k+1}} + \\ &+\cancel{K_{k+1}E\left\{\underline{v}_{k+1}\hat{\underline{\varepsilon}}^{T}_{k+1,k}\right\}\left(I-K_{k+1}H_{k+1}\right)^{T}} + K_{k+1}E\left\{\underline{v}_{k+1}\underline{v}^{T}_{k+1}\right\}K^{T}_{k+1}\end{aligned}$$

Derivation of the linear Kalman Filter

If we consider the noise statistics given earlier as:

$$E\left\{\underline{\hat{\varepsilon}}_{k+1,k+1}\underline{\hat{\varepsilon}}^T_{k+1,k+1}\right\} = \left(I - K_{k+1}H_{k+1}\right)E\left\{\underline{\hat{\varepsilon}}_{k+1,k}\underline{\hat{\varepsilon}}^T_{k+1,k}\right\}\left(I - K_{k+1}H_{k+1}\right)^T +$$
$$+0+0+K_{k+1}E\left\{\underline{v}_{k+1}\underline{v}^T_{k+1}\right\}K^T_{k+1}$$

which reduces to:

$$\boxed{E\left\{\underline{\hat{\varepsilon}}_{k+1,k+1}\underline{\hat{\varepsilon}}^T_{k+1,k+1}\right\} = \left(I - K_{k+1}H_{k+1}\right)P_{k+1,k}\left(I - K_{k+1}H_{k+1}\right)^T + K_{k+1}R_{k+1}K^T_{k+1}}$$

In words, this is the optimal error covariance matrix estimate! We will now proceed to further simplify this expression…

Earlier, we defined for the optimal error covariance:

$$P_{k+1,k+1} = \left(P^{-1}_{k+1,k} + H^T_{k+1}R^{-1}_{k+1}H_{k+1}\right)^{-1}$$

Derivation of the linear Kalman Filter

We will now proceed to simplify the earlier derived expression for the optimal state-estimate covariance matrix:

$$P_{k+1,k+1} = \left(P_{k+1,k}^{-1} + H_{k+1}^{T} R_{k+1}^{-1} H_{k+1}\right)^{-1}$$

If we invert this relationship we get:

$$P_{k+1,k+1}^{-1} = \left(P_{k+1,k}^{-1} + H_{k+1}^{T} R_{k+1}^{-1} H_{k+1}\right)$$

Now left-multiply with $P_{k+1,k+1}$ and right-multiply with $P_{k+1,k}$:

Trick!

$$P_{k+1,k+1} P_{k+1,k+1}^{-1} P_{k+1,k} = P_{k+1,k} = \left(P_{k+1,k+1} P_{k+1,k}^{-1} P_{k+1,k} + P_{k+1,k+1} H_{k+1}^{T} R_{k+1}^{-1} H_{k+1} P_{k+1,k}\right)$$

Which reduces to:

$$P_{k+1,k} = P_{k+1,k+1} + P_{k+1,k+1} H_{k+1}^{T} R_{k+1}^{-1} H_{k+1} P_{k+1,k}$$

Derivation of the linear Kalman Filter

We can further simplify

$$P_{k+1,k} = P_{k+1,k+1} + P_{k+1,k+1} H_{k+1}^{T} R_{k+1}^{-1} H_{k+1} P_{k+1,k}$$

into

$$P_{k+1,k+1} = \left(I - P_{k+1,k+1} H_{k+1}^{T} R_{k+1}^{-1} H_{k+1} \right) P_{k+1,k}$$

Recall the Kalman gain matrix $K_{k+1} = P_{k+1,k+1} H_{k+1}^{T} R_{k+1}^{-1}$

If we substitute this in the covariance matrix estimate $P_{k+1,k+1}$ we get:

$$\boxed{P_{k+1,k+1} = \left(I - K_{k+1} H_{k+1} \right) P_{k+1,k}}$$

In words, what we have done is create an expression for the sample covariance matrix in terms of **only** the **old** covariance matrix!

Derivation of the linear Kalman Filter

Now substitute $\boxed{P_{k+1,k+1} = \left(I - K_{k+1}H_{k+1}\right)P_{k+1,k}}$ into

$$\boxed{E\left\{\underline{\hat{\varepsilon}}_{k+1,k+1}\underline{\hat{\varepsilon}}^T_{k+1,k+1}\right\} = \left(I - K_{k+1}H_{k+1}\right)P_{k+1,k}\left(I - K_{k+1}H_{k+1}\right)^T + K_{k+1}R_{k+1}K^T_{k+1}}$$

which results in:

$$\begin{aligned} E\left\{\underline{\hat{\varepsilon}}_{k+1,k+1}\underline{\hat{\varepsilon}}^T_{k+1,k+1}\right\} &= P_{k+1,k+1}\left(I - K_{k+1}H_{k+1}\right)^T + K_{k+1}R_{k+1}K^T_{k+1} \\ &= P_{k+1,k+1} - P_{k+1,k+1}H^T_{k+1}K^T_{k+1} + K_{k+1}R_{k+1}K^T_{k+1} \end{aligned}$$

which can be rewritten into:

$$E\left\{\underline{\hat{\varepsilon}}_{k+1,k+1}\underline{\hat{\varepsilon}}^T_{k+1,k+1}\right\} = P_{k+1,k+1} + \left(-P_{k+1,k+1}H^T_{k+1} + K_{k+1}R_{k+1}\right)K^T_{k+1}$$

Derivation of the linear Kalman Filter

Now we substitute the Kalman gain matrix $K_{k+1} = P_{k+1,k+1} H_{k+1}^T R_{k+1}^{-1}$ into

$$E\left\{ \underline{\hat{\varepsilon}}_{k+1,k+1} \underline{\hat{\varepsilon}}_{k+1,k+1}^T \right\} = P_{k+1,k+1} + \left(-P_{k+1,k+1} H_{k+1}^T + K_{k+1} R_{k+1} \right) K_{k+1}^T$$

which results in:

$$\begin{aligned} E\left\{ \underline{\hat{\varepsilon}}_{k+1,k+1} \underline{\hat{\varepsilon}}_{k+1,k+1}^T \right\} &= P_{k+1,k+1} + \left[-P_{k+1,k+1} H_{k+1}^T + P_{k+1,k+1} H_{k+1}^T R_{k+1}^{-1} R_{k+1} \right] K_{k+1}^T \\ &= P_{k+1,k+1} \end{aligned}$$

We have now proven that $P_{k+1,k+1}$ is the covariance matrix of the optimal state estimation error $\underline{\hat{\varepsilon}}_{k+1,k+1}$!!!

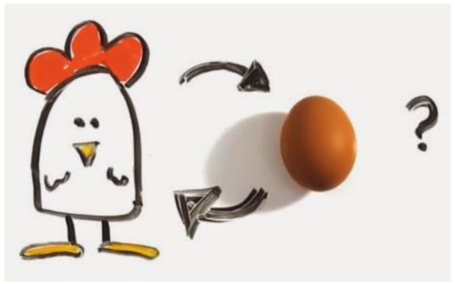
Derivation of the linear Kalman Filter

We had already found an expression for the Kalman gain:

$$K_{k+1} = P_{k+1,k+1} H_{k+1}^{T} R_{k+1}^{-1}$$

We have also found an expression for the optimal state estimation error covariance matrix:

$$\boxed{P_{k+1,k+1} = \left(I - K_{k+1} H_{k+1}\right) P_{k+1,k}}$$



We have a Chicken and Egg problem here!

Covariance matrix $P_{k+1,k+1}$ depends on K_{k+1} and vice versa!

Derivation of the linear Kalman Filter

The chicken-egg problem can be solved by rewriting the earlier un-simplified expression

$$P_{k+1,k+1} = \left(P_{k+1,k}^{-1} + H_{k+1}^{T} R_{k+1}^{-1} H_{k+1} \right)^{-1}$$

into a new expression for the Kalman gain K_{k+1}

We use the matrix inversion lemma to further simplify this un-simplified expression:

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B\left[DA^{-1}B + C^{-1}\right]^{-1} DA^{-1}$$

Derivation of the linear Kalman Filter

The matrices in $[A+BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1}$ are:

$$A = P_{k+1,k}^{-1}, \qquad B = H_{k+1}^{T}$$
$$C = R_{k+1}^{-1}, \qquad D = H_{k+1}$$

With the matrix inversion lemma, the expression for $P_{k+1,k+1}$ can be simplified as follows:

$$\begin{aligned}
P_{k+1,k+1} &= \left(P_{k+1,k}^{-1} + H_{k+1}^{T}R_{k+1}^{-1}H_{k+1}\right)^{-1} \\
&= P_{k+1,k} - P_{k+1,k}H_{k+1}^{T}\left(H_{k+1}P_{k+1,k}H_{k+1}^{T} + R_{k+1}\right)^{-1}H_{k+1}P_{k+1,k} \\
&= \left[I - P_{k+1,k}H_{k+1}^{T}\left(H_{k+1}P_{k+1,k}H_{k+1}^{T} + R_{k+1}\right)^{-1}H_{k+1}\right]P_{k+1,k}
\end{aligned}$$

completed

Derivation of the linear Kalman Filter

Now compare

$$P_{k+1,k+1} = \left[I - P_{k+1,k} H_{k+1}^T \left(H_{k+1} P_{k+1,k} H_{k+1}^T + R_{k+1} \right)^{-1} H_{k+1} \right] P_{k+1,k}$$

with

$$\boxed{P_{k+1,k+1} = \left(I - K_{k+1} H_{k+1} \right) P_{k+1,k}}$$

We have found a new, **recursive** expression for the **Kalman gain**!

$$\boxed{K_{k+1} = P_{k+1,k} H_{k+1}^T \left(H_{k+1} P_{k+1,k} H_{k+1}^T + R_{k+1} \right)^{-1}}$$

Chicken and Egg problem solved!
Which means we have completed our derivation of the Linear Kalman Filter!

The Linear Kalman Filter

$k = 1,2,\dots N$

1. One-step ahead prediction $\quad \underline{\hat{x}}_{k+1,k} = \Phi_{k+1,k}\,\underline{\hat{x}}_{k,k} + \Psi_{k+1,k}\,\underline{u}_k\,; \qquad \underline{\hat{x}}_{0,0} = \underline{\hat{x}}_0$
2. Covariance matrix of state prediction error $\quad P_{k+1,k} = \Phi_{k+1,k} P_{k,k} \Phi^T_{k+1,k} + \Gamma_{k+1,k} Q_{d,k} \Gamma^T_{k+1,k}\,; \qquad P_{0,0} = P_0$
3. Kalman gain calculation $\quad K_{k+1} = P_{k+1,k} H^T_{k+1} \left(H_{k+1} P_{k+1,k} H^T_{k+1} + R_{k+1} \right)^{-1}$
4. Measurement update $\quad \underline{\hat{x}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1} \left(\underline{z}_{k+1} - H_{k+1} \underline{\hat{x}}_{k+1,k} \right)$
5. Covariance matrix of state estimation error $\quad P_{k+1,k+1} = \left(I - K_{k+1} H_{k+1} \right) P_{k+1,k}$

The Linear Kalman Filter

Things to consider during linear Kalman filter implementation:

- **System needs to be linear.**
- A necessary condition for a convergent Kalman filter is the guarantee of a fully observable system.
- System model (F, G, H, D matrices) needs to be known.
- Noise statistical information (Q and R matrices) needs to be known.
- The R matrix should be selected based on the noise level of sensor noises in case of a tightly coupled system.
- For a loosely coupled system the observation noise can be colored. In this case additional states will be augmented to the system. These states will be estimated together with the original states for the observation noise.
- The Q matrix should represent both input sensor noises and model uncertainties.
- Initial condition for $P(0|0)$ **should** be large for unknown initial state error.
- Initial condition of $x(0|0)$ in principle can be chosen arbitrarily.

The Linear Kalman Filter

Defining the noise covariance matrices

Until now, we have not clearly shown how to set the noise covariance matrices Q and R. What we can do is use our knowledge from **calibrating the sensors** to get estimates for Q and R.

If our input vector is defined as $\underline{u} = [u_1 \quad \cdots \quad u_n]^T$ then each input has an associated noise standard deviation $\sigma_{u1}, \ \dots, \sigma_{un}$.

Equivalently, if our measurement vector is defined as $\underline{z} = [z_1 \quad \cdots \quad z_n]^T$ then each measurement has an associated standard deviation $\sigma_{z1}, \ \dots, \sigma_{zn}$.

The noise covariance matrices Q and R can then be defined as follows:

$$Q = \begin{bmatrix} \sigma_{u1}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{un}^2 \end{bmatrix}, \qquad R = \begin{bmatrix} \sigma_{z1}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{\Delta zn}^2 \end{bmatrix}$$

The Linear Kalman Filter

Example 3.2: Linear Kalman filter

Consider the linear system:

$$\dot{x}(t) = -x(t) + w(t); \qquad x(0) = 1$$

$$z_n(t) = 3x(t)$$

$$z(t_k) = z_n(t_k) + v(t_k); \qquad k = 1, 2, \ldots$$

with noise statistics:

$$E\{w(t)\} = 0, \qquad E\{w(t)w(\tau)\} = Q, \qquad Q = 10000$$

$$E\{v(t_k)\} = 0; \qquad E\{v(t_k)v(t_l)\} = R, \qquad R = 4$$

and initial estimates for the state and state estimate covariance 'matrix':

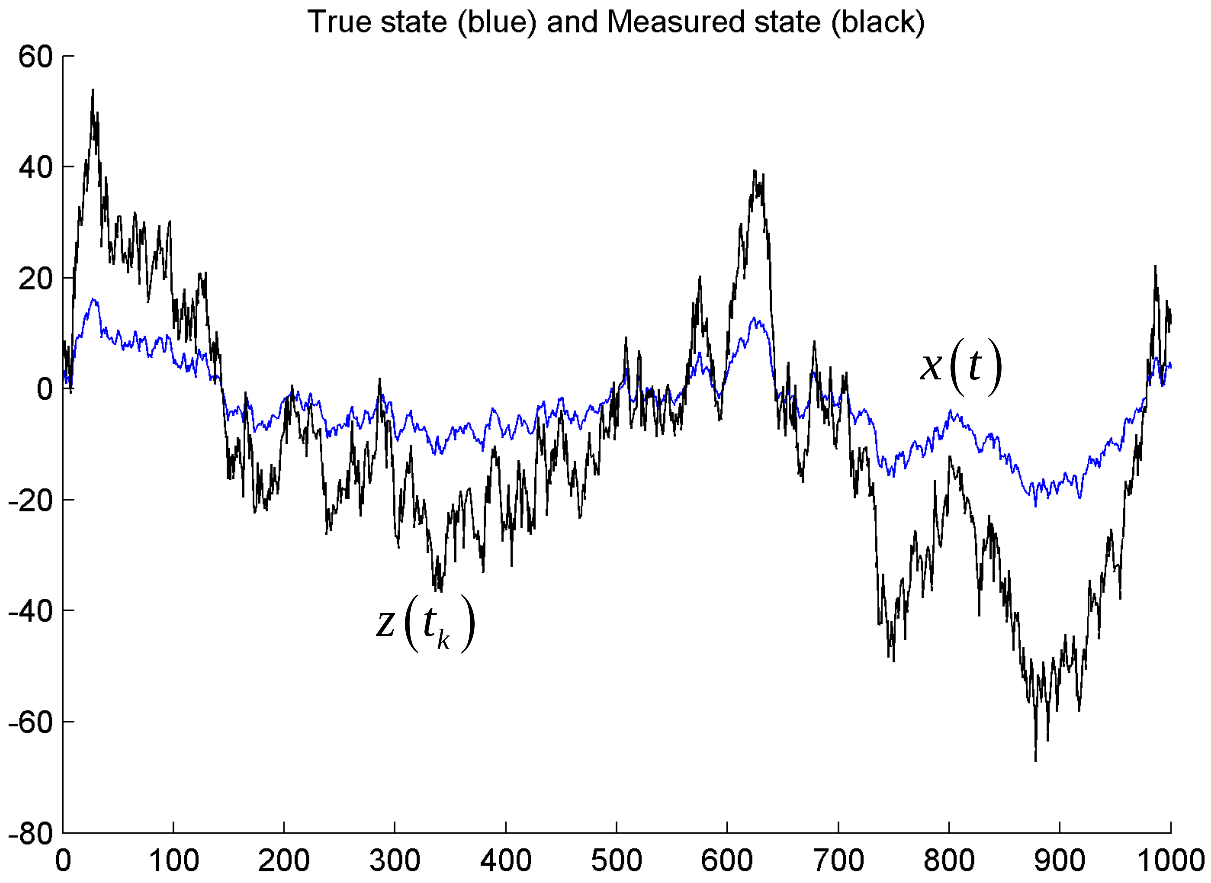
$$\hat{x}_0 = \hat{x}(0|0) = 10, \qquad P(0|0) = 100;$$

and simulation parameters:

$$\Delta T = 0.01, \qquad N = 1000$$

The Linear Kalman Filter

Example 3.2: Linear Kalman filter



The Linear Kalman Filter

Example 3.2: Linear Kalman filter

1. One-step ahead prediction $\quad \underline{\hat{x}}_{k+1,k} = \Phi_{k+1,k}\underline{\hat{x}}_{k,k}; \quad \underline{\hat{x}}_{0,0} = \underline{\hat{x}}_0$

using Matlab command `c2d(-1,1,0.01)` we find:

$$\Phi_{k+1,k} = 0.99, \qquad \Psi_{k+1,k} = 0, \qquad \Gamma_{k+1,k} = 0.01$$

$$\begin{aligned}
&\dot{x}(t) = -x(t) + w(t); \quad x(0) = 1,\\
&z_n(t) = 3x(t),\\
&z(t_k) = z_n(t_k) + v(t_k),\\
&E\{w(t)\} = 0, \qquad Q = 10000\\
&E\{v(t_k)\} = 0; \qquad R = 4\\
&\hat{x}_0 = \hat{x}(0|0) = 10, \qquad P(0|0) = 100;\\
&\Delta T = 0.01
\end{aligned}$$

