Intelligent distributed systems

Daniele Fontanelli

Department of Industrial Engineering
University of Trento
E-mail address: *daniele.fontanelli@unitn.it*

2022/2023

UNIVERSITÀ DI TRENTO | Dipartimento di Ingegneria Industriale

Outline

1. Linear Estimators
   - The Least Squares solution
   - The Kalman filter

2. Nonlinear estimators
   - The Non-linear Least Squares solution
   - The Extended Kalman filter

Outline

1. Linear Estimators
   - The Least Squares solution
   - The Kalman filter
2. Nonlinear estimators
   - The Non-linear Least Squares solution
   - The Extended Kalman filter

Outline

1. Linear Estimators
   - The Least Squares solution
   - The Kalman filter
2. Nonlinear estimators
   - The Non-linear Least Squares solution
   - The Extended Kalman filter

The Least Squares Algorithm

Generalisation to the multidimensional case

Let us consider

$$z = Hx + \varepsilon$$

$x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$, $H \in \mathbb{R}^{m \times n}$, with $m \geq n$ and H full rank, and let us consider the LS solution.
We have that

$$\begin{aligned} z = Hx \Rightarrow J(x) = (z - Hx)^T (z - Hx) \Rightarrow \\ \Rightarrow \hat{x}^{LS} = \arg\min_x J(x) = (H^T H)^{-1} H^T z, \end{aligned}$$

that is the solution of the sensor calibration we have seen previously.

The Least Squares Algorithm

Generalisation to the multidimensional case

Similarly, if we add a *weighting matrix* W (which has to be symmetric and p.d. in order to have the Hessian p.d.), we have

$$z = Hx \Rightarrow J(x) = (z - Hx)^T W (z - Hx) \Rightarrow$$
$$\Rightarrow \hat{x}^{LS} = \arg\min_x J(x) = (H^T W H)^{-1} H^T W z.$$

In other words, the *Weighted Least Squares* (WLS) is actually a BLUE for a generic pdf, while it is the MVUE for the Gaussian case, provided that the covariance matrix of ε is R and $W = R^{-1}$.

The Least Squares Algorithm

LS batch

We are now in a position to present the *LS batch* algorithm with noisy measures.
Consider the following problem:

- We have to determine the value of a *nonrandom* constant parameter;
- We collect the sensor readings from *multiple sensors* deployed in an environment by means of a communication system;
- The data are collected in *different time instants* $k = 0, \ldots$;
- To ensure a correct *sensor fusion*, the data are *timestamped* at the source location and all the sensors are supposed to be *synchronised* at the *Coordinated Universal Time* (UTC), for example using GPS signals;

The Least Squares Algorithm

LS batch

Moreover:

- The set of sensor readings is not necessarily the same and, hence, *may change in time*;
- Each sensor provides the measures with a different (possibly time varying) *uncertainty*, i.e., the stochastic process affecting the sensors is *non-stationary*;
- The data are collected by a single entity fusing all the data, i.e., *centralised approach*.

The Least Squares Algorithm

LS batch

Believe it or not, you know perfectly how to solve this problem:

- First, the synchronisation of the nodes allow us to assume that all the sensor readings are not affected by the *network induced problems*, e.g., jitter, latency, etc.;
- Second, since the quantity to estimate is *nonrandom* and *constant*, a *non Bayesian* approach should be used;
- Since no information is given about the pdf of the noise, it is impractical to use the ML, and hence we go for the LS approach, performed in a centralised way.

The Least Squares Algorithm

LS batch

To properly model this problem, let us define

$$z(i) = H(i)x + \varepsilon(i), \quad i = 1, \dots, k$$

the time varying set of measurements collected at time i.
Then, let us define the following aggregating vectors

$$Z^k = \begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(k) \end{bmatrix}, \quad H^k = \begin{bmatrix} H(1) \\ H(2) \\ \vdots \\ H(k) \end{bmatrix}, \quad \varepsilon^k = \begin{bmatrix} \varepsilon(1) \\ \varepsilon(2) \\ \vdots \\ \varepsilon(k) \end{bmatrix}.$$

The Least Squares Algorithm

LS batch

For the covariance matrix of the noises

$$C^k = \begin{bmatrix} \mathsf{C}\{\varepsilon(1)\} & 0 & \dots & 0 \\ 0 & \mathsf{C}\{\varepsilon(2)\} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathsf{C}\{\varepsilon(k)\} \end{bmatrix}$$

Notice how the noise is assumed uncorrelated in different time instants, while instead it can be correlated for each time instant ($\mathsf{C}\{\varepsilon(i)\}$ is not necessarily diagonal).

The Least Squares Algorithm

LS batch

Following the same idea presented previously, we have

$$
\begin{aligned}
J(x,k) &= \sum_{i=1}^{k} (z(i) - H(i)x)^T \mathsf{C}\,\{\varepsilon(i)\}^{-1} (z(i) - H(i)x) = \\
&= (Z^k - H^k x)^T C^{k^{-1}} (Z^k - H^k x) \Rightarrow \\
&\Rightarrow \hat{x}^{LS} = \arg\min_x J(x,k) = ({H^k}^T {C^k}^{-1} H^k)^{-1} {H^k}^T {C^k}^{-1} Z^k.
\end{aligned}
$$

Notice how the solution has the same properties of the scalar case.

The Least Squares Algorithm

LS and ML

If the sequence of the $\varepsilon(i)$ comprise noises that are zero-mean, normally distributed, the uncorrelatedness becomes *independence*.
Therefore

$$p(\varepsilon(i)) = \frac{1}{\sqrt{|2\pi \mathsf{C}\{\varepsilon(i)\}|}} e^{-\frac{1}{2}\varepsilon(i)^T \mathsf{C}\{\varepsilon(i)\}^{-1}\varepsilon(i)}.$$

As a consequence, the *likelihood function*

$$p(Z^k|x) = \prod_{i=1}^{k} p(z(i)|x) = \prod_{i=1}^{k} p(H(i)x + \varepsilon(i)|x) = \prod_{i=1}^{k} \mathcal{N}(H(i)x, \mathsf{C}\{\varepsilon(i)\})$$

Therefore,

$$\begin{aligned} p(H(i)x + \varepsilon(i)|x) &= ce^{-\frac{1}{2}(H(i)x+\varepsilon(i)-H(i)x)^T \mathsf{C}\{\varepsilon(i)\}^{-1}(H(i)x+\varepsilon(i)-H(i)x)} = \\ &= cp(\varepsilon(i)|x) \end{aligned}$$

The Least Squares Algorithm

LS and ML

Hence, the *likelihood function* becomes

$$p(Z^k|x) = \prod_{i=1}^{k} p(z(i)|x) = \prod_{i=1}^{k} p(H(i)x + \varepsilon(i)|x) = \prod_{i=1}^{k} p(\varepsilon(i)|x) =$$

$$= \prod_{i=1}^{k} \frac{1}{\sqrt{|2\pi \mathsf{C}\{\varepsilon(i)\}|}} \prod_{i=1}^{k} e^{-\frac{1}{2}\varepsilon(i)^T \mathsf{C}\{\varepsilon(i)\}^{-1}\varepsilon(i)} =$$

$$= ce^{-\frac{1}{2}\sum_{i=1}^{k} \varepsilon(i)^T \mathsf{C}\{\varepsilon(i)\}^{-1}\varepsilon(i)}$$

The Least Squares Algorithm

LS and ML

Since

$$z(i) = H(i)x + \varepsilon(i) \Rightarrow \varepsilon(i) = z(i) - H(i)x,$$

we have

$$\begin{aligned} p(Z^k|x) &= ce^{-\frac{1}{2}\sum_{i=1}^{k} \varepsilon(i)^T \mathsf{C}\{\varepsilon(i)\}^{-1}\varepsilon(i)} = \\ &= ce^{-\frac{1}{2}\sum_{i=1}^{k} (z(i)-H(i)x)^T \mathsf{C}\{\varepsilon(i)\}^{-1}(z(i)-H(i)x)} = \\ &= ce^{-\frac{1}{2}(Z^k-H^kx)^T \mathsf{C}\left\{\varepsilon(i)^k\right\}^{-1}(Z^k-H^kx)}, \end{aligned}$$

which has its maximum wherever the exponent is minimum.
Again, if the measurement noise is normally distributed, zero-mean and white, the LS is a "disguised" version of the ML.

The Least Squares Algorithm

Unbiasedness

If the noises $\varepsilon(i)$ are uncorrelated and zero-mean (without *any other assumption on the noise stochastic process description*), the LS is *unbiased*.

