Navigation and Intelligent Vehicles

Lab session 1: Introduction to the Kalman filter (KF)

In this session, you will learn
a) How to implement a Kalman filter in Matlab and use it for estimating a constant
b) About the influence of the parameters of the KF on its performance

State-space models

State-space models are essentially a notational convenience for estimation and control problems, developed to make tractable what would otherwise be a notational-intractable analysis. A general way of writing them is by using a **process model** and a **measurement model**. Together they serve as the basis for virtually all linear estimation methods.

$$x(k+1) = Ax(k) + Bu(k) + Gw(k) \tag{1}$$

$$z(k) = Cx(k) + Hv(k) \tag{2}$$

The index k is used to denote the time step. Eq. (1) is the *process model*, and represents the way a new state $x(k+1)$ is modeled as a linear combination of both the previous state $x(k)$, some known inputs $u(k)$, and some process noise $w(k)$.

Eq. (2) is the *measurement model* (also: observation model) and describes the way the observations $z(k)$ are derived from the internal state $x(k)$ and includes some observation noise $v(k)$.

The random noise variables $w(k)$ and $v(k)$ represent the process and observation noise respectively. They can be interpreted as the model uncertainty, or differently stated, as the plausible mismatch between the used model and reality.

Note that in general, $x(k)$, $z(k)$, $u(k)$, $w(k)$ and $v(k)$ are vectors, while A, B, C, G and H are matrices.

Kalman filter

Some theory

The Kalman filter is essentially a set of mathematical equations that implement a **predictor-corrector type estimator** that is optimal in the sense that it minimizes the estimated error covariance – when some presumed conditions[1] are met. The KF has been popular for quite many years, probably because of its relative simplicity and robust nature. Rarely do the conditions necessary for optimality actually exist, and yet the filter apparently works well for many applications in spite of this situation.

The KF addresses the general problem of estimating the state $x \in \Re^n$ of a discrete-time controlled process that is governed by Eq. (1), with a measurement $z \in \Re^m$ that is governed by Eq. (2). The noise terms $w(k)$ and $v(k)$ are assumed to be independent (of each other), white and with normal probability distributions $N(\mu_x, \sigma_x^2)$.

The $n \times n$ matrix A in the difference equation (1) relates the state at the current time step k to the state at the next time step $k+1$, in absence of either a driving function u or process noise w. The $n \times \ell$ matrix B relates the optional control input $u \in \Re^\ell$ to the state x. The $n \times p$ matrix G relates the process noise $p(w) \sim N(0,Q)$ to the state x.

[1] Main conditions to be met: white Gaussian noise sources, and a linear system that is observable and controllable. Going deeper into the conditions falls out of the scope of this lab session, refer to reference 1.

The $m \times n$ matrix C in the measurement equation (2) relates the state to the measurement z. The $m \times q$ matrix H relates the observation noise $p(v) \sim N(0,R)$ to the measurement z.

Using the KF

The KF estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the KF fall into two groups:

- The **time update equations**, Eq. (3)-(4), are responsible for projecting forward the current state and error covariance estimates (in time) to obtain the *a priori* estimates for the next time step.
- The **measurement update equations**, Eq. (5)-(7), are responsible for the feedback, i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations can also be thought of as *predictor* equations, while the measurement update equation can be thought of as *corrector* equations. Indeed, the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems, as shown in figure 1. An overview of the specific equations for the time and measurement updates can be found in table 1, and they are explained in the remainder of this section.