The one-step ahead prediction then is:

$$\begin{aligned}
\underline{\hat{x}}_{k+1,k} &= 0.99 * 10.0\\
&= 9.9
\end{aligned}$$

The Linear Kalman Filter

Example 3.2: Linear Kalman filter

2. Covariance matrix of state prediction error

$$P_{k+1,k} = \Phi_{k+1,k} P_{k,k} \Phi^T_{k+1,k} + \Gamma_{k+1,k} Q_{d,k} \Gamma^T_{k+1,k}$$

We can now calculate the covariance matrix of the state prediction error:

$$\begin{aligned} \boldsymbol{P}_{\boldsymbol{k+1,k}} &= 0.99 * 100 * 0.99 + 0.01 * 10000 * 0.01 \\ &= 99.01 \end{aligned}$$

$$\dot{x}(t) = -x(t) + w(t); \quad x(0) = 1,$$
$$z_n(t) = 3x(t),$$
$$z(t_k) = z_n(t_k) + v(t_k),$$
$$\Phi_{k+1,k} = 0.99, \Psi_{k+1,k} = 0, \Gamma_{k+1,k} = 0.01$$
$$Q = 10000, \qquad R = 4$$
$$\hat{x}_0 = \hat{x}(0|0) = 10, \qquad P(0|0) = 100;$$
$$\hat{\underline{x}}_{k+1,k} = 9.9$$

The Linear Kalman Filter

Example 3.2: Linear Kalman filter

3. Kalman gain calculation $K_{k+1} = P_{k+1,k} H_{k+1}^T \left(H_{k+1} P_{k+1,k} H_{k+1}^T + R_{k+1} \right)^{-1}$

Using the estimated covariance matrix, we can now calculate the Kalman gain:

$$\begin{aligned} K_{k+1} &= 99.01 * 3 * \left(3 * 99.01 * 3 + 4 \right)^{-1} \\ &= 0.3318 \end{aligned}$$

$$\begin{array}{ll} \dot{x}(t) = -x(t) + w(t); & x(0) = 1, \\ z_n(t) = 3x(t), & \\ z(t_k) = z_n(t_k) + v(t_k)\ , & \\ \Phi_{k+1,k} = 0.99, \Psi_{k+1,k} = 0, \Gamma_{k+1,k} = 0.01 & \\ Q = 10000, & R = 4 \\ \hat{x}_0 = \hat{x}(0 \mid 0) = 10, & P(0 \mid 0) = 100; \\ \hat{\underline{x}}_{k+1,k} = 9.9, & P_{k+1,k} = 99.01 \end{array}$$

The Linear Kalman Filter

Example 3.2: Linear Kalman filter

4. Measurement update $\hat{\underline{x}}_{k+1,k+1} = \hat{\underline{x}}_{k+1,k} + K_{k+1}\left(\underline{z}_{k+1} - H_{k+1}\hat{\underline{x}}_{k+1,k}\right)$

Before being able to calculate the measurement update, we need a value for the measurement $\underline{z}_{k+1}$:

$$w(t_1) = 53.8, \qquad v(t_1) = 1.35, \qquad \Delta T = 0.01$$

$$x(t_1) = x(0) + (-x(0) + w(t_1))\Delta T = 1.528$$

$$\underline{z}_{k+1} = 3x(t_1) + v(t_k) = 3*1.528 + 1.35 = \ 5.93$$

Now update the state estimate with the weighted measurement data:

$$\hat{\underline{x}}_{k+1,k+1} = 9.9 + 0.3318*\left(5.93 - 3*9.9\right)$$
$$= 2.01$$

$$\dot{x}(t) = -x(t) + w(t); \quad x(0) = 1,$$
$$z_n(t) = 3x(t),$$
$$z(t_k) = z_n(t_k) + v(t_k),$$
$$\Phi_{k+1,k} = 0.99, \Psi_{k+1,k} = 0, \Gamma_{k+1,k} = 0.01$$
$$Q = 10000, \qquad R = 4$$
$$\hat{\underline{x}}_{k+1,k} = 9.9, \qquad P_{k+1,k} = 99.01$$
$$K_{k+1} = 0.3318$$

The Linear Kalman Filter

Example 3.2: Linear Kalman filter

5. Covariance matrix of state estimation error $$P_{k+1,k+1} = \left(I - K_{k+1}H_{k+1}\right)P_{k+1,k}$$

Now update the state estimation covariance matrix:

$$\begin{aligned} P_{k+1,k+1} &= \left(1 - 0.3318 * 3\right) * 99.01 \\ &= 0.4554 \end{aligned}$$

We are now ready for the next time step!

$$\dot{x}(t) = -x(t) + w(t); \quad x(0) = 1,$$
$$z_n(t) = 3x(t),$$
$$z(t_k) = z_n(t_k) + v(t_k),$$
$$\Phi_{k+1,k} = 0.99, \Psi_{k+1,k} = 0, \Gamma_{k+1,k} = 0.01$$
$$Q = 10000, \qquad R = 4$$
$$\underline{\hat{x}}_{k+1,k} = 9.9, \qquad P_{k+1,k} = 99.01$$
$$K_{k+1} = 0.3318, \qquad \underline{\hat{x}}_{k+1,k+1} = 2.01$$

The Linear Kalman Filter

Example 3.3: Linear Kalman filter Step 2

Calculate state estimate for k=2 **for yourself** if it is given that:

$$w(t_2) = 20, \qquad v(t_2) = -1.5, \qquad \Delta T = 0.01$$

$$x(t_2) = x(t_1) + (-x(t_1) + w(t_2))\Delta T = 1.528 + (-1.528 + 20) * 0.01 = 1.71$$

$$\underline{z}_{k+1} = 3x(t_2) + v(t_2) = 3 * 1.71 - 1.5 = \ 3.63$$

Answers:

$$\hat{\underline{x}}_{k+1,k} =$$

$$P_{k+1,k} =$$

$$K_{k+1} =$$

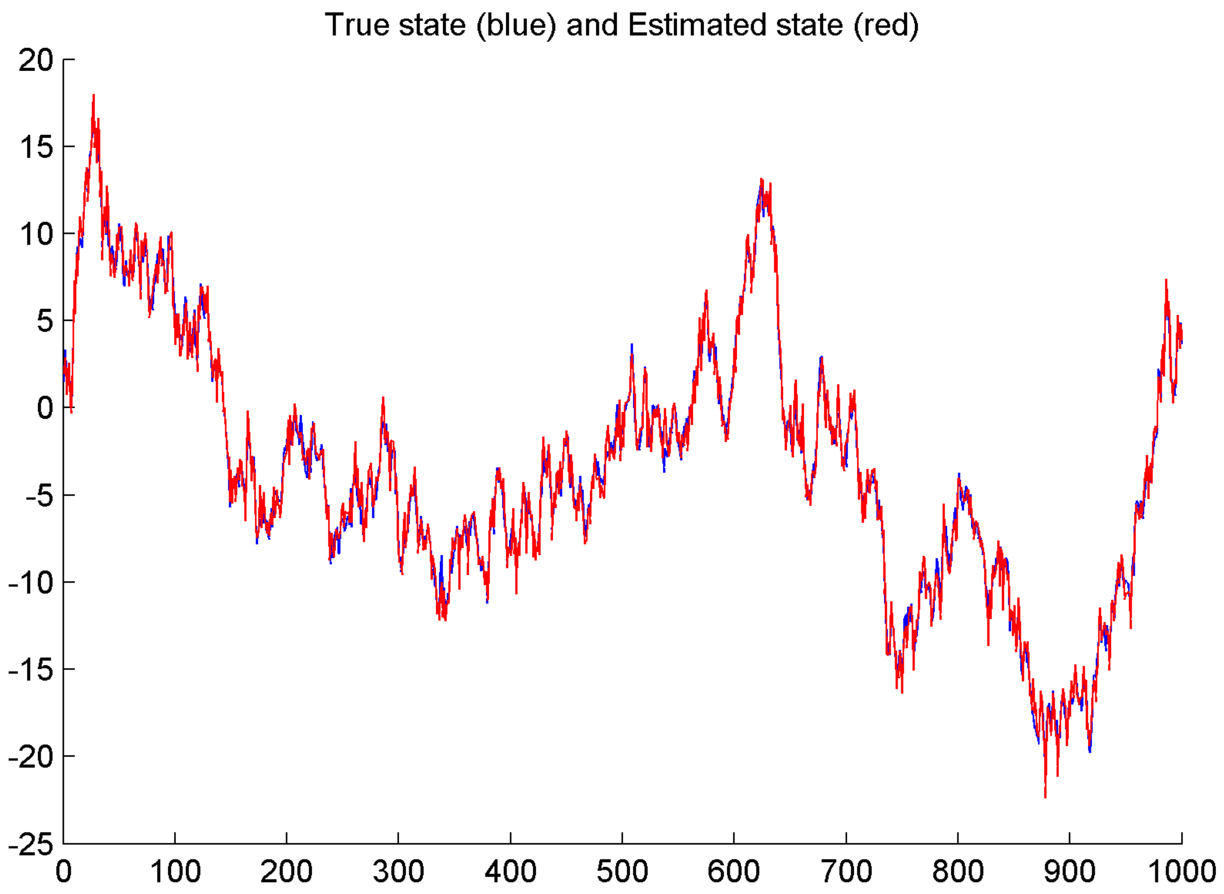
$$\hat{\underline{x}}_{k+1,k+1} =$$

$$P_{k+1,k+1} =$$

$$\dot{x}(t) = -x(t) + w(t); \quad x(1) = 1.528,$$
$$z_n(t) = 3x(t),$$
$$z(t_k) = z_n(t_k) + v(t_k),$$
$$\Phi_{k+1,k} = 0.99, \Psi_{k+1,k} = 0, \Gamma_{k+1,k} = 0.01$$
$$Q = 10000, \qquad R = 4$$
$$\hat{\underline{x}}_{k,k} = 2.01,$$
$$P_{k,k} = 0.4554$$
$$K_{k+1} = 0.3318$$
$$\boxed{k = 2}$$

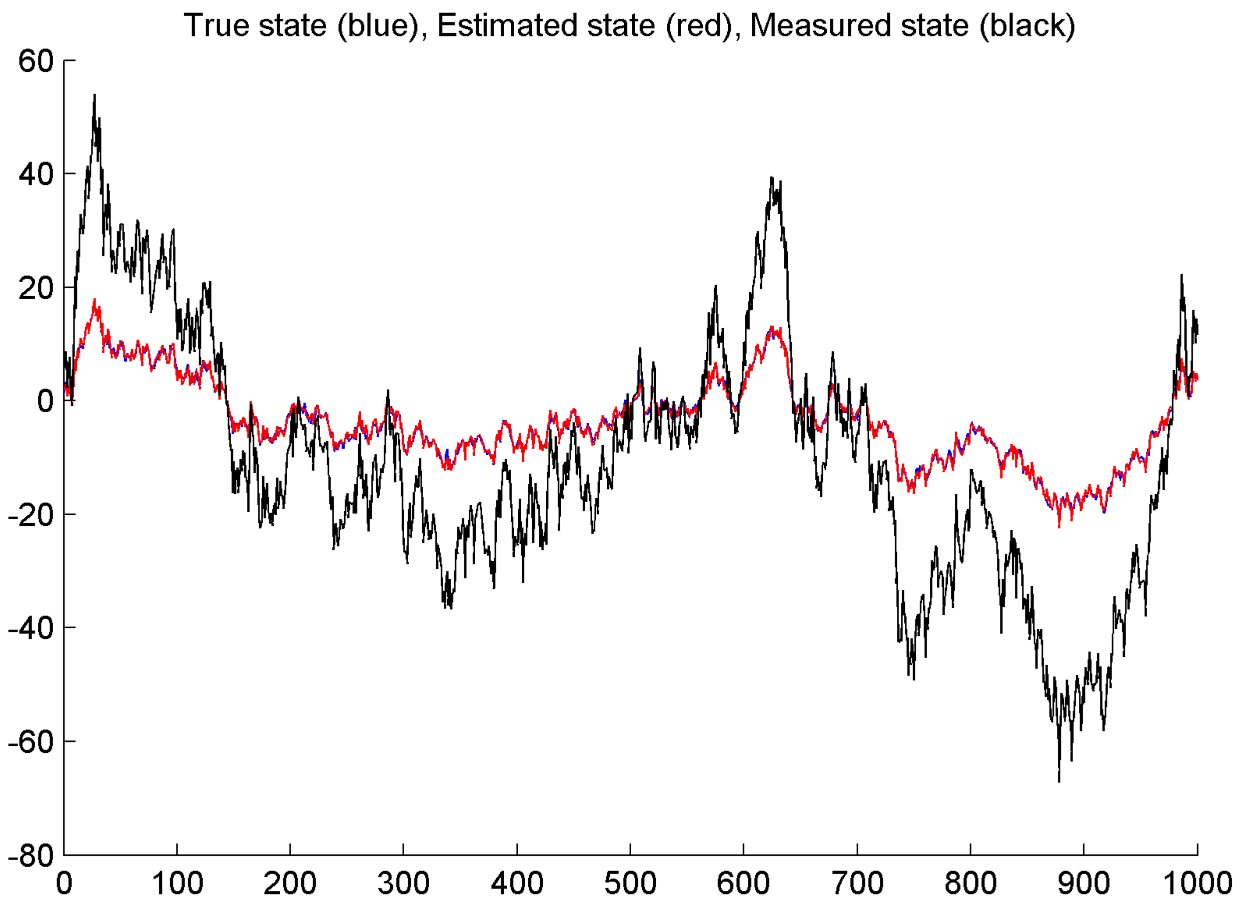
The Linear Kalman Filter

Example 3.3: Linear Kalman filter



The Linear Kalman Filter

Example 3.3: Linear Kalman filter



The Linear Kalman Filter

Example 3.4: The linear Kalman filter

Matlab Demo

Derivation of The Extended Kalman Filter

The Kalman filter has been designed to estimate the state vector of linear systems. In practice, however, the system and measurement equations turn out to be nonlinear most of the time.
If the model turns out to be nonlinear, the **Extended Kalman Filter** (EKF), which is an extension of the Kalman filter to nonlinear systems, can be used to estimate the state vector.

