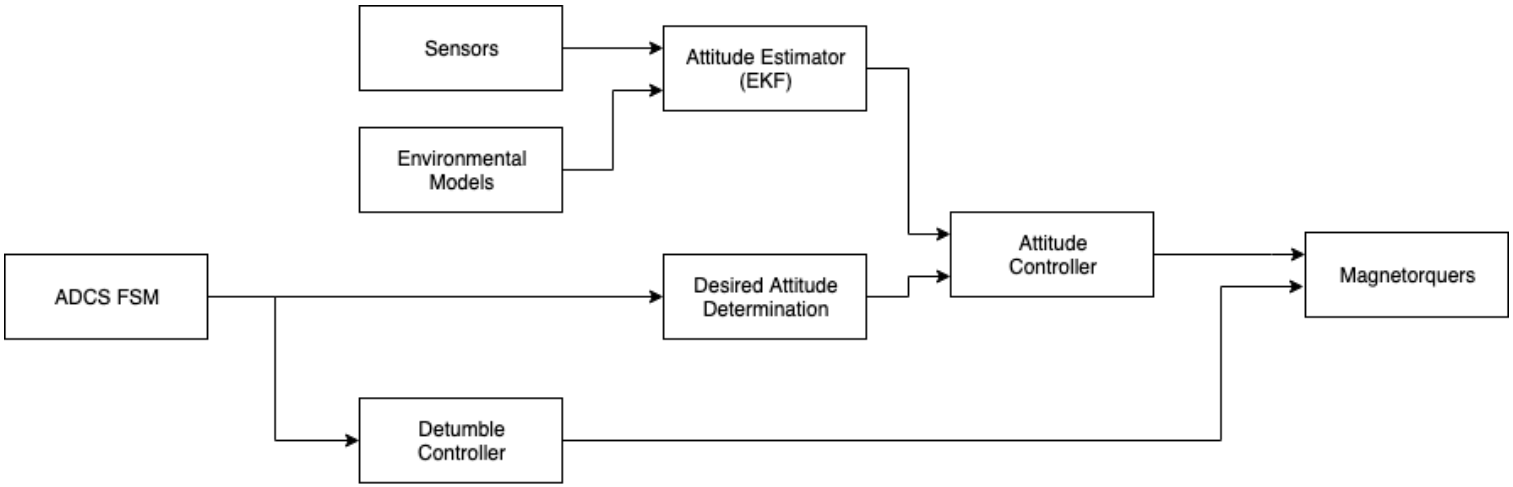
Introduction to the Error-state Kalman filter

Martin Brandt

January 20, 2020

Overview

Motivation



* Note that all submodules of the system will communicate with the FSM. Only the most explicit connections are drawn here.

The controller needs to know the attitude in order to control it, but there is no way to measure it directly $\rightarrow$ we have to estimate it!

Warning

These slides summarize A LOT of stuff, so you will probably not understand everything. But hopefully you will understand somewhat what the Kalman filter is and the reasoning behind the steps to why we use the error-state Kalman filter. This should also serve as an introduction and reference for new adcs people that wants to get into the ESKF.

State space models

Want to represent an arbitrary system of differential equations in vector form. In general: $\dot{x} = f(x, u)$

Continous LTI state space model

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1}$$

Discrete LTI state space model

$$\begin{aligned} x[k+1] &= Ax[k] + Bu[k] \\ y[k] &= Cx[k] + Du[k] \end{aligned} \tag{2}$$

Mass-spring-damper example

How you are used to seeing it

$$m\ddot{x} + d\dot{x} + ky = u \tag{3}$$

State space representation

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \tag{4}$$

Luenberger observer

Let's say we have a state space model of our system. How would we try to estimate the states of the system?

The logical first try (open-loop observer)

$$\dot{\hat{x}} = A\hat{x} + Bu \tag{5}$$

But because of modeling uncertainty our estimate will quickly diverge from the real value $\rightarrow$ include a correction term based on measurements (closing the loop) $\rightarrow$ Luenberger observer

The Luenberger observer

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y}), \quad \hat{y} = C\hat{x} \tag{6}$$

But how do we decide the gain L?