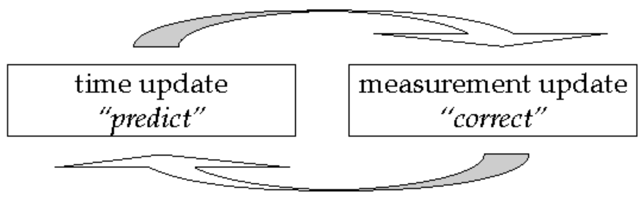


Figure 1: Overview of the Kalman Filter process

Table 1: the time and measurement update equations

Project the state ahead:		Compute the Kalman gain:	
$\hat{x}^-(k) = A\hat{x}(k-1) + Bu(k)$	(3)	$K(k) = P^-(k)C^T[CP^-(k)C^T + HRH^T]^{-1}$	(5)
Project the error covariance ahead:		Update estimate with measurements z_i:	
$P^-(k) = AP(k-1)A^T + GQG^T$	(4)	$\hat{x}(k) = \hat{x}^-(k) + K(k) \cdot [z(k) - C\hat{x}^-(k)]$	(6)
		Update the error covariance:	
		$P(k) = [\mathrm{I} - K(k)C] \cdot P^-(k)$	(7)

The difference $\tilde{z}(k) = z(k) - C\hat{x}^-(k)$ in (6) is called the *measurement innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement $C\hat{x}^-(k)$ and the actual measurement $z(k)$. A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix K in Eq. (5) is chosen to be the *gain* or *blending factor* that minimizes the *a posteriori* error covariance. When the measurement error covariance R approaches zero, the gain K weights the residual more heavily. In the limit this becomes $\lim_{R \to 0} K(k) = C^{-1}$.

On the other hand, as the *a priori* estimate error covariance $P^-(k)$ approaches zero, the gain K weights the residual less heavily. In the limit this becomes $\lim_{P^-(k) \to 0} K(k) = 0$.

Another way of thinking about the weighting (by K) is that as the measurement error covariance R approaches zero, the actual measurement $z(k)$ is "trusted" more and more, while the predicted measurement $C\hat{x}^-(k)$ is trusted less and less. The inverse is valid when the *a priori* estimate error covariance $P^-(k)$ approaches zero.

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used. This recursive nature is one of the very appealing features of the Kalman Filter – it makes practical implementations much more feasible than (for example) an implementation of a

Wiener filter which is designed to operate on *all* of the data *directly* for each estimate. The KF instead recursively conditions the current estimate on all of the past measurements.

Lab assignment: estimating a random constant

To help in developing a better feel for the operation and capability of the filter, you'll implement a very simple system in Matlab, and study the different parameters.

Putting the theory to practice

Process and measurement equations

We'll attempt to estimate a scalar constant, a voltage for example. Let's assume that we have the ability to take measurements of the constant $x \in \Re^1$, but that the measurements are corrupted by a 0.01 volt RMS white measurement noise (e.g. our analogue to digital converter is not very accurate). In this example, our process is governed by the linear difference equation:

$$x(k+1) = x(k) + w(k) \tag{8}$$

With a single direct measurement $z \in \Re^1$ that is

$$z(k) = x(k) + v(k) \tag{9}$$

When comparing back to Eq. (1) and (2), we can make the following observations: the state does not change from step to step (it's a constant!) so $A = 1$. There is no control input, so $u = 0$. Our noisy measurement is of the state directly, so $C = 1$.

About the noise

We will presume a very small process variance, since we are rather sure that a constant is being measured. So let $Q = 10^{-5}$. We could certainly let $Q = 0$, but we'll not set this here, since we want to study the influence of all the parameters. From the data generation, we know that R=*0.01*.

About the initial values

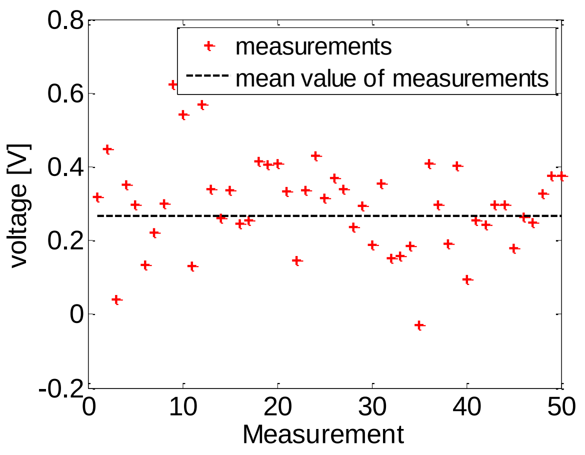
Let's assume that from experience we know that the true value of the random constant has a standard normal probability distribution. Further, we will "seed" the filter with the guess that the constant is 0. In other words, before starting, we let $\hat{x}_0$ =0.