The general nonlinear state space model is:

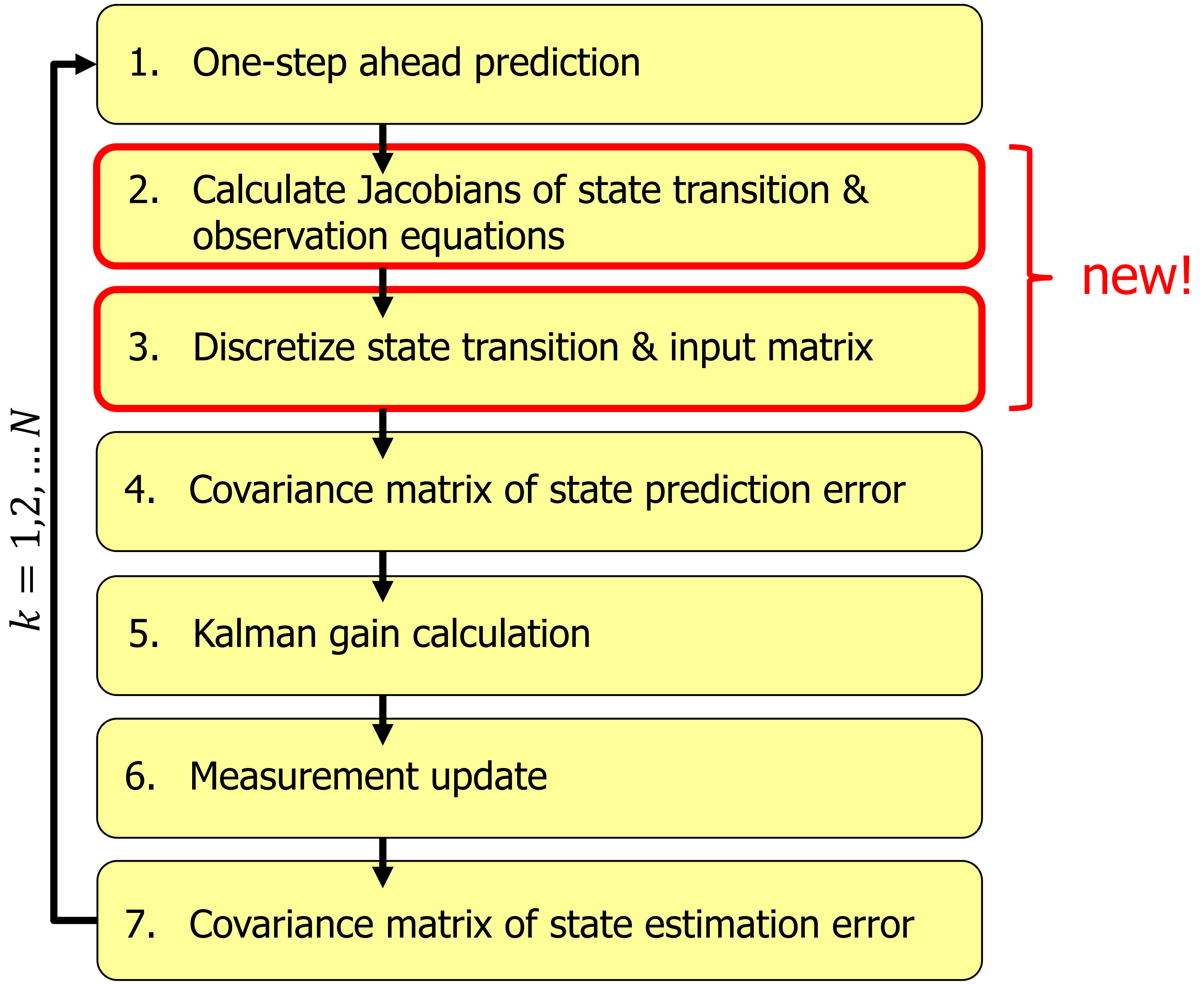
$$\dot{\underline{x}}(t) = f\left(\underline{x}(t), \underline{u}(t), t\right) + G\left(\underline{x}(t), t\right)\underline{w}(t); \qquad \underline{x}(0) = \underline{x}_0$$

$$\underline{z}_n(t) = h\left(\underline{x}(t), \underline{u}(t), t\right)$$

$$\underline{z}(t_k) = \underline{z}_n(t_k) + \underline{v}(t_k)\ ; \qquad\qquad k = 1, 2, \ldots$$

It will be assumed that f and h are continuous and differentiable with respect to all elements of $\underline{x}$ and $\underline{u}$.

The Extended Kalman Filter


1. One-step ahead prediction
2. Calculate Jacobians of state transition & observation equations
3. Discretize state transition & input matrix
4. Covariance matrix of state prediction error
5. Kalman gain calculation
6. Measurement update
7. Covariance matrix of state estimation error
new!
$k = 1,2,\ldots N$

Derivation of The Extended Kalman Filter

Linear Perturbation Equations

The nonlinear state and observation equations $f(\underline{x}(t), \underline{u}(t), t)$ and $h(\underline{x}(t), \underline{u}(t), t)$ will be **linearized** about their <u>nominal values</u> of $\underline{x}^*(t)$, and $\underline{u}^*(t)$ respectively.

The nominal nonlinear state and observation equations are:

$$\dot{\underline{x}}^*(t) = \underline{f}\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$$

$$\underline{z}_n^*(t) = h\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$$

The perturbations in $\underline{x}(t)$, $\underline{u}(t)$, and $\underline{z}_n$ are:

$$\delta\underline{x}(t) = \underline{x}(t) - \underline{x}^*(t), \qquad \delta\underline{z}_n(t) = \underline{z}_n(t) - \underline{z}_n^*(t),$$

$$\delta\underline{u}(t) = \underline{u}(t) - \underline{u}^*(t) = 0$$

since the input is deterministic, $\underline{\delta u}(t) = 0$.

Derivation of The Extended Kalman Filter

Linear Perturbation Equations: State equation

The linear perturbation of the state equation is:

$$\frac{d}{dt}\delta \underline{x}(t) = \delta \dot{\underline{x}}(t)$$

$$= \dot{\underline{x}}(t) - \dot{\underline{x}}^*(t)$$

$$= f\left(\underline{x}(t), \underline{u}(t), t\right) + G\left(\underline{x}(t), t\right)\underline{w}(t) - f\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$$

Consider the Taylor expansion of $f(\underline{x}(t), \underline{u}(t), t)$ at $\underline{x}^*(t)$, and $\underline{u}^*(t)$:

$$f\left(\underline{x}(t), \underline{u}(t), t\right) = f\left(\underline{x}^*(t), \underline{u}^*(t), t\right) + F_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)\delta \underline{x}(t) + R^1$$

with $F_x(\underline{x}^*(t), \underline{u}^*(t), t)$ the $n \times n$ Jacobian matrix given by:

$$\boxed{F_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right)\Big|_{\underline{x}(t)=\underline{x}^*(t), \underline{u}(t)=\underline{u}^*(t)}}$$

Derivation of The Extended Kalman Filter

Linear Perturbation Equations: State equation

The linear perturbation of the state equation is:

$$\delta\dot{\underline{x}}(t) = f\left(\underline{x}(t),\underline{u}(t),t\right) - f\left(\underline{x}^*(t),\underline{u}^*(t),t\right) + G\left(\underline{x}(t),t\right)\underline{w}(t)$$

Using the Jacobian F and neglecting the remainder term R^1 we can make the following substitution:

$$f\left(\underline{x}(t),\underline{u}(t),t\right) - f\left(\underline{x}^*(t),\underline{u}^*(t),t\right) = F_x\left(\underline{x}^*(t),\underline{u}^*(t),t\right)\delta\underline{x}(t)$$

The final state perturbation equation then becomes:

$$\boxed{\delta\dot{\underline{x}}(t) = F_x\left(\underline{x}^*(t),\underline{u}^*(t),t\right)\delta\underline{x}(t) + G\left(\underline{x}^*(t),t\right)\underline{w}(t)}$$

Where we have made the following assumption:

$$G\left(\underline{x}(t),t\right) = G\left(\underline{x}^*(t),t\right)$$

Derivation of The Extended Kalman Filter

Linear Perturbation Equations: Measurement equation

Consider the measurement perturbation:

$$\delta \underline{z}_n(t) = \underline{z}_n(t) - \underline{z}_n^*(t)$$

which is equivalent to:

$$\delta \underline{z}_n(t) = h\left(\underline{x}(t), \underline{u}(t), t\right) - h\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$$

The Taylor expansion of $h(\underline{x}(t), \underline{u}(t), t)$ at $\underline{x}^*(t)$, and $\underline{u}^*(t)$ is:

$$h\left(\underline{x}(t), \underline{u}(t), t\right) = h\left(\underline{x}^*(t), \underline{u}^*(t), t\right) + H_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)\delta \underline{x}(t) + R^1$$

with $H_x(\underline{x}^*(t), \underline{u}^*(t), t)$ the $m \times n$ Jacobian matrix given by:

$$\boxed{H_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)\Big|_{\underline{x}(t)=\underline{x}^*(t), \underline{u}(t)=\underline{u}^*(t)}}$$

Derivation of The Extended Kalman Filter

Linear Perturbation Equations: Measurement equation

The linear perturbation measurement equation is:

$$\delta \underline{z}_n(t) = h\left(\underline{x}(t), \underline{u}(t), t\right) - h\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$$

Using the Jacobian H_x we can make the following substitution:

$$h\left(\underline{x}(t), \underline{u}(t), t\right) - h\left(\underline{x}^*(t), \underline{u}^*(t), t\right) = H_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)\delta \underline{x}(t) + R^1$$

If we neglect the remainder term R^1 the final measurement perturbation equations become:

$$\boxed{\begin{aligned} &\delta \underline{z}_n(t) = H_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)\delta \underline{x}(t) \\ &\delta \underline{z}(t_k) = \delta \underline{z}_n(t_k) + \underline{v}(t_k) \end{aligned}}$$

The Extended Kalman Filter

Note!
$(\bullet) = \left(\underline{x}^*(t), \underline{u}^*(t), t\right)$

$k = 1,2,\ldots N$

1. One-step ahead prediction:

2. Calculate Jacobians: $F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right), \; H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)$

3. Discretize state transition & input matrix:

4. Cov. matrix of state pred. error:

5. Kalman gain calculation:

6. Measurement update:

7. Cov. matrix of state estimation error:

Derivation of The Extended Kalman Filter

Linear Perturbation Equations

Derived continuous linear perturbation equations for our system:

$$\delta\dot{\underline{x}}(t) = F_x\left(\underline{x}^*(t),\underline{u}^*(t),t\right)\delta\underline{x}(t) + G\left(\underline{x}^*(t),t\right)\underline{w}(t)$$

$$\delta\underline{z}_n(t) = H_x\left(\underline{x}^*(t),\underline{u}^*(t),t\right)\delta\underline{x}(t)$$

$$\delta\underline{z}(t_k) = \delta\underline{z}_n(t_k) + \underline{v}(t_k)$$

The discrete perturbation linear state-variable model is the following:

$$\delta\underline{x}_{k+1} = \Phi_{k+1,k}(\bullet)\delta\underline{x}_k + \Gamma_{k+1,k}(\bullet)\underline{w}_{d,k}$$

$$\delta\underline{z}_{k+1} = H_x(\bullet)\delta\underline{x}_{k+1} + \underline{v}_{k+1}$$

Matlab allows us to use the command `c2d` for discretization:

```
[Psi, Gamma] = c2d(DFx, G, deltaT);
```

Matlab tip!

Where DFx is the Jacobian of f : DFx $= F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f(\underline{x}(t), \underline{u}(t), t)$

The Extended Kalman Filter

$k = 1,2,\ldots N$

1. One-step ahead prediction:
2. Calculate Jacobians: $F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right),\ H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)$
3. Discretize state transition & input matrix: $\Phi_{k+1,k}(\bullet),\ \Gamma_{k+1,k}(\bullet)$
4. Cov. matrix of state pred. error:
5. Kalman gain calculation:
6. Measurement update:
7. Cov. matrix of state estimation error:

Derivation of The Extended Kalman Filter

EKF Step 1: State prediction in nonlinear systems

Since the original system state equation is nonlinear, the one-stage ahead prediction step can be calculated directly by integrating the nonlinear state equation with the one step earlier estimate of the state vector.

Using $\underline{x}_k^* = \hat{\underline{x}}_{k,k}$ and $\underline{x}_{k+1}^* = \hat{\underline{x}}_{k+1,k}$ the state vector propagation is:

$$\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k,k} + \int_{t_k}^{t_{k+1}} \underline{f}\left(\hat{\underline{x}}_{k,k}, \underline{u}_k^*, t\right) dt$$

which can be calculated in Matlab using a numerical integration routine such as `ode45` :

Matlab tip!

```
[t x_k_1] = ode45(@kf_calcFx, [t_k t_kp1], x_k_1k_1);
```

The Extended Kalman Filter

$k = 1, 2, \ldots N$

1. One-step ahead prediction: $\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k,k} + \int_{t_k}^{t_{k+1}} \underline{f}\left(\hat{\underline{x}}_{k,k}, \underline{u}_k^*, t\right) dt$
2. Calculate Jacobians: $F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right),\ H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)$
3. Discretize state transition & input matrix: $\Phi_{k+1,k}(\bullet), \Gamma_{k+1,k}(\bullet)$
4. Cov. matrix of state pred. error:
5. Kalman gain calculation:
6. Measurement update:
7. Cov. matrix of state estimation error:

Derivation of The Extended Kalman Filter

The <u>state prediction covariance matrix</u> is calculated the same as for the linear Kalman Filter:

$$\boxed{P_{k+1,k}\left(\bullet\right)=\Phi_{k+1,k}\left(\bullet\right)P_{k,k}\Phi^{T}_{k+1,k}\left(\bullet\right)+\Gamma_{k+1,k}\left(\bullet\right)Q_{d,k}\Gamma^{T}_{k+1,k}\left(\bullet\right)}$$

The <u>Kalman gain</u> is calculated the same as for the linear KF, but uses the Jacobian of the output equation:

$$\boxed{K_{k+1}\left(\bullet\right)=P_{k+1,k}\left(\bullet\right)H_x^T\left(\bullet\right)\left[H_x\left(\bullet\right)P_{k+1,k}\left(\bullet\right)H_x^T\left(\bullet\right)+R_{k+1}\right]^{-1}}$$

The <u>covariance of the state estimate</u> is also calculated the same as for the linear KF, but also requires the Jacobian of the output equation:

$$\boxed{P_{k+1,k+1}\left(\bullet\right)=\left[I_n-K_{k+1}\left(\bullet\right)H_x\left(\bullet\right)\right]P_{k+1,k}\left(\bullet\right)}$$

The Extended Kalman Filter

$k = 1,2,\ldots N$

1. One-step ahead prediction: $\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k,k} + \int_{t_k}^{t_{k+1}} \underline{f}\left(\hat{\underline{x}}_{k,k}, \underline{u}_k^*, t\right)dt$
2. Calculate Jacobians: $F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t),\underline{u}(t),t\right),\ H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t),\underline{u}(t),t\right)$
3. Discretize state transition & input matrices: $\Phi_{k+1,k}(\bullet),\ \Gamma_{k+1,k}(\bullet)$
4. Cov. matrix of state pred. error: $P_{k+1,k}(\bullet) = \Phi_{k+1,k}(\bullet)P_{k,k}\Phi^T_{k+1,k}(\bullet) + \Gamma_{k+1,k}(\bullet)Q_{d,k}(\bullet)\Gamma^T_{k+1,k}(\bullet)$
5. Kalman gain calculation: $K_{k+1}(\bullet) = P_{k+1,k}(\bullet)H_x^T(\bullet)\left[H_x(\bullet)P_{k+1,k}(\bullet)H_x^T(\bullet) + R_{k+1}\right]^{-1}$
6. Measurement update:
7. Cov. matrix of state estimation error: $P_{k+1,k+1}(\bullet) = \left[I_n - K_{k+1}(\bullet)H_x(\bullet)\right]P_{k+1,k}(\bullet)$

completed

Derivation of The Extended Kalman Filter

EKF Step 6: State update equation

$$\delta\underline{\hat{x}}_{k+1,k+1} = \delta\underline{\hat{x}}_{k+1,k} + K_{k+1}(\bullet)\left[\delta\underline{z}_{k+1} - H_x(\bullet)\delta\underline{\hat{x}}_{k+1,k}\right]$$

Note that because we linearize about $\underline{x}^*_{k+1} = \underline{\hat{x}}_{k+1,k}$ we have:

$$\delta\underline{\hat{x}}_{k+1,k} = \underline{\hat{x}}_{k+1,k} - \underline{x}^*_{k+1} = 0$$

$$\delta\underline{\hat{x}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k+1} - \underline{x}^*_{k+1} = \underline{\hat{x}}_{k+1,k+1} - \underline{\hat{x}}_{k+1,k}$$