Indeed:

$$\mathsf{E}\left\{\hat{x}(k)^{LS}\right\} = (H^{k^T} C^{k^{-1}} H^k)^{-1} H^{k^T} C^{k^{-1}} \mathsf{E}\left\{Z^k\right\} = L^k \mathsf{E}\left\{Z^k\right\} =$$

$$L^k \mathsf{E}\left\{H^k x + \varepsilon^k\right\} = L^k H^k x = x$$

The Least Squares Algorithm

Estimation error

The *estimation error* is instead given by:

$$\tilde{x} = x - \hat{x}^{LS} = x - L^k Z^k = x - L^k(H^k x + \varepsilon^k) = -L^k \varepsilon^k.$$

The *uncertainty* of the estimation is instead given by the *covariance matrix* $P(k)$ of the estimates:

$$\begin{aligned} P(k) &= \mathsf{E}\left\{(\hat{x}(k) - \mathsf{E}\left\{\hat{x}(k)\right\})(\hat{x}(k) - \mathsf{E}\left\{\hat{x}(k)\right\})^T\right\} = \\ &= \mathsf{E}\left\{(\hat{x}(k) - x)(\hat{x}(k) - x)^T\right\} = \\ &= \mathsf{E}\left\{\tilde{x}(k)\tilde{x}(k)^T\right\} = \mathsf{E}\left\{-L^k \varepsilon^k (-L^k \varepsilon^k)^T\right\} = \\ &= L^k C^k L^{k^T} = \left(H^{k^T} C^{k^{-1}} H^k\right)^{-1}. \end{aligned}$$

The Least Squares Algorithm

Estimation error

As a consequence:

$$\hat{x}^{LS} = ({H^k}^T {C^k}^{-1} H^k)^{-1} {H^k}^T {C^k}^{-1} Z^k = P(k) {H^k}^T {C^k}^{-1} Z^k.$$

The Least Squares Algorithm

Recursive solution

The main problem of the LS solution presented till now is that it is *batch*: all the data collected up to k are used simultaneously in the same instant. As time goes by, this solution becomes easily unpractical.
However, is it possible to derive a *recursive implementation* of the LS.
In practice, each time that a new set of measurements are available at time $k+1$, we make use of the results of the estimates up to time k. This way, we have a remarkable saving in the computation time.

The Least Squares Algorithm

Recursive solution

To this end, let us define

$$Z^{k+1} = \begin{bmatrix} Z^k \\ z(k+1) \end{bmatrix}, \quad H^{k+1} = \begin{bmatrix} H^k \\ H(k+1) \end{bmatrix}, \quad \varepsilon^{k+1} = \begin{bmatrix} \varepsilon^k \\ \varepsilon(k+1) \end{bmatrix},$$

and

$$C^{k+1} = \begin{bmatrix} C^k & 0 \\ 0 & \mathsf{C}\{\varepsilon(k+1)\} \end{bmatrix}.$$

The Least Squares Algorithm

Recursive solution

Let us start with the definition of the estimator *covariance matrix* given previously. In particular, since it involves an inverse, we are interested in the *inverse of the covariance matrix*:

$$
\begin{aligned}
P(k+1)^{-1} &= H^{k+1^T} C^{k+1^{-1}} H^{k+1} = \\
&= H^{k^T} C^{k^{-1}} H^k + H(k+1)^T \mathsf{C} \left\{\varepsilon(k+1)\right\}^{-1} H(k+1) = \\
&= P(k)^{-1} + H(k+1)^T \mathsf{C} \left\{\varepsilon(k+1)\right\}^{-1} H(k+1)
\end{aligned}
$$

This equation is of *paramount importance*. Indeed, since $P(k)^{-1}$ represents the *Fisher Information Matrix* (FIM), i.e., the amount of information collected from the sensor, it states that *whatever is the uncertainty of the new measurements, it will increase the information about* x.

The Least Squares Algorithm

Recursive solution

In other words, *all* the sensor readings are useful for the estimates: the information grows *linearly*.
To recover the *covariance matrix* $P(k+1)$ we have to invert the previous matrix equation, which is not trivial.
However, we can make use of the *matrix inversion lemma*:

$$(A + BCB^T)^{-1} = A^{-1} - A^{-1}B(B^T A^{-1} B + C^{-1})^{-1} B^T A^{-1}.$$

The Least Squares Algorithm

Recursive solution

Therefore:

$$P(k+1) = P(k) - P(k)H(k+1)^T \left(H(k+1)P(k)H(k+1)^T + \mathsf{C}\left\{\varepsilon(k+1)\right\}\right)^{-1} H(k+1)P(k).$$

The term

$$S(k+1) = H(k+1)P(k)H(k+1)^T + \mathsf{C}\left\{\varepsilon(k+1)\right\},$$

is called the *covariance of the residuals*. It corresponds to the uncertainty accumulated in the estimates of x up to the previous step plus the uncertainty for the new measures.

The Least Squares Algorithm

Recursive solution

The matrix

$$W(k+1) = P(k)H(k+1)^T \left(H(k+1)P(k)H(k+1)^T + \mathsf{C}\left\{\varepsilon(k+1)\right\}\right)^{-1} = = P(k)H(k+1)^T S(k+1)^{-1},$$

is called the *update gain*, and it weights the decrease of the uncertainty as a function of the new measures.
Finally we have:

$$P(k+1) = (I - W(k+1)H(k+1))P(k) = = P(k) - W(k+1)S(k+1)W(k+1)^T.$$

The Least Squares Algorithm

Recursive solution

For the estimates we have:

$$\begin{aligned}\hat{x}(k+1) &= P(k+1)H^{k+1^T}C^{k+1^{-1}}Z^{k+1} = \\ &= P(k+1)H^{k^T}C^{k^{-1}}Z^k + \\ &+ P(k+1)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1}z(k+1)\end{aligned}$$

First, notice that

$$\begin{aligned}&P(k+1)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\ &= (I - W(k+1)H(k+1))P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\ &= P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} + \\ &- W(k+1)H(k+1)P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1}.\end{aligned}$$

The Least Squares Algorithm

Recursive solution

Then

$$
\begin{aligned}
&P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\
&= P(k)H(k+1)^T S(k+1)^{-1} S(k+1) \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\
&= W(k+1) S(k+1) \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\
&= W(k+1)(H(k+1)P(k)H(k+1)^T + \mathsf{C}\left\{\varepsilon(k+1)\right\}) \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\
&= W(k+1)H(k+1)P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} + W(k+1).
\end{aligned}
$$

Hence, plugged in the previous equation, yields

$$
\begin{aligned}
&P(k+1)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\
&= P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} + \\
&- W(k+1)H(k+1)P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = \\
&= W(k+1)H(k+1)P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} + W(k+1) + \\
&- W(k+1)H(k+1)P(k)H(k+1)^T \mathsf{C}\left\{\varepsilon(k+1)\right\}^{-1} = W(k+1).
\end{aligned}
$$

The Least Squares Algorithm

Recursive solution

Hence, by substituting in $\hat{x}(k+1)$ the previous and the update equation for $P(k+1)$, we have

$$\begin{aligned}
&\hat{x}(k+1) = \\
&= (I - W(k+1)H(k+1))\, P(k){H^k}^T {C^k}^{-1} Z^k + W(k+1)z(k+1) = \\
&= P(k){H^k}^T {C^k}^{-1} Z^k + W(k+1)(z(k+1) - H(k+1)P(k){H^k}^T {C^k}^{-1} Z^k).
\end{aligned}$$

We then recognise that

$$\hat{x}(k) = P(k){H^k}^T {C^k}^{-1} Z^k.$$

Finally we get:

$$\hat{x}(k+1) = \hat{x}(k) + W(k+1)\left(z(k+1) - H(k+1)\hat{x}(k)\right).$$

The Least Squares Algorithm

Recursive solution

Remark

The update rule of the estimates for the recursive LS estimator

$$\hat{x}(k+1) = \hat{x}(k) + W(k+1)\left(z(k+1) - H(k+1)\hat{x}(k)\right),$$

is given by the previous estimates plus a correction term. This term is the product between a gain and a residual.

Remark

The residual is the difference between the new measures $z(k+1)$ and the predicted value of the measures based on the sensor model $H(k+1)$ and the available estimates $\hat{x}(k)$.