The Kalman filter

Let us first assume that our process model and measurement model includes **normally distributed** noise:

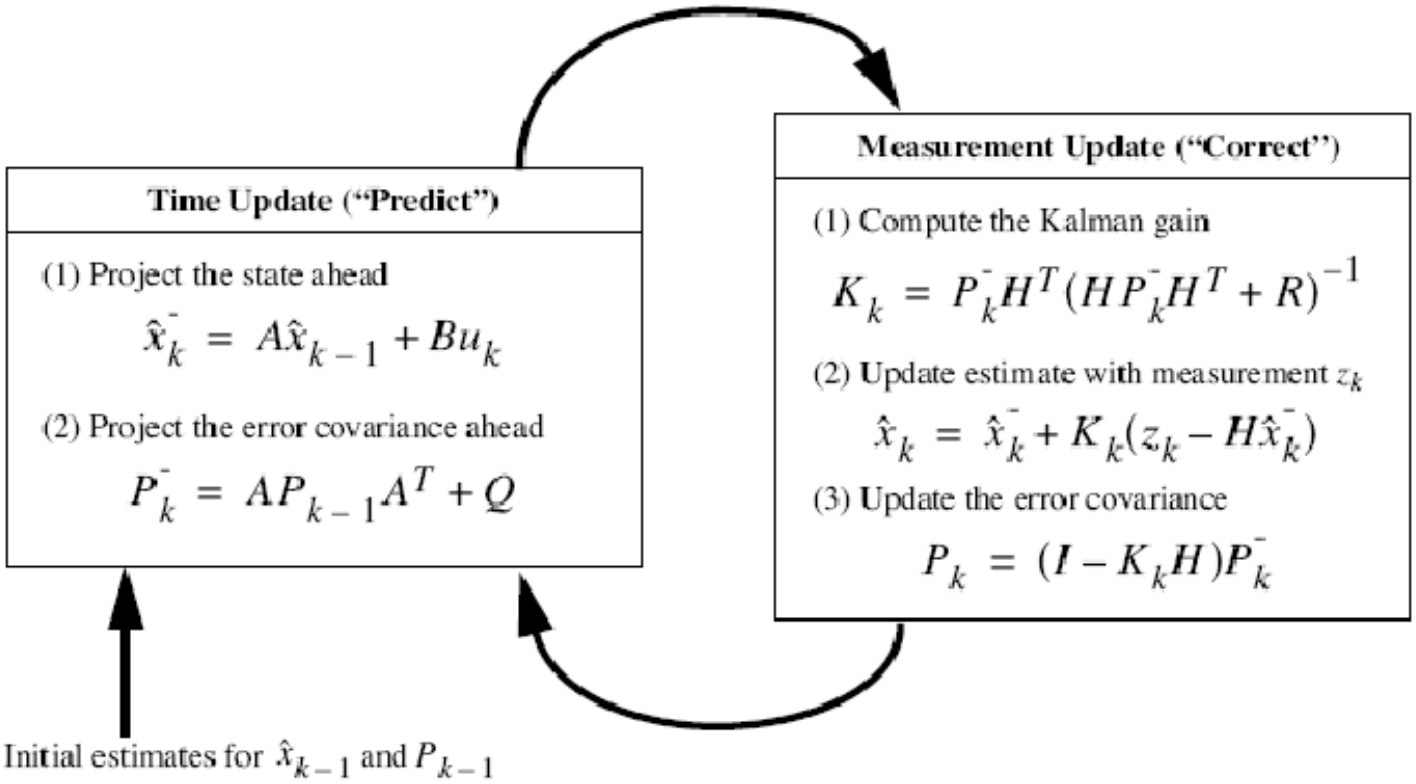
Stochastic LTI system

$$\begin{aligned} \dot{x} &= Ax + Bu + w \\ y &= Cx + Du + v \end{aligned} \tag{7}$$

The Kalman Filter is the optimal Luenberger observer for this system, in the sense that it minimizes **mean squared error**, i.e. $E\{(x - \hat{x})^2\}$.

The Kalman filter equations

For the discrete case (which is what you would implement on a microcontroller) the Kalman filter equations are:



The details are not important here, but note the predict + correct steps.

The satellite kinematics and kinetics

Let's try to apply the Kalman filter to our satellite:

Satellite kinematics

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} \tag{8}$$

Satellite kinetics

$$\dot{\omega} = J^{-1}\left[\mathbf{L} - \omega \times (J\omega)\right] \tag{9}$$

Problem: the system is highly nonlinear, so the Kalman filter cannot be directly applied (since it assumes a linear model).

Extended Kalman filter

The easiest solution to this problem would be to **linearize the nonlinear dynamics at each timestep** $\rightarrow$ Extended Kalman filter (EKF).

Problem: modeling uncertainty - the kinetics require that we know the inertia matrix of the satellite and since the dynamics are highly nonlinear the EKF might diverge:(

$\rightarrow$ Drop the kinetics, only use the kinematics: $\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega}$.
We let ω be the "control input", which we measure with the IMU.

Error-state Kalman filter

Now we are getting closer to a somewhat usable algorithm, but the dynamics are still highly nonlinear, which means the EKF will behave poorly (easily diverge).

The solution is to investigate the **error quaternion** instead: $q = \delta q \otimes \hat{q}$

If we approximate $\delta q \approx \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix}$, and ignore some terms we get:

Error quaternion kinematics ish

$$\delta\dot{\theta} = -S\left(\omega_m\right)\delta\theta - n_r \tag{10}$$

This is linear, so we can discretize it and use the good old Kalman filter. This is much more stable than the EKF, very nice, very nice.