Similarly, the perturbed measurements can be written as:

$$\delta\underline{z}_{k+1} = \underline{z}_{k+1} - \underline{z}^*_{k+1} = \underline{z}_{k+1} - \underline{h}\left(\underline{x}^*_{k+1}, \underline{u}^*_{k+1}, k+1\right) = \underline{z}_{k+1} - \underline{h}\left(\underline{\hat{x}}_{k+1,k}, \underline{u}^*_{k+1}\right)$$

Substitution of $\delta\underline{\hat{x}}_{k+1,k} = 0$, $\delta\underline{\hat{x}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k+1} - \underline{\hat{x}}_{k+1,k}$,
$\delta\underline{z}_{k+1} = \underline{z}_{k+1} - \underline{h}\left(\underline{\hat{x}}_{k+1,k}, \underline{u}^*_{k+1}\right)$ in the expression for $\delta\underline{\hat{x}}_{k+1,k+1}$ results in:

$$\boxed{\underline{\hat{x}}_{k+1,k+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1}(\bullet)\left[\underline{z}_{k+1} - \underline{h}\left(\underline{\hat{x}}_{k+1,k}, \underline{u}^*_{k+1}\right)\right]}$$

The Extended Kalman Filter

$k = 1,2,\ldots N$

1. One-step ahead prediction: $\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k,k} + \int_{t_k}^{t_{k+1}} \underline{f}\left(\hat{\underline{x}}_{k,k}, \underline{u}_k^*, t\right) dt$

2. Calculate Jacobians: $F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right),\ H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)$

3. Discretize state transition & input matrices: $\Phi_{k+1,k}(\bullet),\ \Gamma_{k+1,k}(\bullet)$

4. Cov. matrix of state pred. error: $P_{k+1,k}(\bullet) = \Phi_{k+1,k}(\bullet) P_{k,k} \Phi_{k+1,k}^T(\bullet) + \Gamma_{k+1,k}(\bullet) Q_{d,k}(\bullet) \Gamma_{k+1,k}^T(\bullet)$

5. Kalman gain calculation: $K_{k+1}(\bullet) = P_{k+1,k}(\bullet) H_x^T(\bullet)\left[H_x(\bullet) P_{k+1,k}(\bullet) H_x^T(\bullet) + R_{k+1}\right]^{-1}$

6. Measurement update: $\hat{\underline{x}}_{k+1,k+1} = \hat{\underline{x}}_{k+1,k} + K_{k+1}(\bullet)\left[\underline{z}_{k+1} - \underline{h}\left(\hat{\underline{x}}_{k+1,k}, \underline{u}_{k+1}^*\right)\right]$

7. Cov. matrix of state estimation error: $P_{k+1,k+1}(\bullet) = \left[I_n - K_{k+1}(\bullet) H_x(\bullet)\right] P_{k+1,k}(\bullet)$

The Extended Kalman Filter

Example 3.4: Extended Kalman filter

Consider the nonlinear system:

$$\dot{x}(t) = -0.3\cos^3(x(t)) + w(t); \qquad x(0) = 3$$
$$z_n(t) = x^3(t)$$
$$z(t_k) = z_n(t_k) + v(t_k); \qquad k = 1, 2, \dots$$

with noise statistics:

$$E\{w(t)\} = 0, \qquad E\{w(t)w(\tau)\} = Q; \qquad Q = 100$$
$$E\{v(t_k)\} = 0, \qquad E\{v(t_k)v(t_l)\} = R; \qquad R = 1$$

and initial estimates for the state and state estimate covariance 'matrix':

$$\hat{x}_0 = \hat{x}(0|0) = 10, \qquad P(0|0) = 100;$$

and simulation parameters:

$$\Delta T = 0.01, \qquad N = 1000$$

The Extended Kalman Filter

Example 3.5: Extended Kalman filter

Matlab Demo

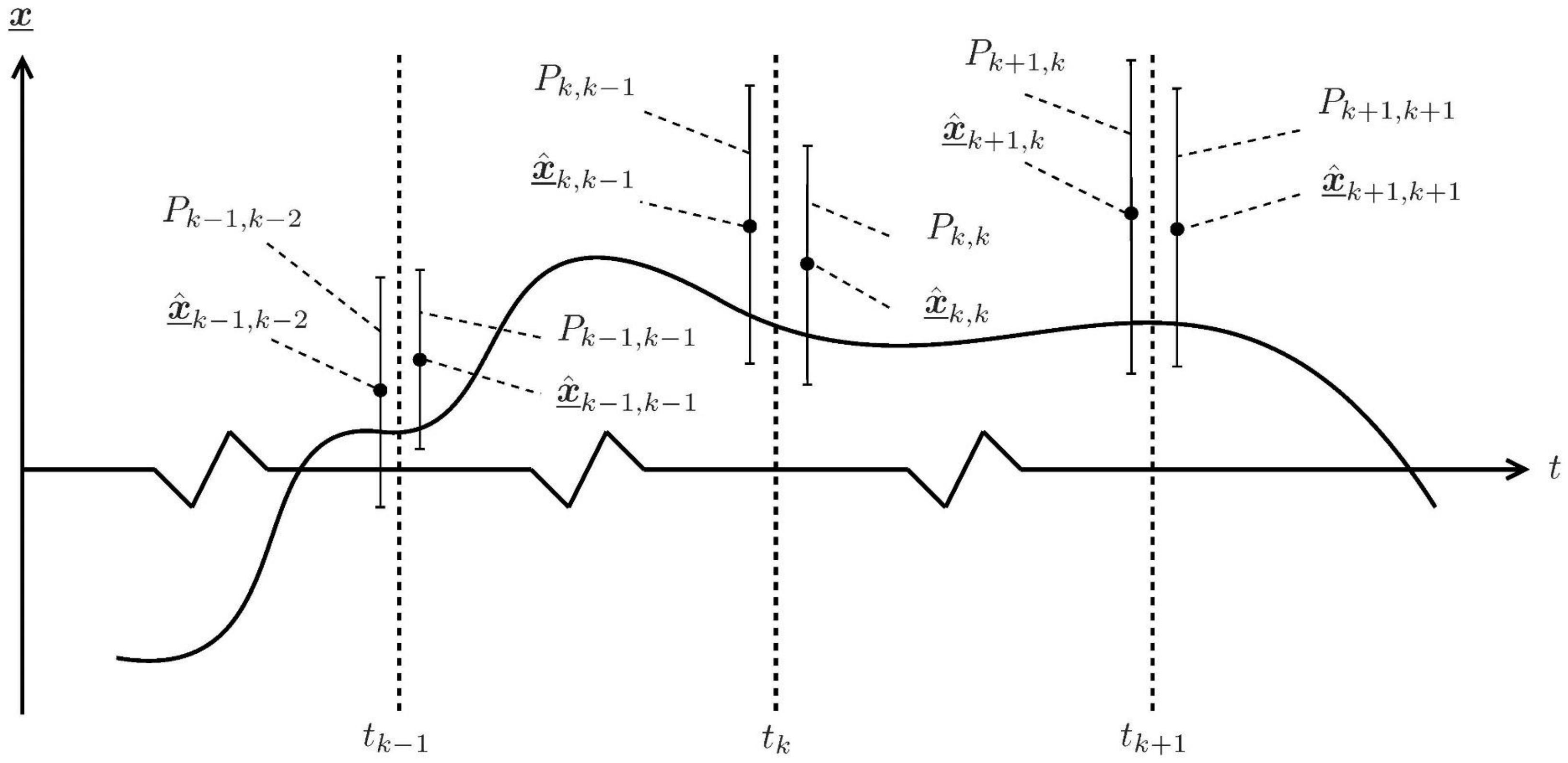
Derivation of the Iterated EKF

Limitations of the EKF

Although EKF has been applied widely, it **does not provide the guarantee of global convergence** to its optimal solution.
This is due the fact that EKF only uses the first order approximation of the nonlinear system. In order to ensure the approximation is relevant, the perturbed state vector should be small. This means that the initial condition to start EKF should be selected fairly close to its optimal solution. Otherwise the EKF might converge slowly or even diverge. This limits the application of EKF. To improve the convergence of EKF, the Iterated EKF, or **IEKF** is introduced.

In many applications, observation models present higher nonlinearity than dynamic models. The IEKF specifically tries to improve the convergence of the EKF using local iterations through measurement update steps. These local iterations ensure more accurate linearizations using updated states.

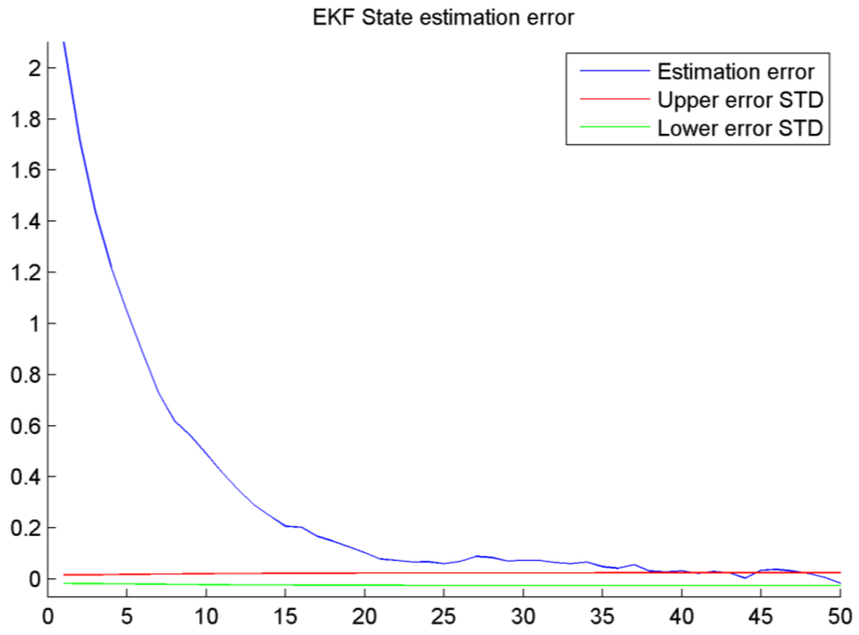
The Extended Kalman Filter



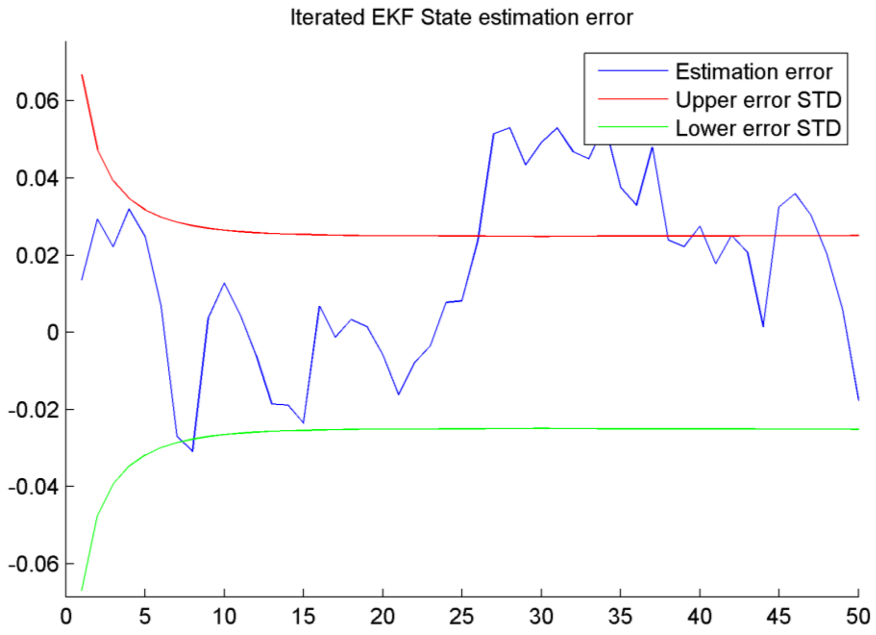
$\underline{x}$
$P_{k-1,k-2}$
$\hat{\underline{x}}_{k-1,k-2}$
$P_{k-1,k-1}$
$\hat{\underline{x}}_{k-1,k-1}$
$P_{k,k-1}$
$\hat{\underline{x}}_{k,k-1}$
$P_{k,k}$
$\hat{\underline{x}}_{k,k}$
$P_{k+1,k}$
$\hat{\underline{x}}_{k+1,k}$
$P_{k+1,k+1}$
$\hat{\underline{x}}_{k+1,k+1}$
t
t_{k-1}
t_k
t_{k+1}