The Least Squares Algorithm

Recursive solution

Remark

It is easy now to see that the covariance of the residuals is:

$$S(k+1) = \\ = E\left\{(z(k+1) - H(k+1)\hat{x}(k))(z(k+1) - H(k+1)\hat{x}(k))^T\right\}.$$

The Least Squares Algorithm

Recursive solution

To recap, the *non Bayesian* WLS recursive solution is given by the following equations, to be computed iteratively once a new set of measures become available:

$$
\begin{aligned}
S(k+1) &= H(k+1)P(k)H(k+1)^T + \mathsf{C}\{\varepsilon(k+1)\} \\
W(k+1) &= P(k)H(k+1)^T S(k+1)^{-1} \\
\hat{x}(k+1) &= \hat{x}(k) + W(k+1)\left(z(k+1) - H(k+1)\hat{x}(k)\right) \\
P(k+1) &= (I - W(k+1)H(k+1))P(k)
\end{aligned}
$$

The Least Squares Algorithm

Recursive solution

Remark

Since the presented scheme is recursive, it needs an initialisation. This is done by using a batch technique on a small amount of data or using "a priori" information on the value of x and of its covariance matrix P.

The Least Squares Algorithm

Recursive solution

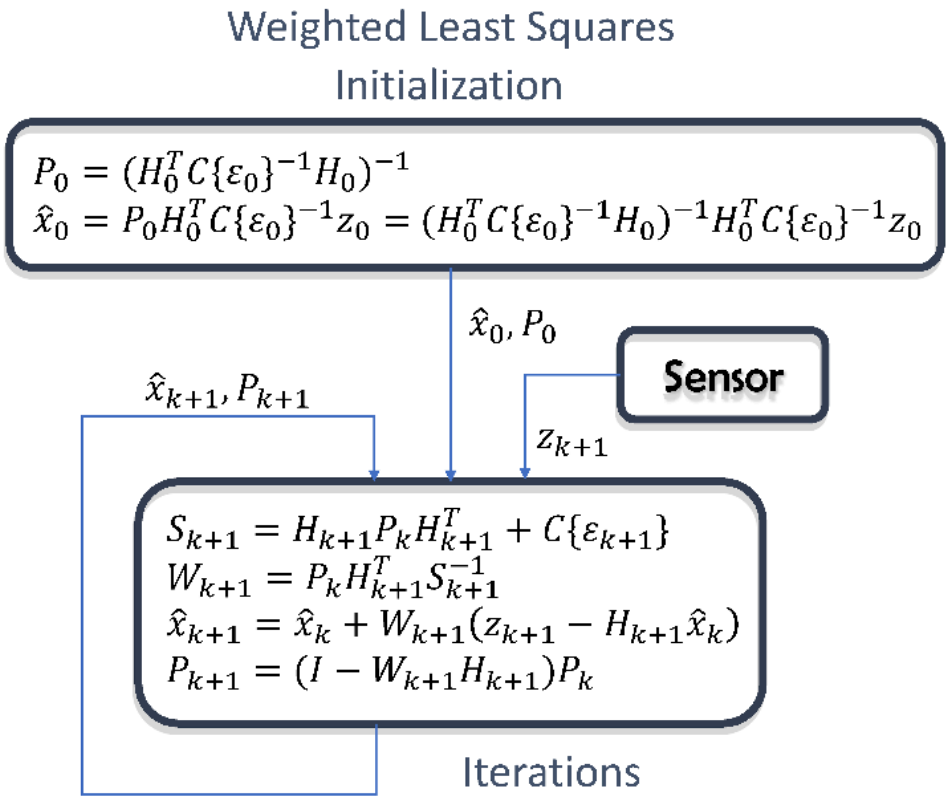

Figure: Recursive WLS scheme with initialisation.

The Least Squares Algorithm

Recursive solution: example

Consider noisy scalar observations of a *nonrandom* quantity

$$z(i) = x + \varepsilon(i), \quad i = 1, \ldots, k$$

Assuming $\varepsilon(i)$ zero-mean *i.i.d.* with variance σ^2, we have:

$$H^k = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad \text{and} \quad C^k = \sigma^2 I.$$

The Least Squares Algorithm

Recursive solution: example

For the *batch* solution we have

$$\hat{x}(k) = \left(H^{k^T} C^{k^{-1}} H^k\right)^{-1} C^{k^{-1}} H^k Z^k = \frac{1}{k} \sum_{i=1}^{k} z(i),$$

and

$$P(k) = \left(H^{k^T} C^{k^{-1}} H^k\right)^{-1} = \frac{\sigma^2}{k},$$

the same results obtained previously.

The Least Squares Algorithm

Recursive solution: example

For the *recursive* solution, we obviously start from $\hat{x}(1) = z(1)$ and $P(1) = \sigma^2$. Hence, after k steps we have:

$$\hat{x}(k) = \left(H^{k^T} C^{k^{-1}} H^k \right)^{-1} C^{k^{-1}} H^k Z^k = \frac{1}{k} \sum_{i=1}^{k} z(i),$$

and

$$P(k) = \left(H^{k^T} C^{k^{-1}} H^k \right)^{-1} = \frac{\sigma^2}{k},$$

as in the *batch* case.

The Least Squares Algorithm

Recursive solution: example

For the $k+1$ step, involving the $k+1$–th measure, we have then:

$$\begin{aligned} S(k+1) &= H(k+1)P(k)H(k+1)^T + \mathsf{C}\left\{\varepsilon(k+1)\right\} = \\ &= \frac{\sigma^2}{k} + \sigma^2 = \frac{k+1}{k}\sigma^2 \\ W(k+1) &= P(k)H(k+1)^T S(k+1)^{-1} = \\ &= \frac{\sigma^2}{k}\left(\frac{k+1}{k}\sigma^2\right)^{-1} = \frac{1}{k+1} \end{aligned}$$

The Least Squares Algorithm

Recursive solution: example

Hence:

$$\begin{aligned}
\hat{x}(k+1) &= \hat{x}(k) + W(k+1)\left(z(k+1) - H(k+1)\hat{x}(k)\right) = \\
&= \hat{x}(k) + \frac{1}{k+1}\left(z(k+1) - H(k+1)\hat{x}(k)\right) = \\
&= \frac{1}{k+1}\sum_{i=1}^{k+1} z(i), \\
P(k+1) &= (I - W(k+1)H(k+1))P(k) = \\
&= (1 - \frac{1}{k+1})\frac{\sigma^2}{k} = \frac{\sigma^2}{k+1}
\end{aligned}$$

as expected.

Outline

1. Linear Estimators
   - The Least Squares solution
   - The Kalman filter
2. Nonlinear estimators
   - The Non-linear Least Squares solution
   - The Extended Kalman filter

Kalman Filter

The most popular Bayesian filter is the Kalman Filter (KF), named after the developers Kalman and Bucy.
Since it is a Bayesian filter, we will use the *Bayes theorem*.
In doing so, we will recall that a Bayesian solution is the combination of the *prior* and the *likelihood*. Recalling, what we have seen for the *Maximum A Posteriori* (MAP), which is equivalent to the MMSE in the Gaussian linear case, we modify the *Maximum Likelihood* (ML) estimator with the *prior*.
The *prior* comes from the system dynamics $x(k) = f(x(k-1), \nu(k-1))$.
We recall that in the linear case, Gaussian case, ML and WLS are the same, so given the *recursive solution* of the WLS presented previously, we have just to add the effect of the *prior*.

Kalman Filter

We define, as usual, $x^k = \{x(i)\}_{i=0,\dots,k}$ and $Z^k = \{z(i)\}_{i=1,\dots,k}$ and we are considering the prior $p(x(0))$, which is a given. Notice that both $x(k)$ and $z(k)$ are *discrete time continuous stochastic processes*.
Hence, the *Bayes theorem* application reads like this:

$$p(x(k)|Z^k) = \frac{p(Z^k|x(k))p(x(k))}{p(Z^k)},$$

in which the *posterior*, the *likelihood* and the *prior* are clearly visible.
For the normalisation factor, we have (*conditional probability*)

$$p(Z^k) = p(z(1), z(2), \dots, z(k)) = p(z(k)|Z^{k-1})p(Z^{k-1}).$$

Kalman Filter

We can easily recognise here that the *likelihood* can be rewritten as (*conditional probability*)

$$p(Z^k|x(k)) = p(z(k)|x(k), Z^{k-1})p(Z^{k-1}|x(k)),$$

and that, using the Markovian property and the fact that the noises are white

$$p(Z^k|x(k)) = p(z(k)|x(k))p(Z^{k-1}).$$

Substituting the normalisation factor and the likelihood in the original Bayes formulation, we have

$$p(x(k)|Z^k) = \frac{p(Z^k|x(k))p(x(k))}{p(Z^k)} = \frac{p(z(k)|x(k))p(x(k))}{p(z(k)|Z^{k-1})}.$$

Kalman Filter

We then notice that the *prior* is given by all the measurements up to time $k-1$.
Hence, we first notice that by the marginalisation on the previous states $x^{k-1} = [x(0), \ldots, x(k-1)]^T$ we have

$$p(x(k)) = p(x(k)|Z^{k-1}) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} p(x(k), x^{k-1}|Z^{k-1}) dx^{k-1}.$$

Kalman Filter

Since the noise $\nu(k-1)$ acting on the system dynamics $x(k) = f(x(k-1), \nu(k-1))$ is *white*, the system is *Markovian* and hence we can make use of the *Markovian property* to get

$$p(x(k), x^{k-1}|Z^{k-1}) = p(x(k), x(k-1)|Z^{k-1}).$$

Then, using the *conditional pdf*, we have

$$p(x(k), x^{k-1}|Z^{k-1}) = p(x(k)|x(k-1), Z^{k-1})p(x(k-1)|Z^{k-1}).$$

Finally, the prior is given by

$$p(x(k)) = p(x(k)|Z^{k-1}) = \int_{-\infty}^{+\infty} p(x(k)|x(k-1))p(x(k-1)|Z^{k-1})dx(k-1).$$

Kalman Filter

For the denominator of the Bayes equation (i.e., the normalisation factor), we consider the *Total Probability Law* and the Markovian property, thus having

$$p(z(k)|Z^{k-1}) = \int_{-\infty}^{+\infty} p(z(k)|x(k))p(x(k)|Z^{k-1})dx(k),$$

that is, as usual, the integral of the numerator (normalisation).