Error-state Kalman filter

Final problem: IMU measurements drift... $\rightarrow$ estimate the rate gyro bias b as well as the attitude quaternion q. We assume that the bias error follows the following dynamics (random walk):

Bias error dynamics

$$\dot{\delta b} = \dot{b} - \dot{\hat{b}} = n_w \tag{11}$$

The new error quaternion dynamics with bias are:

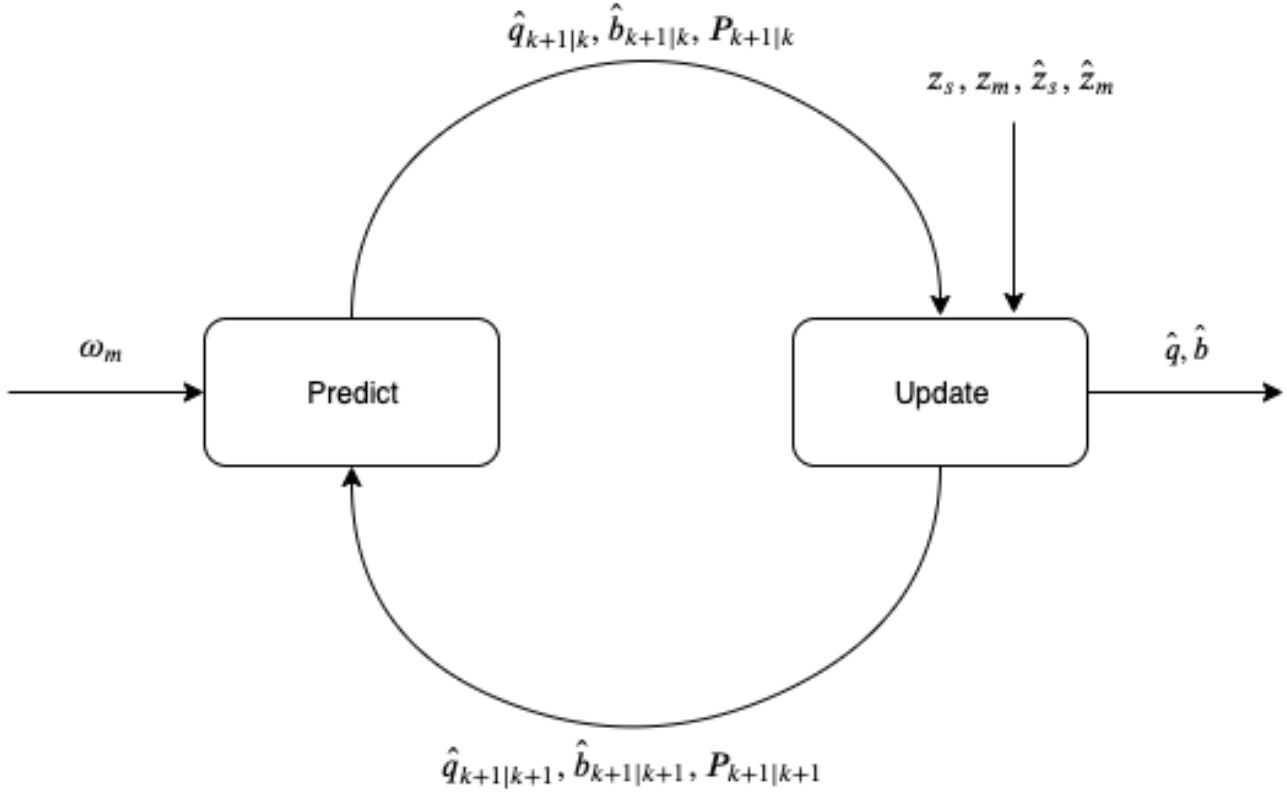
Error quaternion kinematics ish

$$\dot{\delta\theta} = -S\left(\omega_m - b\right)\delta\theta - \delta b - n_r \tag{12}$$

If we apply the KF equations to these equations we eventually (after a lot of math) arrive at the ESKF algorithm we currently use. I'm not going to bother writing all the resulting equations, you can look it up if you need to.

Error-state Kalman filter

While all this seems quite complicated, it is really just the regular Kalman filter with a few extra tricks (estimating the error instead of the true state, using gyro measurements as input and also estimating the bias). As a user of the ESKF this is really all you need to know:



ESKF code

Let's have a brief look at the code I guess?

Code: `https://git.orbitntnu.no/adcs/libeskf`
Tests: `https://git.orbitntnu.no/adcs/eskf_test`
Matlab tests: `https://git.orbitntnu.no/Martimos/adcs_simulation/tree/eskf-c-test`

Thank you for coming to my TEDx talk