Derivation of the Iterated EKF

Limitations of the EKF

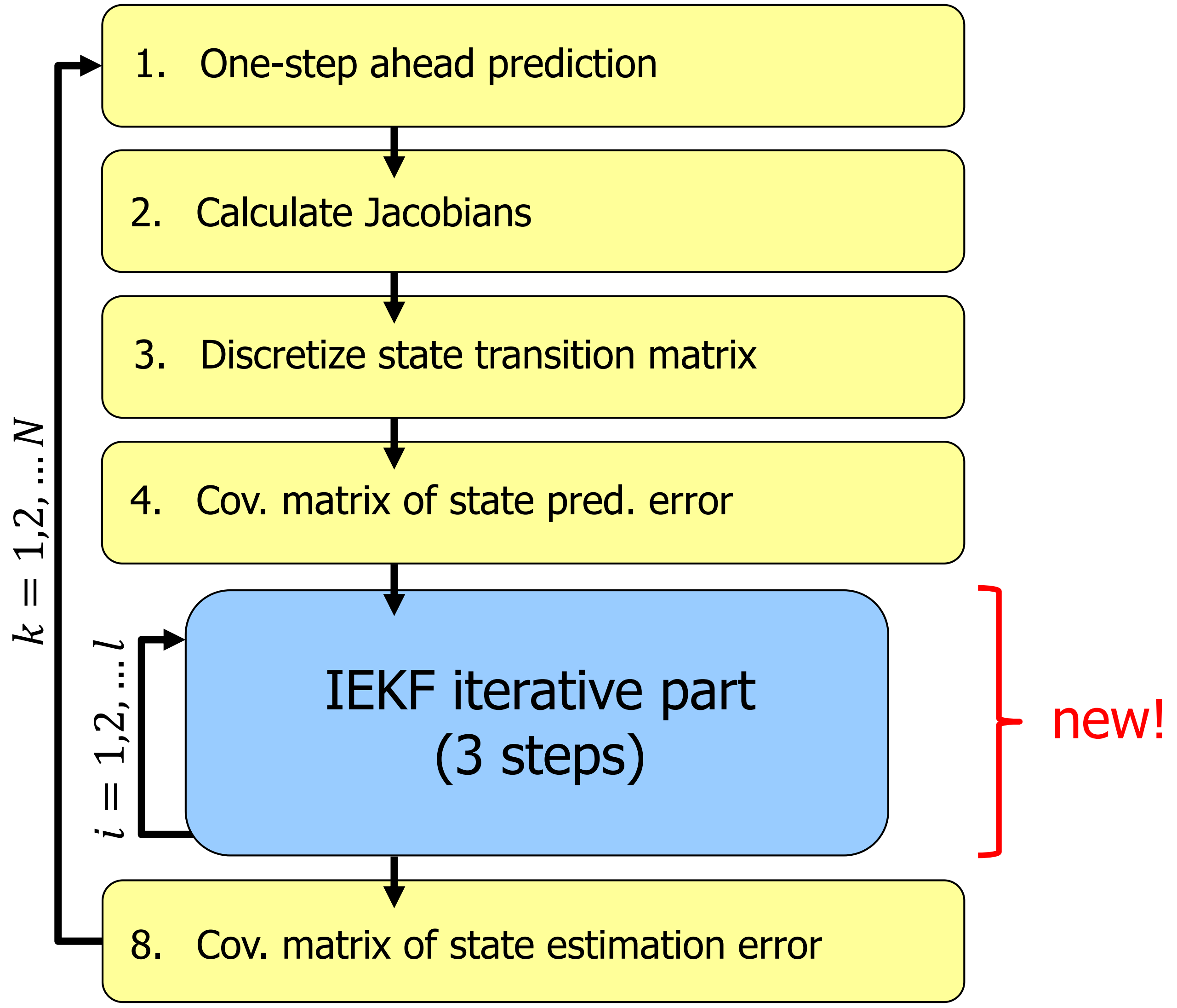


EKF state estimation error

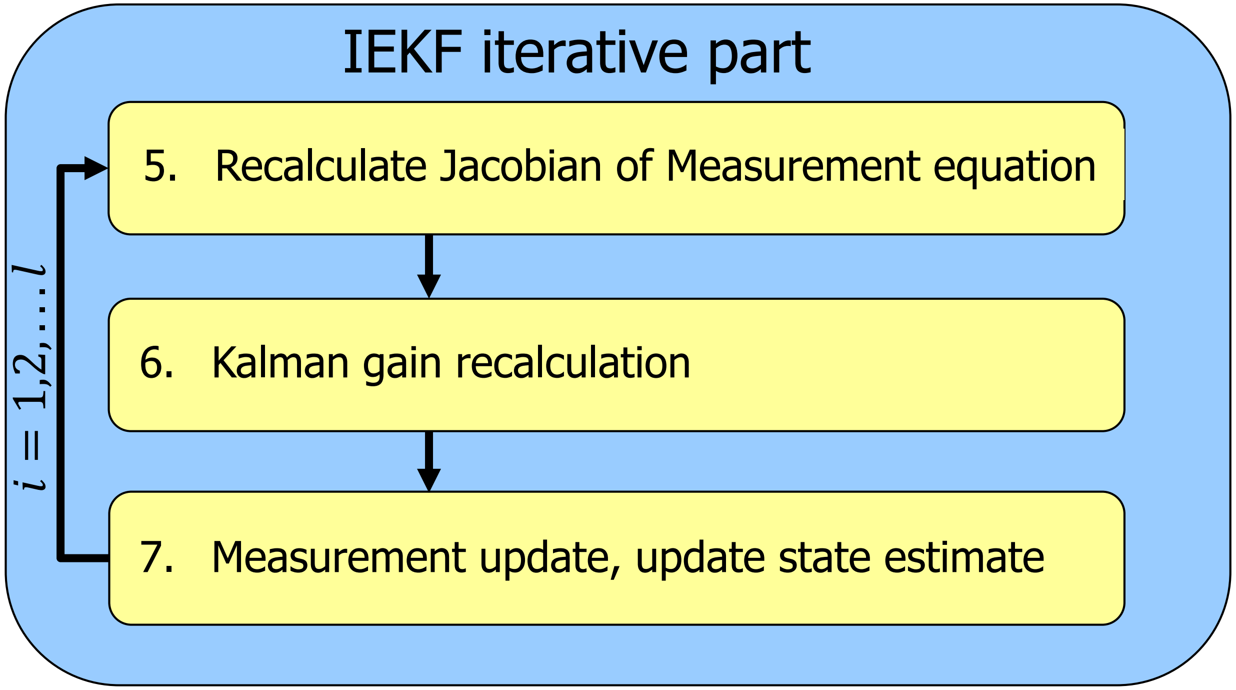


Iterated EKF state estimation error

The Iterated Extended Kalman Filter


1. One-step ahead prediction
2. Calculate Jacobians
3. Discretize state transition matrix
4. Cov. matrix of state pred. error
IEKF iterative part (3 steps)
8. Cov. matrix of state estimation error
$k = 1,2,\ldots N$
$i = 1,2,\ldots l$
new!

The Iterated Extended Kalman Filter


IEKF iterative part
5. Recalculate Jacobian of Measurement equation
6. Kalman gain recalculation
7. Measurement update, update state estimate
$i = 1, 2, \ldots l$

Derivation of the Iterated EKF

The first 4 steps of the IEKF are identical to the EKF:

IEKF Step 1: EKF State Prediction:

$$\boxed{\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k,k} + \int_{t_k}^{t_{k+1}} \underline{f}\left(\hat{\underline{x}}_{k,k}, \underline{u}_k^*, t\right) dt}$$

IEKF Step 2: Calculate state transition and observation Jacobians:

$$\boxed{F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right)\Big|_{\underline{x}(t)=\hat{\underline{x}}_{k,k}, \underline{u}(t)=\underline{u}_{k,k}, t}} \quad \boxed{H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)\Big|_{\underline{x}(t)=\hat{\underline{x}}_{k+1,k}, \underline{u}(t)=\underline{u}_{k+1,k}, t}}$$

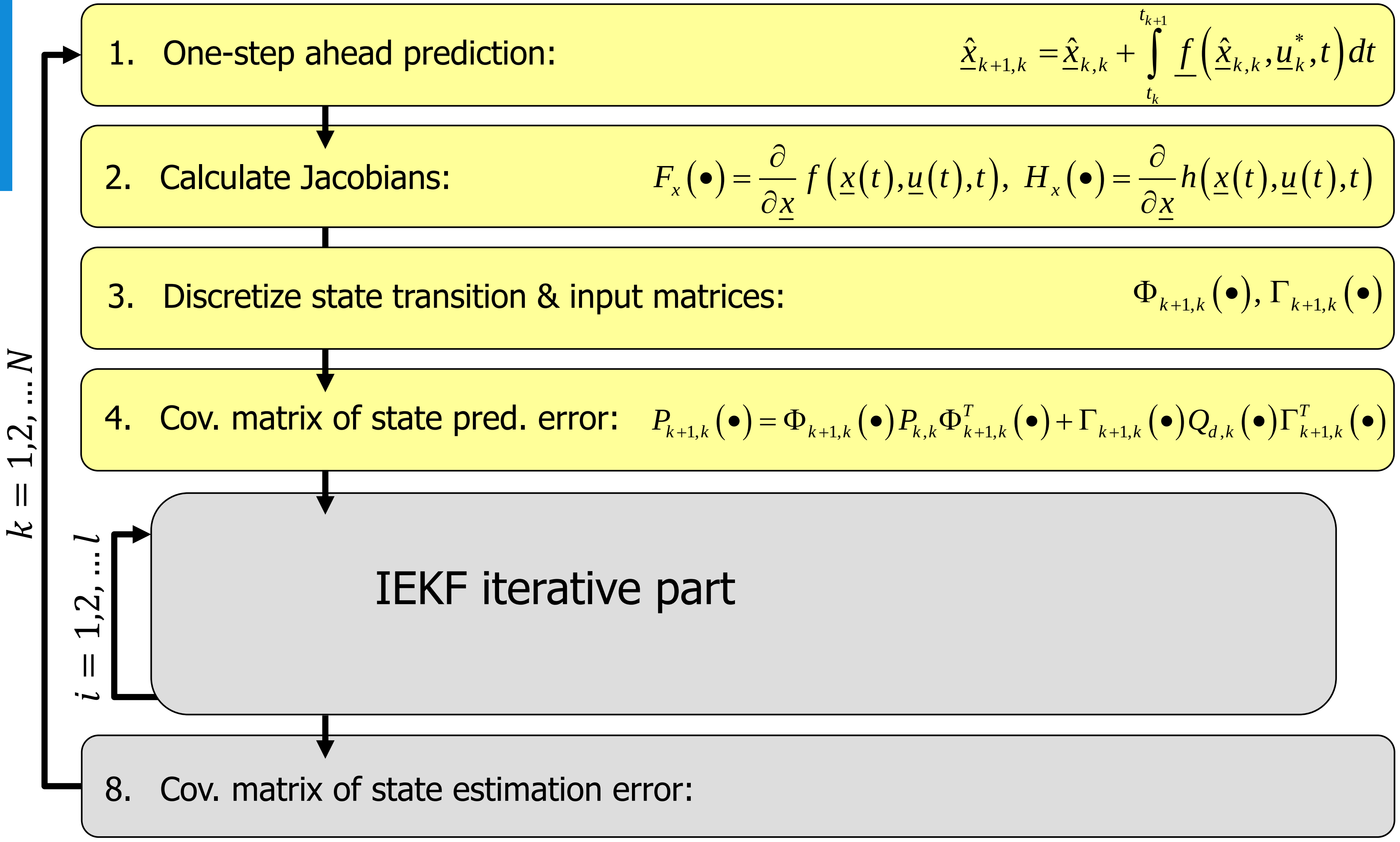
IEKF Step 3: Discretize state transition and input equation:

$$\boxed{\Phi_{k+1,k}(\bullet) = \exp\left[F_x(\bullet)(t_{k+1} - t_k)\right]} \quad \boxed{\Gamma_{k+1,k}(\bullet) = \exp\left[F_x(\bullet)(t_{k+1} - t_k)\right] \cdot F_u}$$

IEKF Step 4: EKF Covariance matrix of state prediction error:

$$\boxed{P_{k+1,k}(\bullet) = \Phi_{k+1,k}(\bullet) P_{k,k} \Phi_{k+1,k}^T(\bullet) + \Gamma_{k+1,k}(\bullet) Q_{d,k}(\bullet) \Gamma_{k+1,k}^T(\bullet)}$$

The Iterated Extended Kalman Filter



1. One-step ahead prediction: $\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k,k} + \int_{t_k}^{t_{k+1}} \underline{f}\left(\hat{\underline{x}}_{k,k}, \underline{u}_k^*, t\right) dt$

2. Calculate Jacobians: $F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right),\ H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)$

3. Discretize state transition & input matrices: $\Phi_{k+1,k}(\bullet),\ \Gamma_{k+1,k}(\bullet)$

4. Cov. matrix of state pred. error: $P_{k+1,k}(\bullet) = \Phi_{k+1,k}(\bullet) P_{k,k} \Phi^T_{k+1,k}(\bullet) + \Gamma_{k+1,k}(\bullet) Q_{d,k}(\bullet) \Gamma^T_{k+1,k}(\bullet)$

IEKF iterative part

8. Cov. matrix of state estimation error:

Derivation of the Iterated EKF

IEKF: Iterative Steps

After Step 4, we define an **iterator** to be equal to the perturbed nominal state:

$$\underline{\eta}_i = \underline{x}^*_{k+1}$$

The idea now is to update the nominal states using the current measurement and re-linearizing the system using the improved nominal states.

The iterator is initialized using the state estimation:

$$\underline{\eta}_1 = \hat{\underline{x}}_{k+1,k}$$

The perturbation equations can now be written as follows:

$$\delta\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k+1,k} - \underline{\eta}_i$$

$$\delta\hat{\underline{x}}_{k+1,k+1} = \hat{\underline{x}}_{k+1,k+1} - \underline{x}^*_{k+1} = \underline{\eta}_{i+1} - \underline{\eta}_i$$

Derivation of the Iterated EKF

IEKF: Iterative Steps

Recall the earlier found perturbation equation:

$$\delta\hat{\underline{x}}_{k+1,k+1} = \delta\hat{\underline{x}}_{k+1,k} + K_{k+1}(\bullet)\left[\delta\underline{z}_{k+1} - H_x(\bullet)\delta\hat{\underline{x}}_{k+1,k}\right]$$

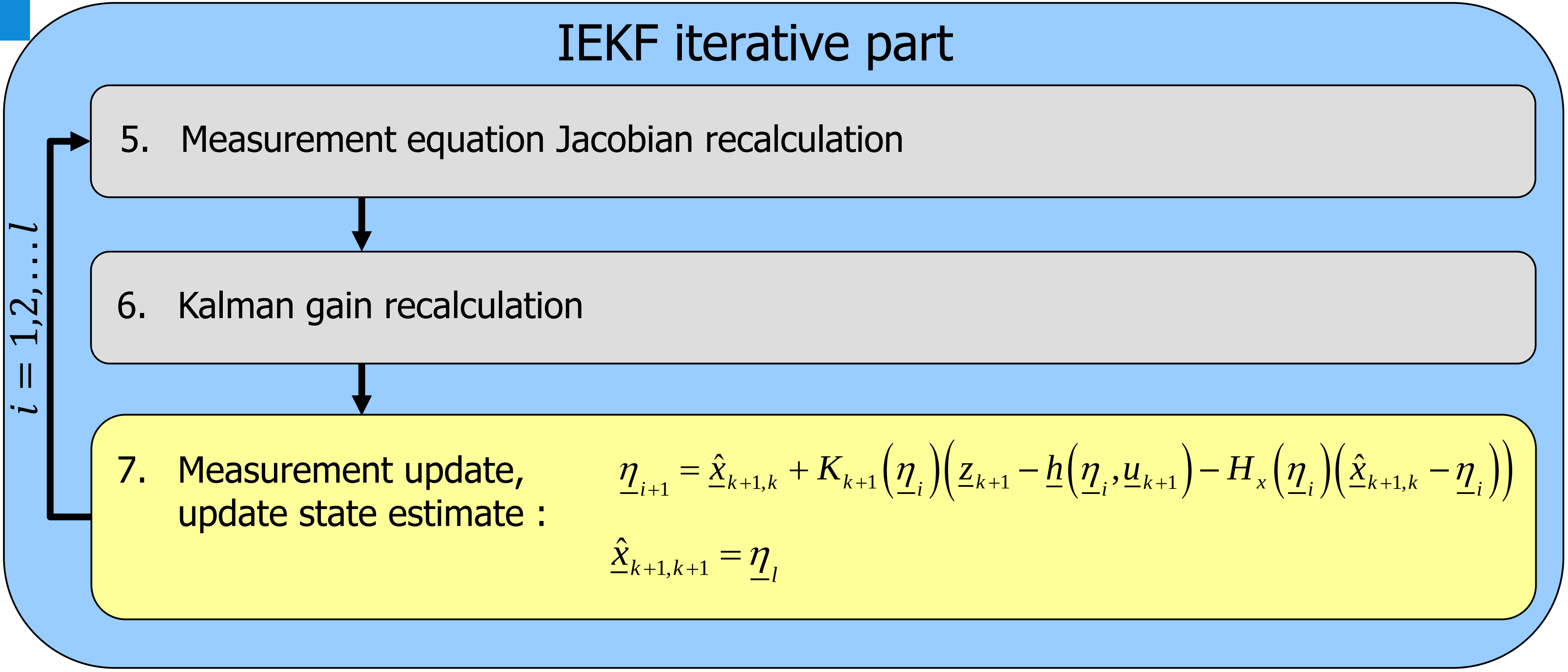
Now substitute $\delta\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k+1,k} - \underline{\eta}_i$ and the earlier found relation $\delta\underline{z}_{k+1} = \underline{z}_{k+1} - \underline{h}(\underline{\eta}_i, \underline{u}^*_{k+1})$ in this expression, which results in:

$$\delta\hat{\underline{x}}_{k+1,k+1} = \hat{\underline{x}}_{k+1,k} - \underline{\eta}_i + K_{k+1}(\underline{\eta}_i)\left[\underline{z}_{k+1} - \underline{h}(\underline{\eta}_i, \underline{u}_{k+1}) - H_x(\underline{\eta}_i)(\hat{\underline{x}}_{k+1,k} - \underline{\eta}_i)\right]$$

Substitution of $\delta\hat{\underline{x}}_{k+1,k+1} = \underline{\eta}_{i+1} - \underline{\eta}_i$ in this expression leads to the **iterative version of the measurement update equation**:

$$\underline{\eta}_{i+1} = \hat{\underline{x}}_{k+1,k} + K_{k+1}(\underline{\eta}_i)\left(\underline{z}_{k+1} - \underline{h}(\underline{\eta}_i, \underline{u}_{k+1}) - H_x(\underline{\eta}_i)(\hat{\underline{x}}_{k+1,k} - \underline{\eta}_i)\right), i = 1, \dots, l$$