Kalman Filter

We can then easily recognise two steps for the Bayes filter:

- The first computes the *new* estimate given all the previous measurements. In doing so, it makes use of the *system dynamics*, i.e.

$$p(x(k)|Z^{k-1}) = \int_{-\infty}^{+\infty} p(x(k)|x(k-1))p(x(k-1)|Z^{k-1})dx(k-1).$$

  This step is called *prediction*: it predicts the next value $x(k)$ given the previous knowledge $x(k-1)$;
- The second refines the prior with the *new set of measurements* (i.e. the new gained knowledge) using the likelihood function, i.e.

$$p(x(k)|Z^k) = \frac{p(z(k)|x(k))p(x(k)|Z^{k-1})}{p(z(k)|Z^{k-1})}.$$

  Since this second step updates the prediction with new knowledge, it is called *update*.

Kalman Filter

We can now interpret the second step as just a refined estimate by means of the dynamic model that can be used in a WLS scheme.
This is exactly the spirit of the MAP recalled previously: refine the estimate of a non-Bayesian approach with the *knowledge* coming from the system model.

Kalman filter

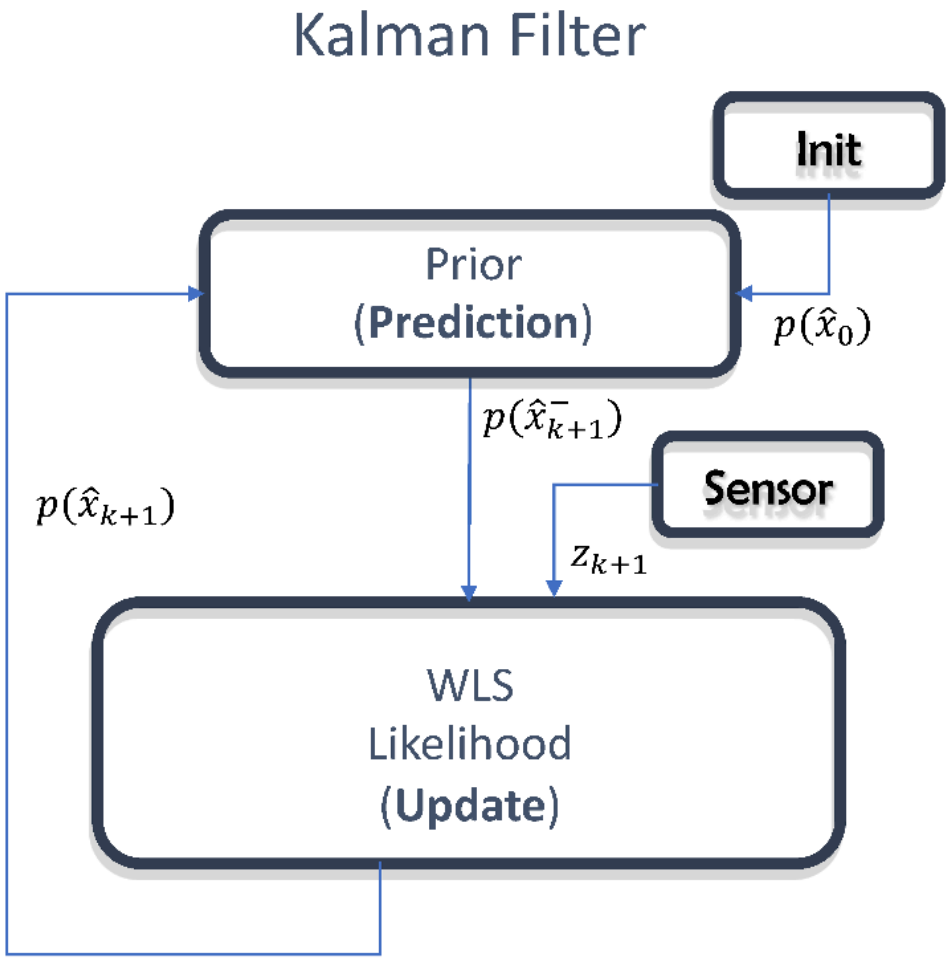

Figure: Bayes scheme with WLS as update step.

Kalman filter

Gaussian noises example for a scalar linear system

Let us consider the following scalar and linear system ($x(k), \nu(k) \in \mathbb{R}$):

$$x(k+1) = ax(k) + \nu(k),$$

where $\nu(k) \sim \mathcal{N}(0, \sigma_\nu^2)$ (i.e., a zero-mean i.i.d. sequence).
If we assume to know $x(k)$, then $\mathsf{E}\{x(k+1)|x(k)\} = ax(k)$. However, this assumption comes with some level of uncertainty, that is in general described by assuming $\mathsf{V}\{x(k+1)|x(k)\} = \sigma^2$.
We want to compute explicitly the prior

$$p(x(k+1)|Z^k) = \int_{-\infty}^{+\infty} p(x(k+1)|x(k))p(x(k)|Z^k)dx(k).$$

Kalman filter

Gaussian noises example for a scalar linear system

To this end, we immediately have that

$$p(x(k+1)|x(k)) = p(x_{k+1}|x_k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x_{k+1}-ax_k)^2}{\sigma^2}}.$$

Instead, we assume in general that $x_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$, i.e.

$$p(x_k|Z^k) = p(x_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{1}{2}\frac{(x_k-\mu_k)^2}{\sigma_k^2}}.$$

Therefore, the prior is given by

$$p(x_{k+1}|Z^k) = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{1}{2}\frac{\sigma_k^2(x_{k+1}-ax_k)^2+\sigma^2(x_k-\mu_k)^2}{\sigma^2\sigma_k^2}} dx_k.$$

Kalman filter

Gaussian noises example for a scalar linear system

Let us rewrite the exponent as a quadratic function of x_k (the integration variable), i.e.

$$\sigma_k^2(x_{k+1}-ax_k)^2+\sigma^2(x_k-\mu_k)^2 = b_2x_k^2+b_1x_k+b_0 = b_2\left(x_k+\frac{b_1}{2b_2}\right)^2+b_0-\frac{b_1^2}{4b_2}$$

Substituting, we have immediately that

$$\begin{aligned}
&\sigma_k^2(x_{k+1}-ax_k)^2+\sigma^2(x_k-\mu_k)^2 = \\
&= (\sigma^2+a^2\sigma_k^2)\left(x_k-\frac{a\sigma_k^2x_{k+1}+\sigma^2\mu_k}{\sigma^2+a^2\sigma_k^2}\right)^2+\gamma = \\
&= (\sigma^2+a^2\sigma_k^2)\,(x_k-\alpha)^2+\gamma,
\end{aligned}$$

where

$$\gamma = \frac{(\sigma^2+a^2\sigma_k^2)(\sigma^2\mu_k^2+\sigma_k^2x_{k+1}^2)-(a\sigma_k^2x_{k+1}+\sigma^2\mu_k)^2}{\sigma^2+a^2\sigma_k^2}.$$