Similarly an initial value for $P(k-1)$ needs to be chosen, call it P_0. If we were absolutely certain that our initial state estimate $\hat{x}_0 = 0$ was correct, we would let $P_0 = 0$, which would cause the filter initially and always (wrongly) to believe that $\hat{x}(k) = 0$. However, given the uncertainty in our initial estimate $\hat{x}_0$, we'll start our filter with $P_0 = 1$.

Programming task

Simulate measurement data

Choose a scalar constant, let's say $x = 0.26578$. Generate now 50 distinct measurements $z(k)$ that have an error normally distributed around zero with a standard deviation of 0.1. Save the data. The reason why we pre-generate the measurement data is that it will allow us to run several simulations with the same exact measurements.



Program the KF

This is completely up to you.

Analyse the KF

This is the main part of the lab session. In the next section some help is provided. Create a separate file for each analysis you make, so that you can redo them later – if needed.

Analysis task

Now that you have a working Kalman Filter, it's time to study the influence of the parameters. But first we'll see how we can evaluate the outcome of the filter.

Error sources

When the filter diverges, thus yielding unacceptably large state estimation errors, this is mostly due to one (or more) of the following: modelling errors, numerical errors, or programming errors.

In the other case, when the filter converges towards the solution, the question of what is acceptable for the estimation error remains. This is discussed in the next section.

Performance indicators

- **The Instant Squared Error (ISE).** The filter estimate $\hat{x}(k)$ is compared at each instant with the true value $x(i)$ by calculating the squared error:

$$ISE(k) = \left[x(k) - \hat{x}(k)\right]^2 \tag{10}$$

- **The Mean Squared Error (MSE).** The filter estimate $\hat{x}(k)$ is compared with the true value $x(i)$ by calculating the mean (=history taken into account) squared error:

$$MSE(i) = \frac{1}{i}\sum_{k=1}^{i}\left[x(k) - \hat{x}(k)\right]^2 \tag{11}$$

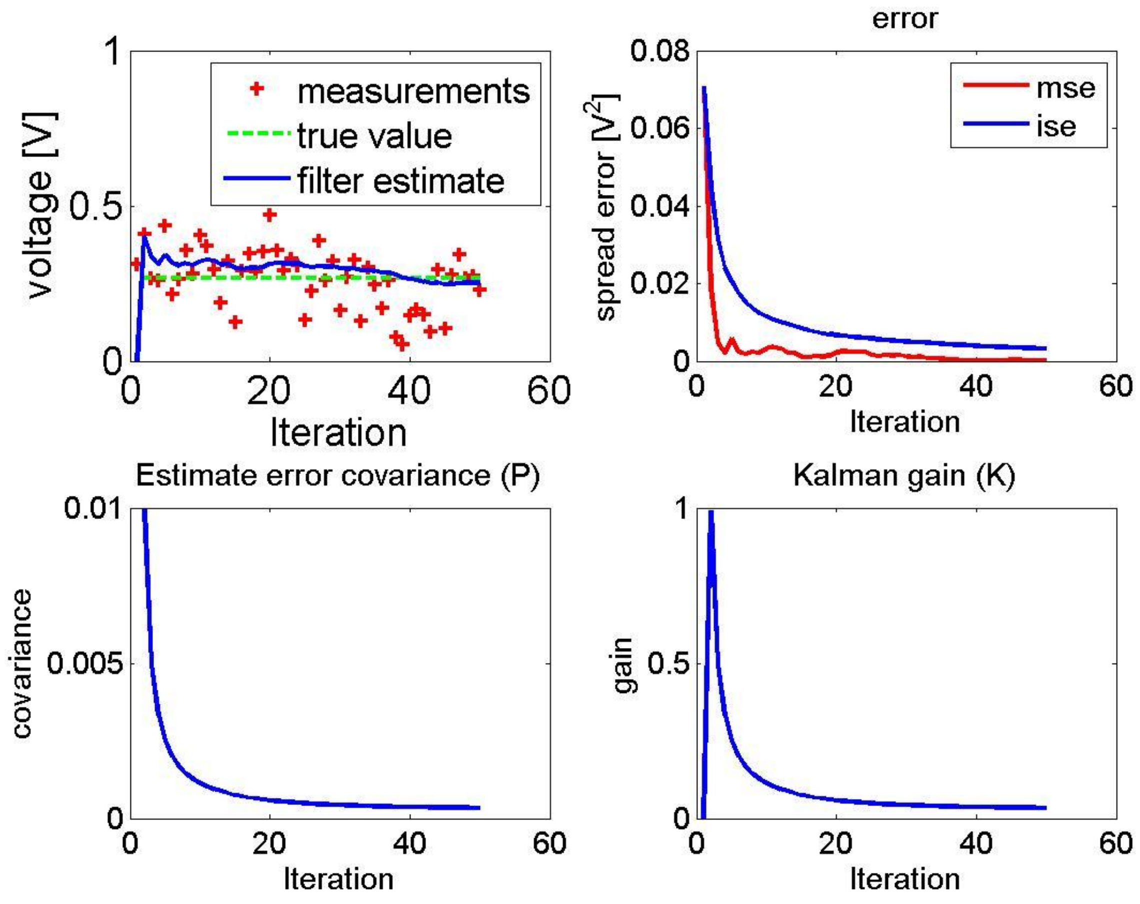
- The **estimated error covariance** P . This is an internal KF variable that reflects the quality of the estimate. The closer it gets to 0, the more the KF *trusts* its prediction.

Influence of the parameters

- Check the influence of the **measurement noise variance *R*** by increasing or decreasing it with a factor of 1000. When concluding, link your observations to the "speed" with which the filter "believes" the measurements.
- Do the same for the **process noise variance *Q***.
- What if the **data noise** is no longer zero-mean? How to solve this?
- What happens if you change the **initial value of the error covariance** P_0 ? Increase/decrease it. What is (are) its boundary value(s)? What are critical values for the correct functioning?
- Does the value of the **Kalman gain *K*** contain information?
- Are there any other parameters that influence your KF ?

Example of a result

You should be able to obtain the following information on your filter outcome/performance. Please note that not everything is shown on this graph and you'll not necessarily need all of these graphs to explain the requested issues. It is simply an illustration.



Report task

Make a <u>short</u> (maximum five pages) report in which you discuss (text) and illustrate (graphs) how a Kalman Filter works, and which parameters are important for its correct functioning. Show clearly that you understand the difference between the noise on the data, the process and observation noise. Discuss also how the parameters influence the Kalman filter. Send your MATLAB code by e-mail.
Deadline: next session.

References

- Yaakov Bar-Shalom, Xiao-Rong Li, "Estimation and Tracking, Principles, Techniques, and Software", Artech House Publishing 1993, ISBN 0-89006-643-4
- Greg Welch and Gary Bishop, "An Introduction to the Kalman Filter", SIGGRAPH 2001, August 12-17 2001.
- T. B. Schön and F. Lindsten, "Learning of dynamical systems - particle filters and Markov chain methods," 2017, Lecture notes, Appendix B. [Online]. Available: http://www.it.uu.se/research/systems_and_control/education/2017/smc/SMC2017.pdf