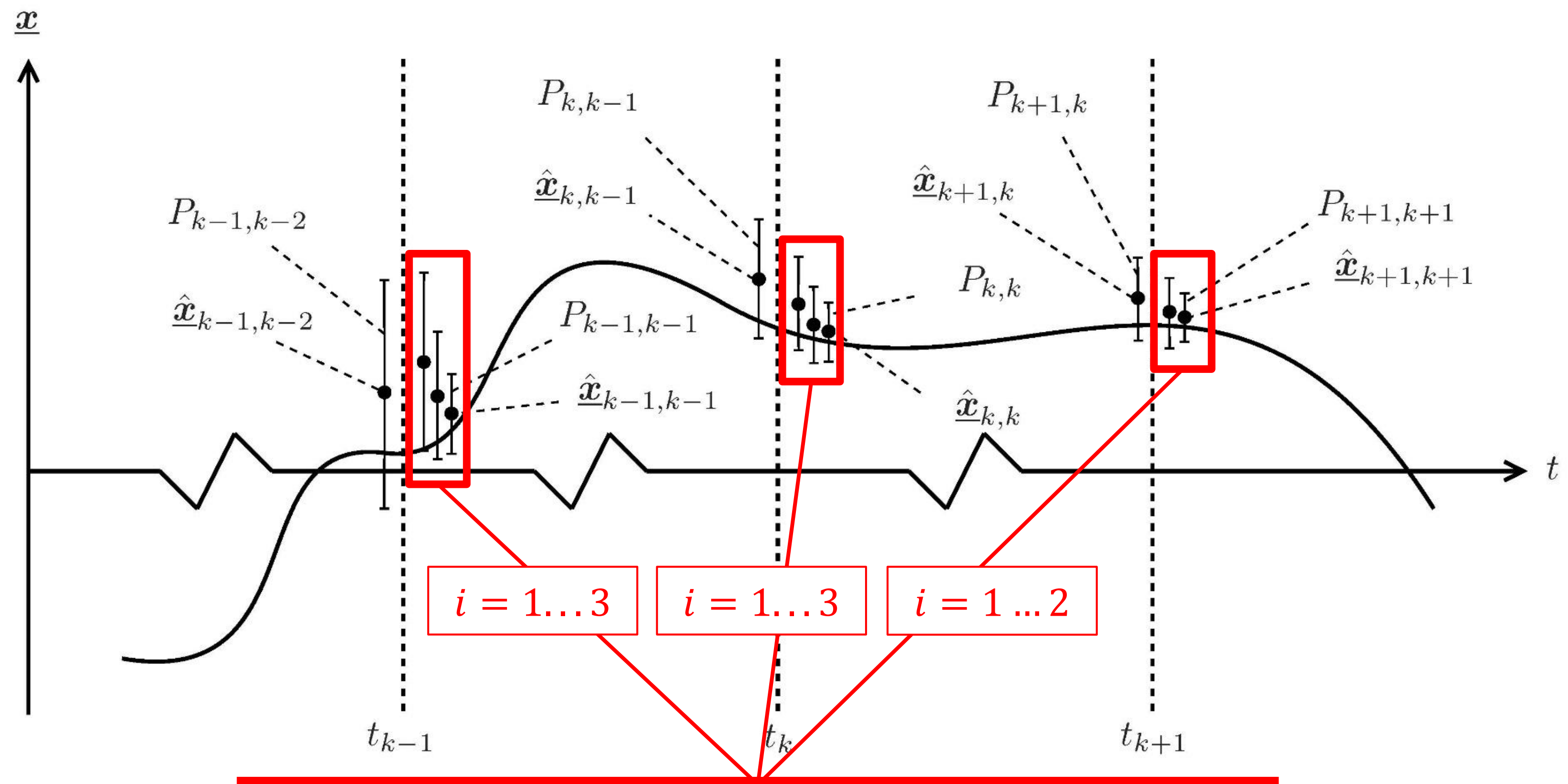
The Iterated Extended Kalman Filter


IEKF iterative part
5. Measurement equation Jacobian recalculation
6. Kalman gain recalculation
7. Measurement update, update state estimate :
$i = 1,2,\ldots l$

$$\underline{\eta}_{i+1} = \hat{\underline{x}}_{k+1,k} + K_{k+1}\left(\underline{\eta}_i\right)\left(\underline{z}_{k+1} - \underline{h}\left(\underline{\eta}_i, \underline{u}_{k+1}\right) - H_x\left(\underline{\eta}_i\right)\left(\hat{\underline{x}}_{k+1,k} - \underline{\eta}_i\right)\right)$$

$$\hat{\underline{x}}_{k+1,k+1} = \underline{\eta}_l$$

The Iterated Extended Kalman Filter



$\underline{x}$
t
$P_{k-1,k-2}$
$\hat{\underline{x}}_{k-1,k-2}$
$P_{k,k-1}$
$\hat{\underline{x}}_{k,k-1}$
$P_{k-1,k-1}$
$\hat{\underline{x}}_{k-1,k-1}$
$P_{k+1,k}$
$\hat{\underline{x}}_{k+1,k}$
$P_{k,k}$
$\hat{\underline{x}}_{k,k}$
$P_{k+1,k+1}$
$\hat{\underline{x}}_{k+1,k+1}$
$i = 1\ldots3$
$i = 1\ldots3$
$i = 1\ldots2$
t_{k-1}
t_k
t_{k+1}

$$\boxed{\underline{\eta}_{i+1} = \underline{\hat{x}}_{k+1,k} + K_{k+1}\left(\underline{\eta}_i\right)\left(\underline{z}_{k+1} - \underline{h}\left(\underline{\eta}_i, \underline{u}_{k+1}\right) - H_x\left(\underline{\eta}_i\right)\left(\underline{\hat{x}}_{k+1,k} - \underline{\eta}_i\right)\right)}$$

Derivation of the Iterated EKF

IEKF Step: Iterative Steps

The iteration is terminated when the following performance measure is reached:

$$\frac{\left\| \underline{\eta}_l - \underline{\eta}_{l-1} \right\|}{\left\| \underline{\eta}_{l-1} \right\|} \leq \varepsilon$$

with ε a small number (e.g. 1e-10).

Finally, when the above performance measure is reached, the new optimal state estimate becomes:

$$\hat{\underline{x}}_{k+1,k+1} = \underline{\eta}_l$$

Note that the Jacobian matrix $H_x(\underline{\eta}_i)$ and the Kalman gain $K_{k+1}(\underline{\eta}_i)$ must be updated inside the iterative loop!

Derivation of the Iterated EKF

The Jacobian of the observation equation must be recalculated at each iteration (and for each new $\underline{\eta}_i$):

$$\boxed{H_x\left(\underline{\eta}_i\right) = \frac{\partial}{\partial \underline{x}} h\left(\underline{\eta}_i, \underline{u}(t), t\right)}$$

The Kalman gain must also be recalculated at each iteration, using the new value found for the Jacobian of the observation matrix $H_x\left(\underline{\eta}_i\right)$.

$$\boxed{K_{k+1}\left(\underline{\eta}_i\right) = P_{k+1,k}(\bullet) H_x^T\left(\underline{\eta}_i\right)\left[H_x\left(\underline{\eta}_i\right) P_{k+1,k}(\bullet) H_x^T\left(\underline{\eta}_i\right) + R_{k+1}\right]^{-1}}$$

The Iterated Extended Kalman Filter

IEKF iterative part

$i = 1,2,\ldots l$

5. Measurement equation Jacobian recalculation: $H_x\left(\underline{\eta}_i\right) = \dfrac{\partial}{\partial \underline{x}} h\left(\underline{\eta}_i, \underline{u}(t), t\right)$

6. Kalman gain recalculation: $K_{k+1}\left(\underline{\eta}_i\right) = P_{k+1,k}(\bullet) H_x^T\left(\underline{\eta}_i\right)\left[H_x\left(\underline{\eta}_i\right) P_{k+1,k}(\bullet) H_x^T\left(\underline{\eta}_i\right) + R_{k+1}\right]^{-1}$

7. Measurement update, update state estimate :

$$\underline{\eta}_{i+1} = \hat{\underline{x}}_{k+1,k} + K_{k+1}\left(\underline{\eta}_i\right)\left(\underline{z}_{k+1} - \underline{h}\left(\underline{\eta}_i, \underline{u}_{k+1}\right) - H_x\left(\underline{\eta}_i\right)\left(\hat{\underline{x}}_{k+1,k} - \underline{\eta}_i\right)\right)$$

$$\hat{\underline{x}}_{k+1,k+1} = \underline{\eta}_l$$

completed

Derivation of the Iterated EKF

The final step of the IEKF is the calculation of the covariance matrix of the state estimate. This is done exactly equal as with the EKF, but uses the final value for the iterate $\underline{\eta}_l$, which is actually equal to $\underline{\hat{x}}_{k+1,k+1}$.

$$\boxed{\begin{aligned}P_{k+1,k+1}\left(\underline{\eta}_l\right) &= \left[I_n - K_{k+1}\left(\underline{\eta}_l\right)H_x\left(\underline{\eta}_l\right)\right]P_{k+1,k}\left(\underline{\eta}_l\right)\left[I_n - K_{k+1}\left(\underline{\eta}_l\right)H_x\left(\underline{\eta}_l\right)\right]^T + \\ &+K_{k+1}\left(\underline{\eta}_l\right)R_{k+1}K_{k+1}^T\left(\underline{\eta}_l\right)\end{aligned}}$$

This concludes the derivation of the Iterated Extended Kalman filter!

The Iterated Extended Kalman Filter

$k = 1,2,\ldots N$

1. One-step ahead prediction: $\hat{\underline{x}}_{k+1,k} = \hat{\underline{x}}_{k,k} + \int_{t_k}^{t_{k+1}} \underline{f}\left(\hat{\underline{x}}_{k,k}, \underline{u}^*_k, t\right) dt$

2. Calculate Jacobians: $F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t),\underline{u}(t),t\right),\ H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t),\underline{u}(t),t\right)$

3. Discretize state transition & input matrices: $\Phi_{k+1,k}(\bullet), \Gamma_{k+1,k}(\bullet)$

4. Cov. matrix of state pred. error: $P_{k+1,k}(\bullet) = \Phi_{k+1,k}(\bullet) P_{k,k} \Phi^T_{k+1,k}(\bullet) + \Gamma_{k+1,k}(\bullet) Q_{d,k}(\bullet) \Gamma^T_{k+1,k}(\bullet)$

$i = 1,2,\ldots l$

IEKF iterative part (see next sheet)

8. Cov. matrix of state estimation error:

$$P_{k+1,k+1}\left(\underline{\eta}_l\right) = \left[I_n - K_{k+1}\left(\underline{\eta}_l\right) H_x\left(\underline{\eta}_l\right)\right] P_{k+1,k}\left(\underline{\eta}_l\right) \left[I_n - K_{k+1}\left(\underline{\eta}_l\right) H_x\left(\underline{\eta}_l\right)\right]^T + \\ + K_{k+1}\left(\underline{\eta}_l\right) R_{k+1} K^T_{k+1}\left(\underline{\eta}_l\right)$$

The Iterated Extended Kalman Filter

IEKF iterative part

$i = 1,2,\ldots l$

5. Measurement equation Jacobian recalculation: $H_x(\underline{\eta}_i) = \frac{\partial}{\partial \underline{x}} h(\underline{\eta}_i, \underline{u}(t), t)$

6. Kalman gain recalculation: $K_{k+1}(\underline{\eta}_i) = P_{k+1,k}(\bullet) H_x^T(\underline{\eta}_i)\left[H_x(\underline{\eta}_i) P_{k+1,k}(\bullet) H_x^T(\underline{\eta}_i) + R_{k+1}\right]^{-1}$

7. Measurement update, update state estimate :

$$\underline{\eta}_{i+1} = \hat{\underline{x}}_{k+1,k} + K_{k+1}(\underline{\eta}_i)\left(\underline{z}_{k+1} - \underline{h}(\underline{\eta}_i, \underline{u}_{k+1}) - H_x(\underline{\eta}_i)\left(\hat{\underline{x}}_{k+1,k} - \underline{\eta}_i\right)\right)$$

$$\hat{\underline{x}}_{k+1,k+1} = \underline{\eta}_l$$

The Iterated Extended Kalman Filter

Example 3.6: Iterated Extended Kalman filter

Matlab Demo

Sensor Fusion

The Kalman filter can be used to facilitate **sensor fusion**.

Sensor fusion is the combining of different sensor sources to obtain a better estimate of a state variable.

- For example, GPS measurements can be used in combination with inertial sensors (gyros, accelerometers) to obtain a better estimate of attitude, position, velocity and acceleration.

By combining sensors of different types and working principles, <u>a new virtual sensor</u> can be produced that combines all strengths of all sensors without the weaknesses.

- For example, the strengths of inertial sensors (high sample rates, high short term accuracy) can be combined with those of GPS (accuracy does not deteriorate over time), to overcome the weaknesses of both (bias drift for inertial sensors, no guarantied service and low short term accuracy for GPS).

Sensor Fusion

Inertial sensor shortcomings:

- Inertial sensor measurements contain biases due to e.g. misalignment of gyro and accelerometer primary axes, manufacturing errors, temperature effects, nonlinear scaling effects.
- Next to biases, inertial sensor measurements are subject to sensor noise.
- The measured accelerations and rotational rates are therefore not the “real” rates:

$$\left.\begin{aligned}
A_{x_m} &= A_x + \lambda_x + w_x \\
A_{y_m} &= A_y + \lambda_y + w_y \\
A_{z_m} &= A_z + \lambda_z + w_z \\
p_m &= p + \lambda_p + w_p \\
q_m &= q + \lambda_q + w_q \\
r_m &= r + \lambda_r + w_r
\end{aligned}\right\} \text{with } \lambda \text{ bias terms, and } w \text{ sensor noise}$$

Sensor Fusion

Inertial sensor shortcomings

$$
\begin{aligned}
A_{x_m} &= A_x + \lambda_x + w_x \\
A_{y_m} &= A_y + \lambda_y + w_y \\
A_{z_m} &= A_z + \lambda_z + w_z \\
p_m &= p + \lambda_p + w_p \\
q_m &= q + \lambda_q + w_q \\
r_m &= r + \lambda_r + w_r
\end{aligned}
$$

measured acceleration & rate

real acceleration & rate

bias term

white sensor noise

Sensor Fusion

Example 3.7: Sensor fusion

Consider the simplified the equations of motion of a rigid aircraft in symmetrical flight conditions:

$$\dot{V}_x = A_x - g\sin\theta - qV_z,$$
$$\dot{V}_z = A_z - g\cos\theta + qV_x,$$
$$\dot{q} = \frac{M}{J_{yy}},$$
$$\dot{\theta} = q$$
$$\dot{z}_E = -V_x\sin(\theta) + V_z\cos(\theta)$$

with V_x, V_z body velocities, q the pitch rate, θ the pitch angle, g the acceleration of gravity, M the total aerodynamic moment, J_{yy} the moment of inertia around the y-axis, and z_E the z-axis coordinate above the earth.