Kalman filter

Gaussian noises example for a scalar linear system

Therefore, substituting back in the integral

$$p(x_{k+1}|Z^k) = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{1}{2}\frac{(\sigma^2+a^2\sigma_k^2)(x_k-\alpha)^2+\gamma}{\sigma^2\sigma_k^2}} dx_k =$$

$$= \int_{-\infty}^{+\infty} \frac{1}{\sqrt{4\pi^2\sigma^2\sigma_k^2}} e^{-\frac{1}{2}\frac{(x_k-\alpha)^2}{\frac{\sigma^2\sigma_k^2}{\sigma^2+a^2\sigma_k^2}}} e^{-\frac{1}{2}\frac{\gamma}{\sigma^2\sigma_k^2}} dx_k =$$

$$= \frac{1}{\sqrt{2\pi(\sigma^2+a^2\sigma_k^2)}} e^{-\frac{1}{2}\frac{\gamma}{\sigma^2\sigma_k^2}} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\frac{\sigma^2\sigma_k^2}{\sigma^2+a^2\sigma_k^2}}} e^{-\frac{1}{2}\frac{(x_k-\alpha)^2}{\frac{\sigma^2\sigma_k^2}{\sigma^2+a^2\sigma_k^2}}} dx_k.$$

Kalman filter

Gaussian noises example for a scalar linear system

Hence

$$p(x_{k+1}|Z^k) = \frac{1}{\sqrt{2\pi(\sigma^2 + a^2\sigma_k^2)}} e^{-\frac{1}{2}\frac{\gamma}{\sigma^2\sigma_k^2}}.$$

After some algebra, it follows that

$$\gamma = \frac{\sigma^2\sigma_k^2}{\sigma^2 + a^2\sigma_k^2}(x_{k+1} - a\mu_k)^2.$$

Therefore

$$p(x_{k+1}|Z^k) = \frac{1}{\sqrt{2\pi(\sigma^2 + a^2\sigma_k^2)}} e^{-\frac{1}{2}\frac{(x_{k+1}-a\mu_k)^2}{\sigma^2+a^2\sigma_k^2}},$$

i.e., a Gaussian pdf $x_{k+1}|Z^k \sim \mathcal{N}(a\mu_k, \sigma^2 + a^2\sigma_k^2)$.

Kalman filter

Gaussian noises example for a scalar linear system

From the definition of the dynamic systems

$$x(k+1) = ax(k) + \nu(k),$$

we have immediately that

$$\mathsf{V}\left\{x(k+1)\right\} = a^2\sigma_k^2 + \sigma_\nu^2,$$

hence $\sigma^2 = \sigma_\nu^2$ i.e., the variance of the r.v. given the model!
Notice how the variance inevitably increases in the *prediction* step.
At this point, we can notice that the same argument used for the MAP can be applied, hence the fusion with the measurement follows the WLS.

Kalman filter

Gaussian noises example for a generic linear system

Let us consider the following linear system with $x(k), \nu(k) \in \mathbb{R}^n$:

$$x(k+1) = Ax(k) + \nu(k),$$

where $\nu(k) \sim \mathcal{N}(0, Q)$ (i.e., a zero-mean i.i.d. sequence).
As before, if we assume to know $x(k)$, then $\mathsf{E}\{x(k+1)|x(k)\} = Ax(k)$.
However, this assumption comes with some level of uncertainty, that is in general described by assuming $\mathsf{V}\{x(k+1)|x(k)\} = \Sigma$.
We want to compute explicitly the prior

$$p(x(k+1)|Z^k) = \int_{-\infty}^{+\infty} p(x(k+1)|x(k))p(x(k)|Z^k)dx(k).$$

Kalman filter

Gaussian noises example for a generic linear system

To this end, we immediately have that

$$p(x_{k+1}|x_k) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}(x_{k+1}-Ax_k)^T \Sigma^{-1}(x_{k+1}-Ax_k)}.$$

Instead, we assume in general that $x_k \sim \mathcal{N}(\mu_k, P_k)$, i.e.

$$p(x_k|Z^k) = p(x_k) = \frac{1}{\sqrt{|2\pi P_k|}} e^{-\frac{1}{2}(x_k-\mu_k)^T P_k^{-1}(x_k-\mu_k)}.$$

Kalman filter

Gaussian noises example for a generic linear system

Following the same steps as before for the matrix form, we end up with

$$p(x_{k+1}|Z^k) = \frac{1}{\sqrt{|2\pi(\Sigma + AP_kA^T)|}} e^{-\frac{1}{2}(x_{k+1}-A\mu_k)^T(\Sigma+AP_kA^T)^{-1}(x_{k+1}-A\mu_k)},$$

i.e., a Gaussian pdf $x_{k+1}|Z^k \sim \mathcal{N}(A\mu_k, \Sigma + AP_kA^T)$.

Kalman filter

Gaussian noises example for a generic linear system

From the definition of the dynamic systems

$$x(k+1) = Ax(k) + \nu(k),$$

we have immediately that

$$\mathrm{V}\{x(k+1)\} = AP_kA^T + Q,$$

hence $\Sigma = Q$ i.e., the covariance matrix of the r.v. $x(k+1)$ given the model!
Notice how the uncertainty inevitably increases in the *prediction* step since AP_kA^T and Q are both p.d.
At this point, we can notice that the same argument used for the MAP can be applied, hence the fusion with the measurement follows the WLS.

Kalman filter

Gaussian noises

To summarise, assuming that $\mathsf{E}\{\nu(k)\nu(j)\} = Q(k)\delta_{k,j}$, $\nu(k) \sim \mathcal{N}(0, Q(k))$, $\mathsf{E}\{\varepsilon(k)\varepsilon(j)\} = R(k)\delta_{k,j}$, $\varepsilon(k) \sim \mathcal{N}(0, R(k))$, $\mathsf{E}\{\nu(k)\varepsilon(j)\} = 0$, $\forall i, j$, and

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + G(k)\nu(k), \\ z(k) &= H(k)x(k) + F(k)\varepsilon(k), \end{aligned}$$

we have

$$\begin{aligned} p(x(k)|Z^k) &= \mathcal{N}(\hat{x}(k), P(k)), \\ p(x(k+1)|x(k)) &= \mathcal{N}(A(k)\hat{x}(k) + B(k)u(k), G(k)Q(k)G(k)^T), \\ p(z(k+1)|x(k+1)) &= \mathcal{N}(H(k+1)\hat{x}(k+1), R(k+1)), \\ p(x(k+1)|Z^{k+1}) &= \mathcal{N}(\hat{x}(k+1), P(k+1)). \end{aligned}$$

Kalman filter

Gaussian noises

In other words, in the Gaussian case, we can just propagate the *mean* $\hat{x}(k)$ and the *covariance matrix* $P(k)$, starting from an initial knowledge $\hat{x}(0)$ and $P(0)$, to estimate the state by its mean, i.e. the most probable state for the Gaussian.
The fact that we are using the mean as the estimate is perfectly in agreement with the *Type A evaluation* in *metrology* and reported in the GUM!
Notice that $P(k) = \mathsf{E}\left\{(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^T\right\} = \mathsf{E}\left\{\tilde{x}(k)\tilde{x}(k)^T\right\}$ is the covariance matrix of the *estimation error*, hence the *filter precision*.
If no knowledge is available at the beginning, just select $\hat{x}(0) = 0$ and $P(0)$ *large enough* to express the ignorance (absence of knowledge, in the information sense).

Kalman filter

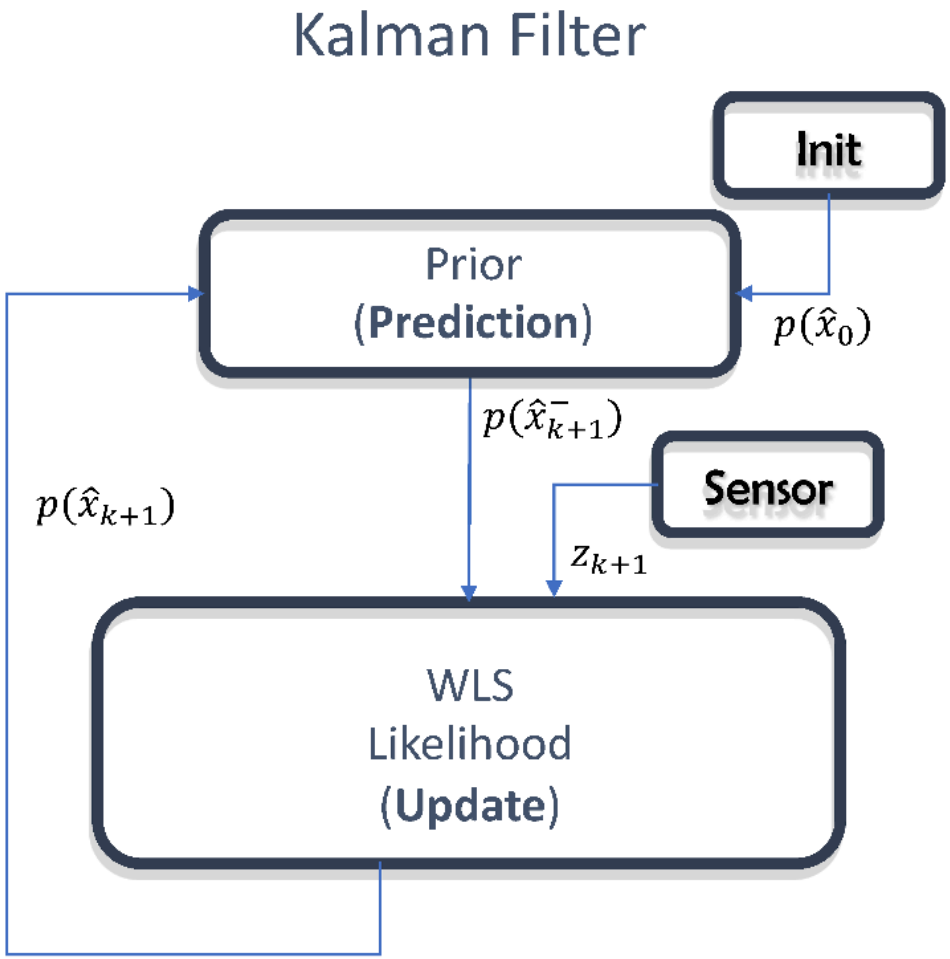

Figure: Bayes scheme with WLS as update step.

Kalman filter

Gaussian noises

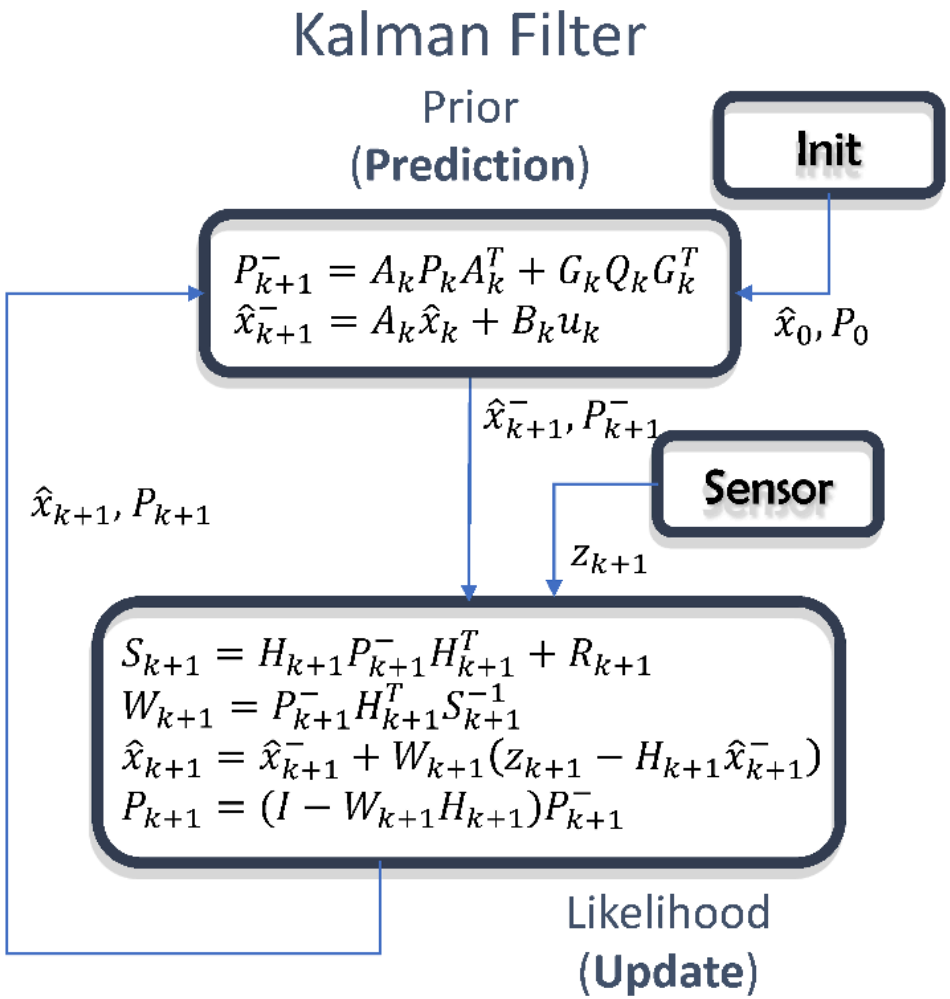

Figure: KF scheme.

Kalman filter

Summary

Summary on the Kalman Filter - Gaussian case

Kalman filter

Summary

The two equations of the *Kalman Filter* are:

- *Prediction step* (based on the model, i.e., the *prior*):

$$\hat{x}(k+1)^- = A(k)\hat{x}(k) + B(k)u(k)$$
$$P(k+1)^- = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T$$

- *Update step* (based on the measurements, i.e., the *posterior*):

$$S(k+1) = H(k+1)P(k+1)^- H(k+1)^T + R(k+1)$$
$$W(k+1) = P(k+1)^- H(k+1)^T S(k+1)^{-1}$$
$$\hat{x}(k+1) = \hat{x}(k+1)^- + W(k+1)\left(z(k+1) - H(k+1)\hat{x}(k+1)^-\right)$$
$$P(k+1) = (I - W(k+1)H(k+1))P(k+1)^-$$

Kalman filter

Summary

- If the *process noise* $\nu(k) \sim \mathcal{N}(0, Q(k))$, the state $x(k) \sim \mathcal{N}(\hat{x}(k), \mathsf{C}\{x(k)\})$;
- If the *measurement noise* $\varepsilon(k) \sim \mathcal{N}(0, R(k))$, than $z(k) = H(k)x(k) + \varepsilon(k)$ is $z(k) \sim \mathcal{N}(\mu_z, \mathsf{C}\{z(k)\})$.

Remark

Recall that $\mathsf{C}\{x(k)\} = \mathsf{E}\{(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^T\}$ *by definition. Hence,* $\mathsf{C}\{x(k)\} = \mathsf{E}\{\tilde{x}(k)\tilde{x}(k)^T\} = P(k)$.

Remark

Recall that if $\mu_\varepsilon \neq 0$, *this bias can be removed by sensor calibration.*

Kalman filter

Summary

Let us see how μ_x, $\mathsf{C}\{x(k)\}$, μ_z and $\mathsf{C}\{z(k)\}$ are given by the *Kalman filter*:

- $\mu_x = \hat{x}(k)$;
- $\mathsf{C}\{x(k)\} = A(k-1)P(k-1)A(k-1)^T + G(k-1)Q(k-1)G(k-1)^T$;
- $\mu_z = \mathsf{E}\{H(k)x(k) + \varepsilon(k)\} = H(k)\hat{x}(k)$;
- $\mathsf{C}\{z(k)\} = \mathsf{E}\left\{(z-\mu_z)(z-\mu_z)^T\right\}$, hence:

$$\mathsf{C}\{z(k)\} = \mathsf{E}\left\{(H(k)\tilde{x}(k) + \varepsilon(k))(H(k)\tilde{x}(k) + \varepsilon(k))^T\right\},$$

Notice that $\tilde{x}(k)$ depends on the noise $\nu(k-1)$. If $\nu(k)$ and $\varepsilon(j)$ are *uncorrelated* $\forall k, j$, it follows

$$\mathsf{C}\{z(k)\} = H(k)P(k)H(k)^T + R(k),$$

that is the *covariance of the residuals* $S(k)$ found previously;

Kalman filter

Summary

Moreover:

- $\mathsf{C}\{x, z\} = \mathsf{E}\left\{(x - \mu_x)(z - \mu_z)^T\right\}$, that is:

$$\mathsf{C}\{x, z\} = \mathsf{E}\left\{\tilde{x}(k)(H(k)\tilde{x}(k) + \varepsilon(k))^T\right\}$$

If $\nu(k)$ and $\varepsilon(j)$ are *uncorrelated* $\forall k, j$, it follows

$$\mathsf{C}\{x, z\} = \mathsf{E}\{\tilde{x}(k)\tilde{x}(k)\} H(k)^T = P(k)H(k)^T.$$

It then follows that x and z are *jointly Gaussian*!
Let us see what are the implications of being *jointly Gaussian*.

Kalman filter

Summary

If x and z are jointly Gaussian, the optimal estimator is the MMSE, whose solution equation are given by:

$$\mathsf{E}\{x|z\} = \mu_x + P_{xz}P_{zz}^{-1}(z - \mu_z) \text{ and } \mathsf{C}\{x|z\} = P_{xx} - P_{xz}P_{zz}^{-1}P_{zx}.$$

Hence, if the noises are Gaussian, zero-mean and white, i.e. $x(k) \sim \mathcal{N}(\hat{x}(k), \mathsf{C}\{x(k)\})$ and $z(k) \sim \mathcal{N}(\mu_z, \mathsf{C}\{z(k)\})$, is the Kalman filter the *optimal linear MMSE estimator*?
To answer this question, let us see how those quantities are *computed and combined* in the Kalman filter.