Sensor Fusion

Example 3.7: Sensor fusion

Lets assume we have 2 linear accelerometers, 1 pitch rate gyro, 1 pitot static tube, a pitch angle sensor, and an (barometric) altitude sensor:

$$
\begin{aligned}
A_{x_m} &= A_x + \lambda_x + w_x, \\
A_{z_m} &= A_z + \lambda_z + w_z, \\
q_m &= q + \lambda_q + w_q, \\
V_m &= \sqrt{V_x^2 + V_z^2} + v_V, \\
\theta_m &= \theta + v_\theta, \\
\Delta h_m &= -z_E + v_{\Delta h}
\end{aligned}
$$

Sensor Fusion

Example 3.7: Sensor fusion

With the sensor biases and noise, the equations of motion become:

$$\dot{V}_x = (A_{xm} - \lambda_x - w_x) - g\sin\theta - (q_m - \lambda_q - w_q)V_z$$

$$\dot{V}_x = (A_{zm} - \lambda_z - w_z) + g\cos\theta + (q_m - \lambda_q - w_q)V_x$$

$$\dot{\theta} = (q_m - \lambda_q - w_q)$$

$$\dot{z}_E = -V_x \sin\theta + V_z \cos\theta$$

The state vector is: $\underline{x} = \begin{bmatrix} V_x & V_z & \theta & z_E \end{bmatrix}^T$

The input vector is: $u_m = \begin{bmatrix} A_{xm} & A_{zm} & q_m \end{bmatrix}^T$

The measurement vector is: $z_m = \begin{bmatrix} V_m & \theta_m & \Delta h_m \end{bmatrix}^T$

Sensor Fusion

Example 3.7: Sensor fusion: State transition equation

The state transition equation can be written as:

$$\begin{bmatrix} \dot{V}_x \\ \dot{V}_z \\ \dot{\theta} \\ \dot{z}_E \end{bmatrix} = \begin{bmatrix} (A_x - \lambda_x) - g\sin\theta - (q - \lambda_q)V_z - w_x + w_q V_z \\ (A_z - \lambda_z) + g\cos\theta + (q - \lambda_q)V_x - w_z - w_q V_x \\ (q - \lambda_q) - w_q \\ -V_x \sin\theta + V_z \cos\theta \end{bmatrix}$$

Which when written in the form $\underline{\dot{x}} = f(\underline{x}, \underline{u}) + Gw$ becomes:

$$\begin{bmatrix} \dot{V}_x \\ \dot{V}_z \\ \dot{\theta} \\ \dot{z}_E \end{bmatrix} = \begin{bmatrix} (A_x - \lambda_x) - g\sin\theta - (q - \lambda_q)V_z \\ (A_z - \lambda_z) + g\cos\theta + (q - \lambda_q)V_x \\ (q - \lambda_q) \\ -V_x \sin\theta + V_z \cos\theta \end{bmatrix} + \begin{bmatrix} -1 & 0 & V_z \\ 0 & -1 & -V_x \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_z \\ w_q \end{bmatrix}$$

Sensor Fusion

Example 3.7: Sensor fusion: Jacobian of state transition equation

The Jacobian of the state transition equation is:

$$F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right)$$

$$= \begin{bmatrix} 0 & -q+\lambda_q & -g\cos\theta & 0 \\ q-\lambda_q & 0 & -g\sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ -\sin\theta & \cos\theta & -V_x\cos\theta - V_z\sin\theta & 0 \end{bmatrix}$$

Sensor Fusion

Example 3.7: Sensor fusion: Measurement equation

Using the measurement vector $z_m = \begin{bmatrix} V_m & \theta_m & \Delta h_m \end{bmatrix}^T$

The observation equation can be written in the form $\underline{z}_m = h(\underline{x}, \underline{u}) + \underline{v}$:

$$\begin{bmatrix} V_m \\ \theta_m \\ \Delta h_m \end{bmatrix} = \begin{bmatrix} \sqrt{V_x^2 + V_z^2} \\ \theta \\ -z_E \end{bmatrix} + \begin{bmatrix} v_V \\ v_\theta \\ v_{\Delta h} \end{bmatrix}$$

which can be interpreted as a sensor for V_m (pitot tube), a sensor for θ_m (vertical gyro), and a sensor for Δh_m (barometric altitude)

Sensor Fusion

Example 3.7: Sensor fusion: Jacobian of observation equation

The Jacobian of the observation equation is:

$$
\begin{aligned}
H_x(\bullet) &= \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right) \\
&= \begin{bmatrix} \frac{V_x}{\sqrt{V_x^2 + V_z^2}} & \frac{V_z}{\sqrt{V_x^2 + V_z^2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ & & & \end{bmatrix}
\end{aligned}
$$

Sensor Fusion

Example 3.9: Defining the noise covariance matrices

Until now, we have not clearly shown how to set the noise covariance matrices Q and R. What we can do is use our knowledge from calibrating the sensors to get initial estimates for Q and R.

Let our input vector be defined as $u_m = [A_{xm} \quad A_{zm} \quad q_m]^T$ with standard deviations $\sigma_{Ax}, \sigma_{Az}, \sigma_q$ respectively.
Let our measurement vector be defined as $z_m = [V_m \quad \theta_m \quad \Delta h_m]^T$ with standard deviations σ_V , σ_θ , $\sigma_{\Delta h}$ respectively.

The noise covariance matrices Q and R can then be defined as follows:

$$Q = \begin{bmatrix} \sigma_{Ax}^2 & 0 & 0 \\ 0 & \sigma_{Az}^2 & 0 \\ 0 & 0 & \sigma_q^2 \end{bmatrix}, \qquad R = \begin{bmatrix} \sigma_V^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_{\Delta h}^2 \end{bmatrix}$$

Sensor Fusion

Example 3.10: Sensor fusion with bias estimation

The simplified equations of motion with sensor biases and noise are:

$$\dot{u} = (A_x - \lambda_x - w_x) - g\sin\theta - (q - \lambda_q - w_q)w$$

$$\dot{w} = (A_z - \lambda_z - w_z) + g\cos\theta + (q - \lambda_q - w_q)u$$

$$\dot{\theta} = (q - \lambda_q - w_q)$$

$$\dot{z}_E = -u\sin\theta + w\cos\theta$$

Now we will also attempt to estimate the accelerometer and gyro biases. For this, we augment the state vector $\underline{x} = [V_x \quad V_z \quad \theta \quad z_E]^T$ with the biases, creating an **augmented state vector**:

$$\underline{x}_{aug} = [V_x \quad V_z \quad \theta \quad z_E \quad \lambda_x \quad \lambda_z \quad \lambda_q]^T$$

The observation vector is: $z_m = [V_m \quad \theta_m \quad \Delta h_m]^T$

The input vectors is: $u_m = [A_{xm} \quad A_{zm} \quad q_m]^T$

$\Rightarrow$ unchanged!

Sensor Fusion

Example 3.10: Sensor fusion: State transition equation

The **augmented** state transition equation $\dot{\underline{x}} = f(\underline{x}, \underline{u}) + Gw$ is:

$$\begin{bmatrix} \dot{V}_x \\ \dot{V}_z \\ \dot{\theta} \\ \dot{z}_E \\ \dot{\lambda}_x \\ \dot{\lambda}_z \\ \dot{\lambda}_q \end{bmatrix} = \begin{bmatrix} (A_x - \lambda_x) - g\sin\theta - (q - \lambda_q)V_z \\ (A_z - \lambda_z) + g\cos\theta + (q - \lambda_q)V_x \\ q - \lambda_q \\ -V_x \sin\theta + V_z \cos\theta \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & V_z \\ 0 & -1 & -V_x \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_z \\ w_q \end{bmatrix}$$

Sensor Fusion

Example 3.10: Sensor fusion: Jacobian of state transition equation

The Jacobian of the state transition equation is:

$$F_x(\bullet) = \frac{\partial}{\partial \underline{x}} f\left(\underline{x}(t), \underline{u}(t), t\right)$$

$$= \begin{bmatrix} 0 & -q+\lambda_q & -g\cos\theta & 0 & -1 & 0 & V_z \\ q-\lambda_q & 0 & -g\sin\theta & 0 & 0 & -1 & -V_x \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ -\sin\theta & \cos\theta & -V_x\cos\theta - V_z\sin\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sensor Fusion

Example 3.10: Sensor fusion: State transition equation

The **augmented** state observation equation $\underline{z}_m = h(\underline{x}, \underline{u}) + \underline{v}$ is the same as for the non-augmented system:

$$\begin{bmatrix} V_m \\ \theta_m \\ \Delta h_m \end{bmatrix} = \begin{bmatrix} \sqrt{V_x^2 + V_z^2} \\ \theta \\ -z_E \end{bmatrix} + \begin{bmatrix} v_V \\ v_\theta \\ v_{\Delta h} \end{bmatrix}$$

Sensor Fusion

Example 3.10: Sensor fusion: Jacobian of observation equation

The Jacobian of the observation equation is:

$$H_x(\bullet) = \frac{\partial}{\partial \underline{x}} h\left(\underline{x}(t), \underline{u}(t), t\right)$$

$$= \begin{bmatrix} \frac{V_x}{\sqrt{V_x^2 + V_z^2}} & \frac{V_z}{\sqrt{V_x^2 + V_z^2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}$$

Sensor Fusion

Analysis of state observability

We have seen in a previous example that any given state vector such as:

$$\underline{x} = \begin{bmatrix} V_x & V_z & \theta & z_E \end{bmatrix}^T$$

Can be augmented with bias terms, which will then also be estimated during state estimation:

$$\underline{x}_{aug} = \begin{bmatrix} \underline{x} & \underline{x}_{bias} \end{bmatrix}$$

Which can for example result in:

$$\underline{x}_{aug} = \begin{bmatrix} V_x & V_z & \theta & z_E & \lambda_x & \lambda_z & \lambda_q \end{bmatrix}^T$$

Sensor Fusion

Analysis of state observability

Now we must ask ourselves, can $\underline{x}_{aug}$ actually be reconstructed given the sensor measurements that we have available in z_m?

In general, the question that we are asking is:
Given the system description, is the state $\underline{x}$ actually **observable**?

To answer this question, we must check the rank of the **observability matrix** indicated by O.

$\text{rank } O = n$ $\Longrightarrow$ State is observable

$\text{rank } O < n$ $\Longrightarrow$ State is not observable

Sensor Fusion

Analysis of state observability: Linear Systems

The observability matrix is calculated differently for linear and nonlinear systems.

For linear systems we get:

$$O = \begin{bmatrix} H_{k+1,k} \\ H_{k+1,k}\Phi_{k+1,k} \\ H_{k+1,k}\Phi_{k+1,k}^{2} \\ \vdots \\ H_{k+1,k}\Phi_{k+1,k}^{n-1} \end{bmatrix}$$

With $H_{k+1,k}$ the observation matrix and with $\Phi_{k+1,k}$ the discretized state transition matrix.

Sensor Fusion

Analysis of state observability: Nonlinear Systems

The observability matrix for nonlinear systems is:

$$O = \begin{bmatrix} \partial_x h \\ \partial_x (L_f h) \\ \partial_x (L_f L_f h) \\ \vdots \\ \partial_x (\underbrace{L_f \cdots L_f}_{n-1} h) \end{bmatrix}$$

With n the number of states, h the observation equation and with $L_f h$ **Lie derivative** of h which is defined as:

$$\begin{aligned} L_f h &= \partial_x h \cdot f \\ L_f L_f h &= \partial_x (L_f h) \cdot f \\ L_f L_f L_f h &= \partial_x (L_f L_f h) \cdot f \\ &\vdots \end{aligned}$$

Sensor Fusion

Analysis of state observability: Nonlinear Systems

In general, the Lie derivative L_f of a function h along the vector field f, is defined as follows:

$$L_f h = \partial_x h \cdot f = H_x \cdot f$$

The sequence of Lie derivatives is:

$$\begin{aligned}
&L_f h = \partial_x \left(h \cdot f \right) \\
&L_f L_f h = \partial_x \left(L_f h \cdot f \right) = \partial_x \left(\partial_x (h \cdot f) \cdot f \right) \\
&L_f L_f L_f h = \partial_x \left(L_f L_f h \cdot f \right) = \partial_x \left(\partial_x \left(\partial_x (h \cdot f) \cdot f \right) \right) \cdot f \\
&\qquad\qquad \vdots
\end{aligned}$$

Sensor Fusion

Analysis of state observability

- If the state is not observable, the Kalman filter will **not converge**.
- In order to make the state observable, we must then either include more sensor sources, and/or increase the density of the observation equations.
- Another option is to limit our state reconstruction to the observable subspace of the system, which means that we can only reconstruct only the part of $\underline{x}$ that is observable.

Sensor Fusion

Example 3.11: Analysis of state observability

We will investigate whether the state from **Example 3.7** is observable.

In this case, we have the following state transition and observation functions:

$$f(\underline{x}) = \begin{bmatrix} (A_x - \lambda_x) - g\sin\theta - (q - \lambda_q)V_z \\ (A_z - \lambda_z) + g\cos\theta + (q - \lambda_q)V_x \\ (q - \lambda_q) \\ -V_x \sin\theta + V_z \cos\theta \end{bmatrix}, \qquad h(\underline{x}) = \begin{bmatrix} \sqrt{V_x^2 + V_z^2} \\ \theta \\ -z_E \end{bmatrix}$$

Where we are trying to reconstruct the state given by:

$$\underline{x} = \begin{bmatrix} V_x & V_z & \theta & z_E \end{bmatrix}^T$$

Sensor Fusion

Example 3.11: Analysis of state observability

The state observation matrix is:

$$O = \begin{bmatrix} \partial_x h \\ \partial_x (L_f h) \\ \partial_x (L_f L_f h) \end{bmatrix} = \begin{bmatrix} \partial_x h \\ \partial_x (\partial_x h \cdot f) \\ \partial_x \left(\partial_x (h \cdot f) \cdot f \right) \end{bmatrix}$$

Matlab DEMO

Conclusion: the state $\underline{x} = \begin{bmatrix} V_x & V_z & \theta & z_E \end{bmatrix}^T$ **is observable!**

Sensor Fusion

Example 3.12: Analysis of state observability

We will investigate whether the augmented state from **Example 3.9** is observable.

In this case, we have the following state transition and observation functions:

$$f(\underline{x}) = \begin{bmatrix} (A_x - \lambda_x) - g\sin\theta - (q - \lambda_q)V_z \\ (A_z - \lambda_z) + g\cos\theta + (q - \lambda_q)V_x \\ q - \lambda_q \\ -V_x \sin\theta + V_z \cos\theta \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad h(\underline{x}) = \begin{bmatrix} \sqrt{V_x^2 + V_z^2} \\ \theta \\ -z_E \end{bmatrix}$$

Where we are trying to reconstruct the augmented state given by:

$$\underline{x}_{aug} = \begin{bmatrix} V_x & V_z & \theta & z_E & \lambda_x & \lambda_z & \lambda_q \end{bmatrix}^T$$

Sensor Fusion

Example 3.12: Analysis of state observability

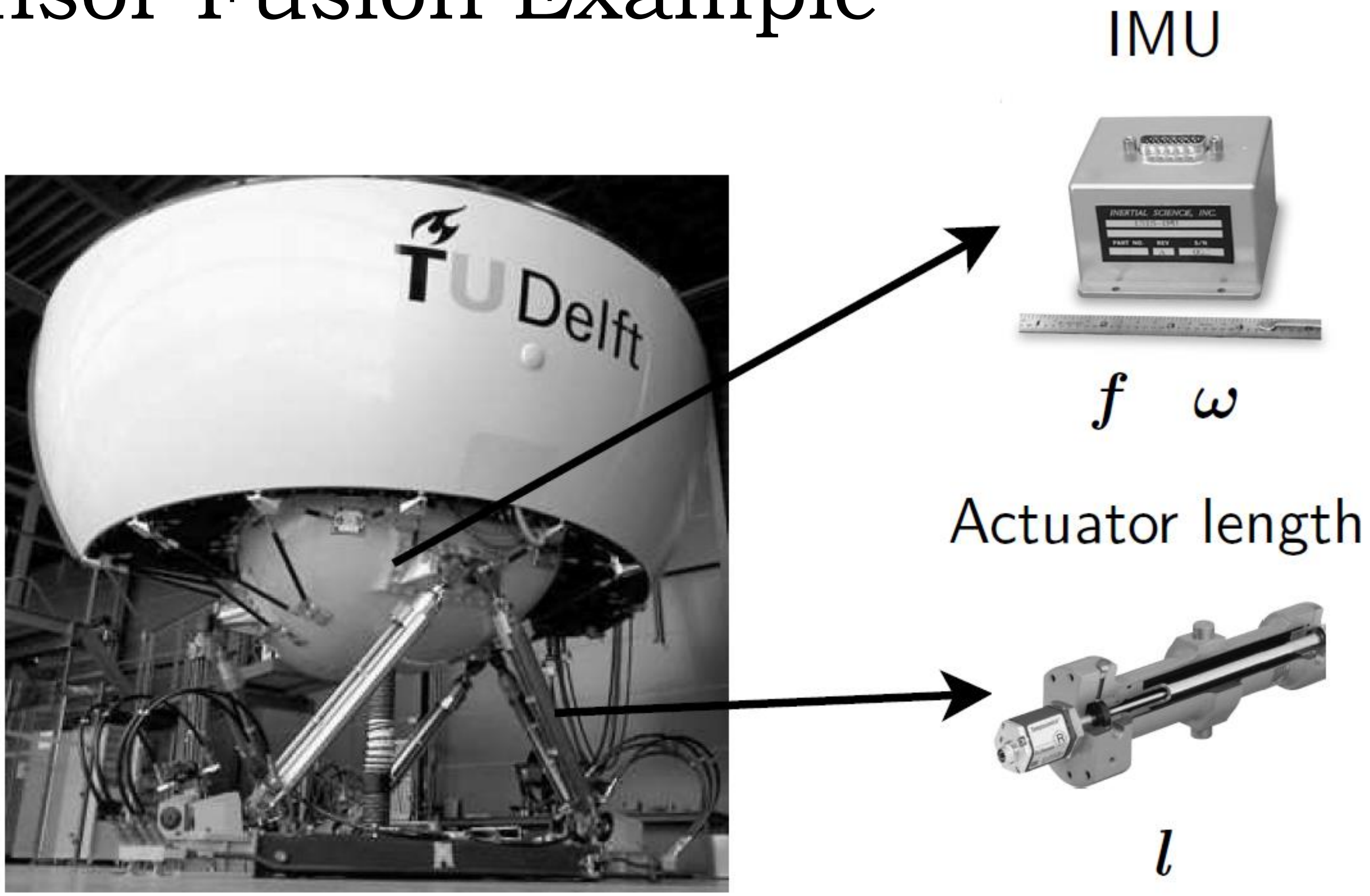
The state observation matrix is:

$$O = \begin{bmatrix} \partial_x h \\ \partial_x (L_f h) \\ \partial_x (L_f L_f h) \\ \partial_x (L_f L_f L_f h) \\ \partial_x (L_f L_f L_f L_f h) \\ \partial_x (L_f L_f L_f L_f L_f h) \end{bmatrix}$$