Kalman filter

Summary

At time k, we have:

- $\mu_x = \hat{x}(k)$;
- $P_{xx} = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T = P(k)$;
- $\mu_z = \mathsf{E}\{H(k)x(k) + \varepsilon(k)\} = H(k)\hat{x}(k)$;
- $P_{zz} = \mathsf{E}\left\{(z - \mu_z)(z - \mu_z)^T\right\} = H(k)P(k)H(k)^T + R(k) = S(k)$;
- $P_{xz} = P_{zx}^T = \mathsf{E}\left\{(x - \mu_x)(z - \mu_z)^T\right\} = \mathsf{E}\left\{\tilde{x}(k)\tilde{x}(k)\right\} H(k)^T = P(k)H(k)^T$.

Kalman filter

Summary

By plugging the previous equations into the MMSE solution we finally have:

$$
\begin{aligned}
\mathsf{E}\{x|z\} &= \mu_x + P_{xz}P_{zz}^{-1}(z-\mu_z) = \\
&= \hat{x}(k) + P(k)H(k)^T S(k)^{-1}(z - H(k)\hat{x}(k)) = \\
&= \hat{x}(k) + W(k)(z - H(k)\hat{x}(k)), \\
\mathsf{C}\{x|z\} &= P_{xx} - P_{xz}P_{zz}^{-1}P_{zx} = \\
&= P(k) - P(k)H(k)^T S(k)^{-1}H(k)P(k) = \\
&= P(k) - W(k)H(k)P(k) = (I - W(k)H(k))P(k),
\end{aligned}
$$

i.e. *exactly the equation of the Kalman filter*.

Kalman filter

Gaussian noises

Remark

The Kalman filter is an MMSE estimator *and, hence,* the Kalman filter is the optimal filter for Gaussian noises, *while it is the* best linear filter in all the other cases.

Kalman filter

Summary

This fact is not surprising at all since the Kalman filter is split into two phases:

- *Prediction step*: it computes the evolution of the estimates based on the knowledge of the stochastic description of $\hat{x}(k)$, i.e., $P(k)$, and the stochastic noise process $\nu(k)$. In practice this the *prior* we want to use;
- *Update step*: it computes the updated values of the estimates based on the *likelihood function*, which is the same function maximised in the LS for Gaussian noises.

Hence, the *Kalman filter* is a linear MMSE *Bayesian estimator*, which combines the prior with the likelihood function.

Kalman filter

Summary

Remark

The Kalman filter presented in these slides is just the discrete-discrete Kalman filter, in which both the model and the measurement process are assumed to be discrete-time.
This is not the only choice available. Indeed there exists also other versions, such as the continuous-continuous or the continuous-discrete Kalman filters, that can be adopted depending on the particular problem.

Remark

We restrict ourselves to the discrete-discrete Kalman filters because are the most relevant for applications, especially when the information on the models and on the measures are known only at discrete time intervals.
This is always the case when we have to deals with distributed systems.

Kalman filter

Summary

Summary on the Kalman Filter - Generic case

Kalman filter

Summary

We assume a *wide sense whiteness* property on the noises, i.e.,

$$\begin{aligned}
\mathsf{E}\{\nu(k)\} &= \mu_\nu \\
\mathsf{C}\{\nu(k)\} &= \mathsf{E}\left\{(\nu(k)-\mu_\nu)(\nu(j)-\mu_\nu)^T\right\} = Q(k)\delta_{kj} \\
\mathsf{E}\{\varepsilon(k)\} &= \mu_\varepsilon \\
\mathsf{C}\{\varepsilon(k)\} &= \mathsf{E}\left\{(\varepsilon(k)-\mu_\varepsilon)(\varepsilon(j)-\mu_\varepsilon)^T\right\} = R(k)\delta_{kj}
\end{aligned}$$

with $\mu_\nu = \mu_\varepsilon = 0$
and noises *uncorrelatedness*

$$\mathsf{E}\left\{(\nu(k)-\mu_\nu)(\varepsilon(j)-\mu_\varepsilon)^T\right\} = 0, \quad \forall k, j.$$

Notice that *no assumption has been made on the pdf*.

Kalman filter

Summary

The Kalman filter is split into two phases:

- *Prediction step*: it computes the evolution of the estimates based on the knowledge of the stochastic description of $\tilde{x}(k)$, i.e., $P(k)$, and the stochastic noise process $\nu(k)$. In practice this the *prior* we want to use;
- *Update step*: it computes the updated values of the estimates based on the *likelihood function* of the *measurements*.

Hence, the *Kalman filter* is a linear *Bayesian estimator*, which combines the prior with the likelihood function.
Therefore, the filter is the *best linear filter* for noises that are generically distributed. Indeed, by approximating the noise as Gaussian, a *linear approximation* is actually carried out.

Kalman filter

Summary

The two equations of the *Kalman Filter* are in general given exactly in the same way:

- *Prediction step* (based on the model, i.e., the *prior*):

$$\hat{x}(k+1)^- = A(k)\hat{x}(k) + B(k)u(k)$$
$$P(k+1)^- = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T$$

- *Update step* (based on the measurements, i.e., the *posterior*):

$$S(k+1) = H(k+1)P(k+1)^- H(k+1)^T + R(k+1)$$
$$W(k+1) = P(k+1)^- H(k+1)^T S(k+1)^{-1}$$
$$\hat{x}(k+1) = \hat{x}(k+1)^- + W(k+1)\left(z(k+1) - H(k+1)\hat{x}(k+1)^-\right)$$
$$P(k+1) = (I - W(k+1)H(k+1))P(k+1)^-$$

Again, the *Update step* comprises the WLS solution for the *iterative solution*.

Kalman filter

Summary

The fact that the Kalman filter remains unchanged for non Gaussian noise requires a different perspective to be actually shown, which is given next.

Luenberger observer

An alternative presentation using Luenberger observers

Luenberger observer

Let us consider the problem of *estimating a certain state variable* $x \in \mathbb{R}^n$, with *known linear time-varying dynamic* governed by the *known inputs* $u \in \mathbb{R}^m$.
The information on the state x are provided by a set of sensors returning the measurement values $z \in \mathbb{R}^p$.
Therefore

$$\begin{cases} x(k+1) &= A(k)x(k) + B(k)u(k) \\ z(k) &= H(k)x(k) \end{cases}$$

Luenberger observer

In this case it is possible to nullify the *estimation error* $\tilde{x}(k) = x(k) - \hat{x}(k)$ using the following observer

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)),$$

that, providing the system is *observable* (or at least *detectable*), ensures $\tilde{x}(k) \to 0$ by properly choosing $L(k)$, i.e.

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k).$$

Luenberger observer

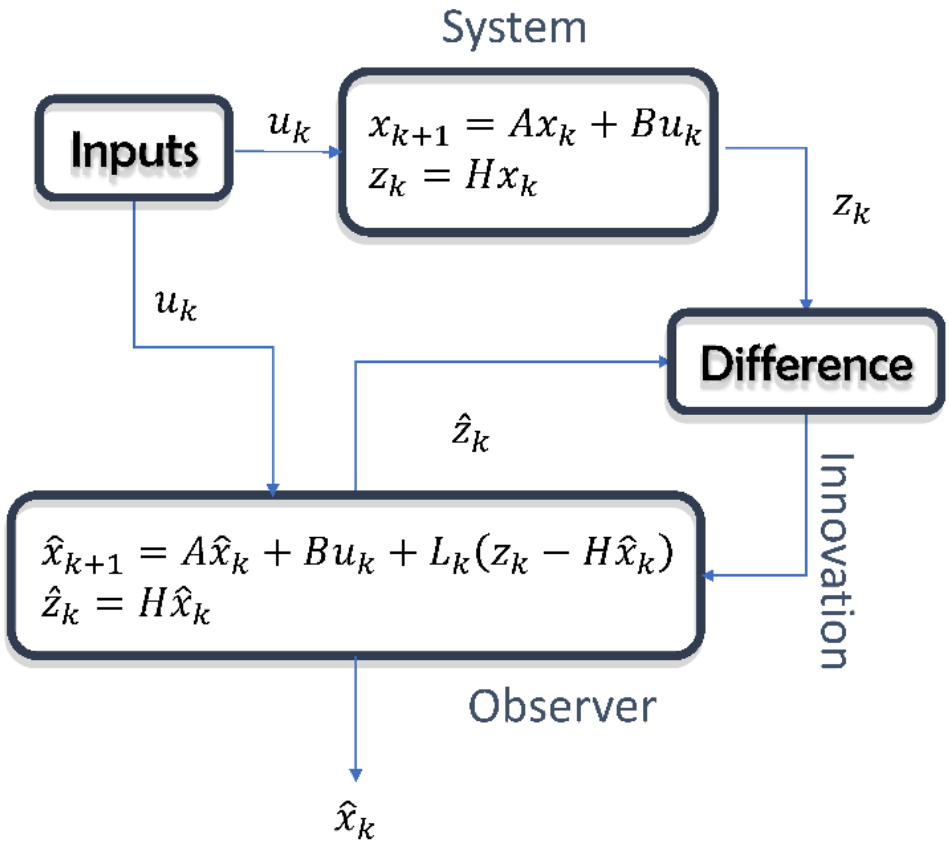

Figure: Luenberger observer scheme.