Matlab DEMO

Conclusion: the state $\underline{x}_{aug} = \begin{bmatrix} V_x & V_z & \theta & z_E & \lambda_x & \lambda_z & \lambda_q \end{bmatrix}^T$ **is observable!**

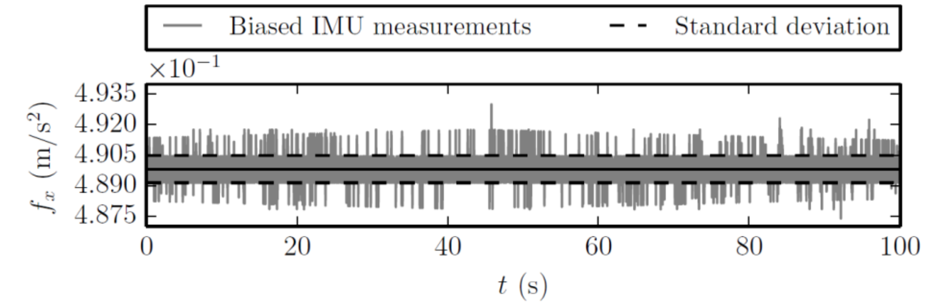
Sensor Fusion Example

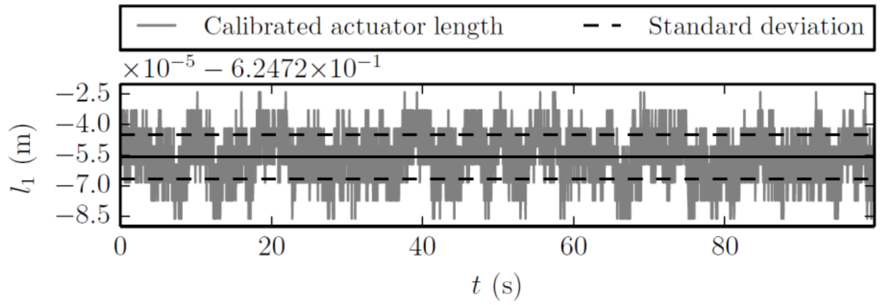


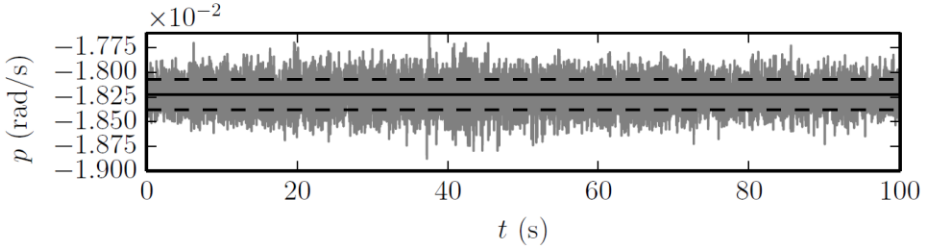
Miletović et al., *Improved Stewart platform state estimation using inertial and actuator position measurements* (2017), in: Control Engineering Practice, 62(102-115).

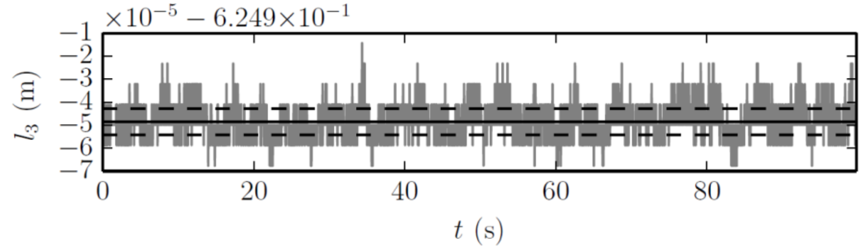
Sensor Fusion Example

Static measurements to determine measurement noise levels:





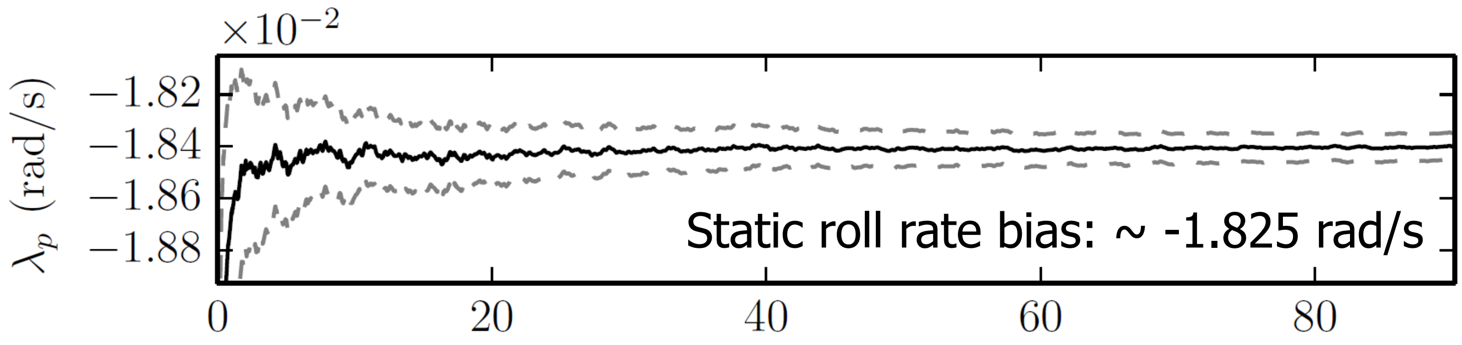




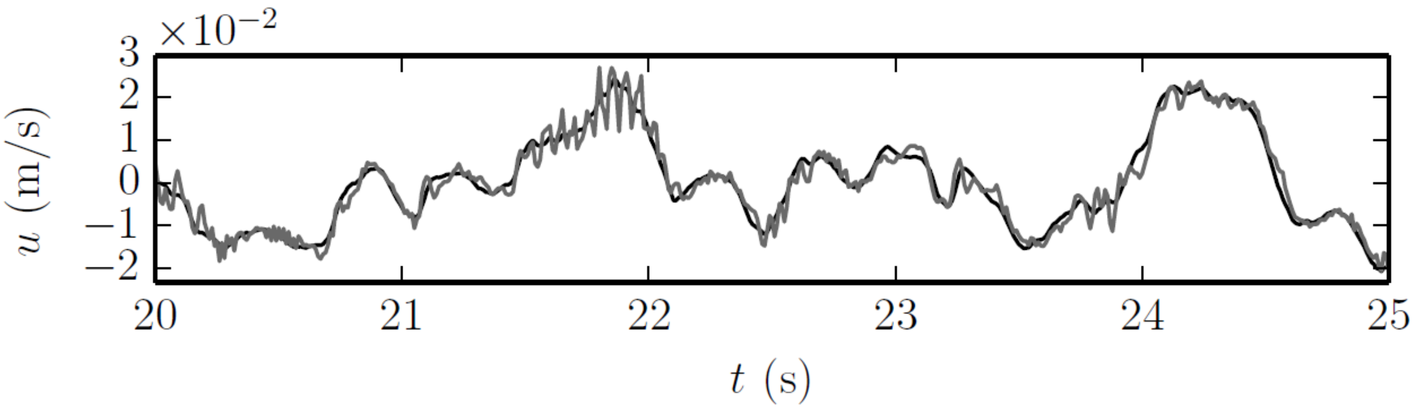
Static roll rate bias: ~ -1.825 rad/s

Sensor Fusion Example

Augmented roll rate bias estimate from Kalman filter:



Improvement of state estimates with Kalman filter:



Goals of this Lecture

Questions that were answered during this lecture:

1. Q1: What is state estimation, and why do we need it?
    - ➢ State estimation aims to obtain <u>accurate estimates of the real system states</u> from biased and noisy sensor measurements.
    - ➢ State estimation is necessary because real life systems are not only subject to <u>sensor noise</u>, but also to <u>process noise</u>. If the sensor measurements are used directly to identify a model, this model will be biased, and in general be of low quality.

2. Q2: What is the relationship between state estimation and sensor fusion?
    - ➢ Some outputs or states <u>may not be measurable directly</u>, like for example the true angle of attack. In that case, the state estimator allows us to <u>combine (fuse) sensors</u> of different types into a single <u>virtual sensor</u> that is able to measure the requested output or state.

Goals of this Lecture

Questions that were answered during this lecture:

3. Q3: What is a Kalman filter?
   - A Kalman filter is a recursive stochastic filter that corrects a predicted state with the weighted error between the real measurement and the predicted measurement.

4. Q4: What different types of Kalman filters are there and when do we use which type?
   - There are many different types of Kalman filters. For example, the ordinary KF, the Extended KF, the Iterated Extended KF, the Unscented KF, etc. The ordinary (linear) Kalman filter can only be used with linear systems. Most other Kalman filter types are aimed at state estimation for nonlinear systems.

Goals of this Lecture

Questions that were answered during this lecture:

5. Q5: How do we create a Kalman filter state estimator for a given system?
   - It is assumed that we have full knowledge on the state transition equation $f(\underline{x}, \underline{u})$, the output equation $h(\underline{x}, \underline{u})$, the noise input matrix G, and the covariance matrices of the process and output noise, Q and R, respectively. The Kalman filter is then constructed by reformulating a weighted least squares optimization problem into a recursive form.
6. Q6: What are the limitations of Kalman filter state estimation?
   - We need full knowledge of the system, although in most cases this knowledge is limited to knowledge of the system kinematics. The ordinary KF is an optimal filter, it will always converge to the optimal state estimation, but only for linear systems. The nonlinear KF techniques (i.e. EKF, IEKF, UKF) are not optimal filters, and are not guaranteed to converge to the optimal estimated state.

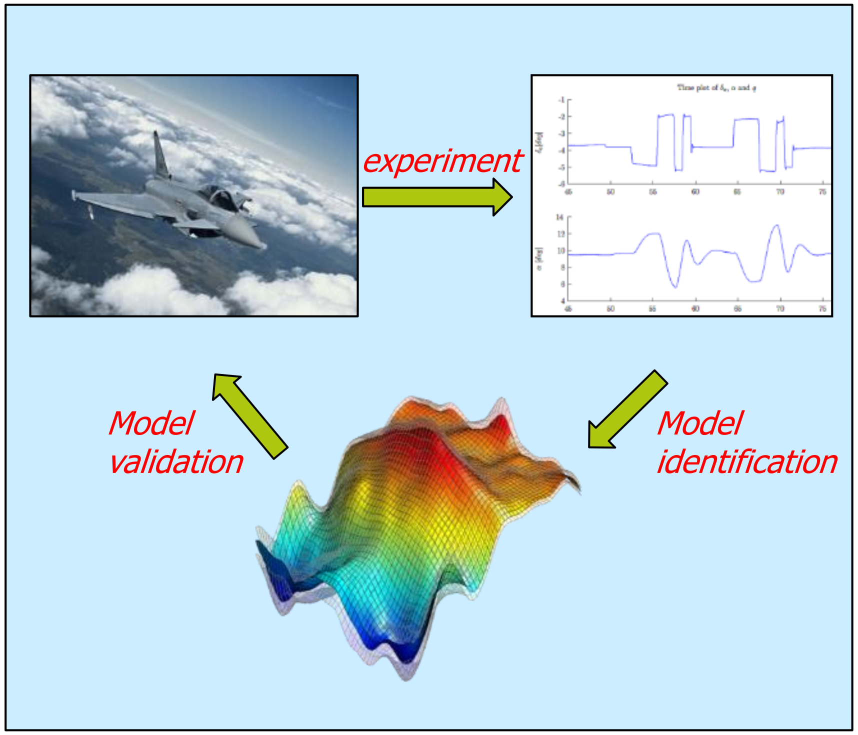
SysID High Level Overview

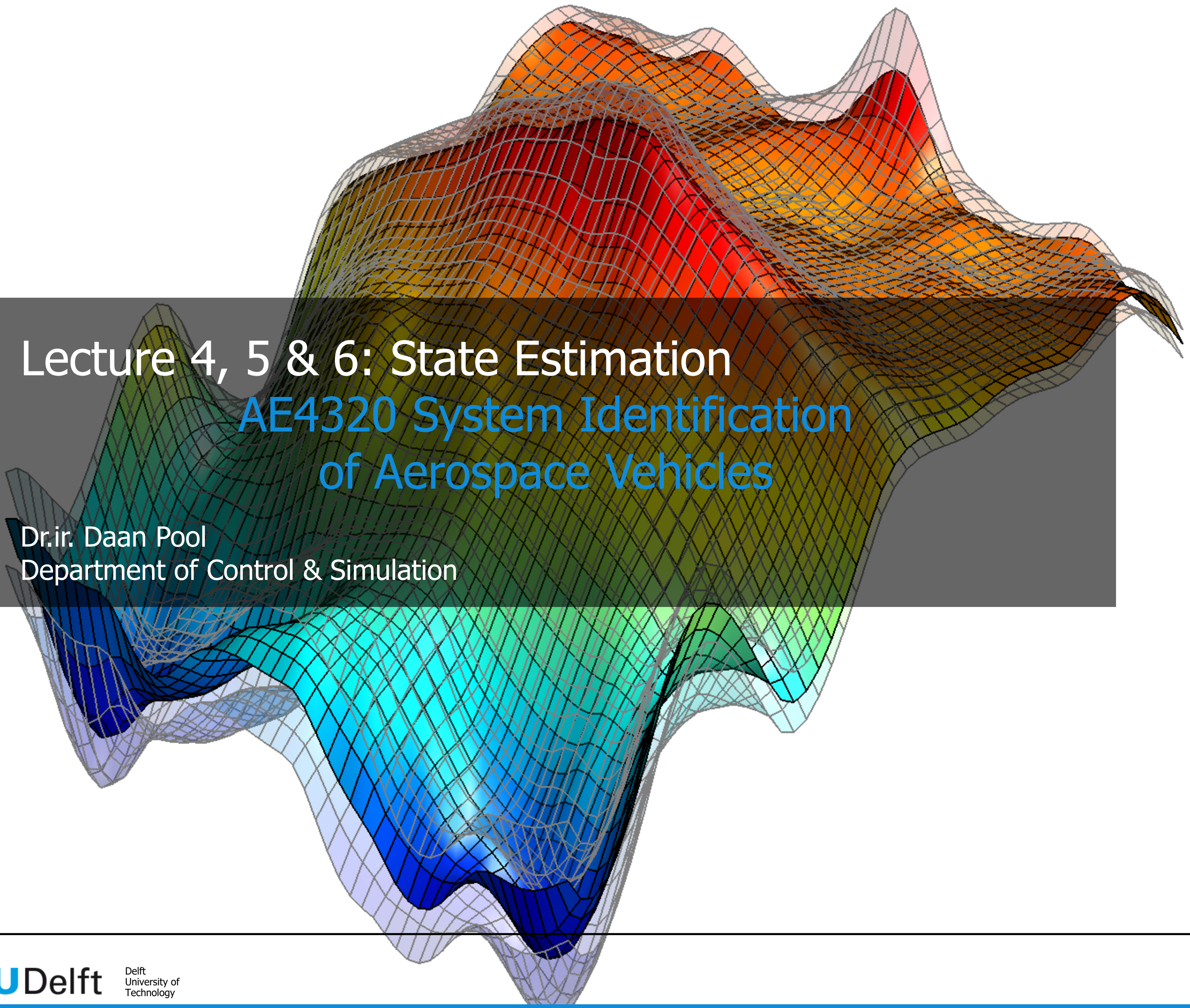
Where we are now in the System Identification Cycle:

Experiment phase
- Plant analysis
- Experiment design and execution
- Data logging and pre-processing

Model identification phase
- State estimation
- Model structure definition
- Parameter estimation

Model validation phase
- Model validation





Lecture 4, 5 & 6: State Estimation

AE4320 System Identification of Aerospace Vehicles

Dr.ir. Daan Pool
Department of Control & Simulation

TUDelft Delft University of Technology