Kalman filter

Let us now consider the problem of *estimating a certain state variable* $x \in \mathbb{R}^n$, with *known linear dynamic* governed by the *known inputs* $u \in \mathbb{R}^m$ and *model noise* $\nu \in \mathbb{R}^q$.
The information on the variable x are provided by a set of sensors returning the measures $z \in \mathbb{R}^p$, which are affected by the *measurement noise*, i.e. *random effects*, $\varepsilon \in \mathbb{R}^p$.

Kalman filter

The *time varying, discrete-time linear* model we are dealing with is then

$$\begin{cases} x(k+1) &= A(k)x(k) + B(k)u(k) + G(k)\nu(k) \\ z(k) &= H(k)x(k) + \varepsilon(k) \end{cases}$$

Notice that $\nu(k)$ models either a) the *imperfect* application of the actuation, b) the *noise* in the actuation commands sensors, c) *no or partial knowledge* of the system inputs.

Kalman filter

Hence, starting from

$$\begin{cases} x(k+1) &= A(k)x(k) + B(k)u(k) + G(k)\nu(k) \\ z(k) &= H(k)x(k) + \varepsilon(k) \end{cases}$$

we can apply the same idea of the *Luenberger observer*, as

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)),$$

that yields the modified estimation error version

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k) + G(k)\nu(k) + L(k)\varepsilon(k).$$

It is evident that the terms $G(k)\nu(k) + L(k)\varepsilon(k)$ modify the picture a little bit...

Luenberger observer

Indeed, the *estimation error* $\tilde{x}(k) = x(k) - \hat{x}(k)$ is now a **rv**, hence the convergence concept $\tilde{x}(k) \to 0$ *cannot be applied* as is.
In these cases, a *convergence under the expected operator* is instead considered, i.e. $\mathsf{E}\{\tilde{x}(k)\} \to 0$.
By considering

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k) + G(k)\nu(k) + L(k)\varepsilon(k),$$

it then follows that convergence is attained if *necessarily* $\mathsf{E}\{\nu(k)\} = \mathsf{E}\{\varepsilon(k)\} = 0$, which yields to

$$\mathsf{E}\{\tilde{x}(k+1)\} = (A(k) - L(k)H(k))\mathsf{E}\{\tilde{x}(k)\},$$

as in the *Luenberger observer* case.

Luenberger observer

From the previous condition $\mathsf{E}\{\nu(k)\} = \mathsf{E}\{\varepsilon(k)\} = 0$, we can easily recognised that the *estimated* value is the *mean value* of the **rv** $x(k)$, i.e. $\hat{x}(k) = \mathsf{E}\{x(k)\}$.
This is perfectly in agreement with the *Type A evaluation* in *metrology* and reported in the GUM!

Luenberger observer

We finally noticed that for the Luenberger observer we have

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)),$$

that can be rewritten as:

$$\begin{aligned}\hat{x}(k+1)^- &= A(k)\hat{x}(k) + B(k)u(k),\\ \hat{x}(k) &= \hat{x}(k)^- + W(k)(z(k) - H(k)\hat{x}(k)^-),\end{aligned}$$

where the superscript $\cdot^-$ stands for *model-based* estimate, i.e. *prior estimate*, while the second equation is the *measurement-based* estimation, i.e. *posterior estimate*.
Indeed, by substituting the second in the first:

$$\hat{x}(k+1)^- = A(k)\hat{x}(k)^- + B(k)u(k) + A(k)W(k)(z(k) - H(k)\hat{x}(k)^-),$$

where $L(k) = A(k)W(k)$, to have the same Luenberger observer update rule.

Kalman filter

Model

We will now revisit the previous concepts using the linear model previously introduced, i.e.

$$\hat{x}(k+1)^- = A(k)\hat{x}(k) + B(k)u(k)$$
$$\hat{x}(k+1) = \hat{x}(k+1)^- + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1)^-),$$

where, again, the superscript $\cdot^-$ stands for the *predicted* estimate.

Kalman filter

Model

The dynamic of the *estimation error* $\tilde{x}(k) = x(k) - \hat{x}(k)$ using only the *prior knowledge on the model*, i.e. the *prior estimation error* $\tilde{x}(k)^-$, is

$$\begin{aligned}\tilde{x}(k+1)^- &= x(k+1) - \hat{x}(k+1)^- = \\ &= A(k)x(k) + B(k)u(k) + G(k)\nu(k) - (A(k)\hat{x}(k) + B(k)u(k)) = \\ &= A(k)(x(k) - \hat{x}(k)) + G(k)\nu(k) = A(k)\tilde{x}(k) + G(k)\nu(k).\end{aligned}$$

By the previous considerations, we have

$$\mathsf{E}\left\{\tilde{x}(k+1)^-\right\} = \mathsf{E}\left\{x(k+1)\right\} - \hat{x}(k+1)^- = 0.$$

Kalman filter

Model

For the *prior covariance matrix* of the *prior estimation error* $\tilde{x}(k+1)^-$ we have then

$$\begin{aligned}
P(k+1)^- &= \mathsf{E}\left\{\tilde{x}(k+1)\tilde{x}(k+1)^T\right\} = \\
&= \mathsf{E}\left\{(A(k)\tilde{x}(k)+G(k)\nu(k))(A(k)\tilde{x}(k)+G(k)\nu(k))^T\right\} \\
&= \mathsf{E}\left\{A(k)\tilde{x}(k)\tilde{x}(k)^T A(k)^T\right\} + \mathsf{E}\left\{G(k)\nu(k)\nu(k)^T G(k)^T\right\} + \\
&\mathsf{E}\left\{A(k)\tilde{x}(k)\nu(k)^T G(k)^T\right\} + \mathsf{E}\left\{G(k)\nu(k)\tilde{x}(k)^T A(k)^T\right\} = \\
&= A(k)\mathsf{E}\left\{\tilde{x}(k)\tilde{x}(k)^T\right\} A(k)^T + G(k)\mathsf{E}\left\{\nu(k)\nu(k)^T\right\} G(k)^T,
\end{aligned}$$

where the mixed terms

$$A(k)\mathsf{E}\left\{\tilde{x}(k)\nu(k)^T\right\} G(k)^T \text{ and } G(k)\mathsf{E}\left\{\nu(k)\tilde{x}(k)^T\right\} A(k)^T$$

are both zero if and only if $\nu(k)$ is generated by a *white* process (*prove it!*).

Kalman filter

Model

Hence, for the *prior covariance matrix* of the *prior estimation error* $\tilde{x}(k+1)^-$ we finally have

$$P(k+1)^- = A(k)\mathsf{E}\left\{\tilde{x}(k)\tilde{x}(k)^T\right\}A(k)^T + G(k)\mathsf{E}\left\{\nu(k)\nu(k)^T\right\}G(k)^T = \\ = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T,$$

where $Q(k)$ is the *covariance matrix* of the model uncertainties.

Kalman filter

Model

To summarise:

- $$\hat{x}(k+1)^- = A(k)\hat{x}(k) + B(k)u(k),$$
$$P(k+1)^- = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T;$$
- The *model noise* $\nu(k)$ should be *white* and *zero mean*.

Kalman filter

Measurement

For the measurements, we have the *innovations* given by

$$\begin{aligned} z(k+1) - \hat{z}(k+1) &= H(k+1)x(k+1) + \varepsilon(k+1) - H(k+1)\hat{x}(k+1)^- = \\ &= H(k+1)(x(k+1) - \hat{x}(k+1)^-) + \varepsilon(k+1) = \\ &= H(k+1)\tilde{x}(k+1)^- + \varepsilon(k+1) = e_z(k+1)^-. \end{aligned}$$

In the spirit of the *Luenberger observer* we have the *measurement updated* estimates as

$$\begin{aligned} \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - \hat{z}(k+1)) = \\ &= \hat{x}(k+1) + W(k+1)e_z(k+1)^- = \\ &= \hat{x}(k+1) + W(k+1)(H(k+1)\tilde{x}(k+1)^- + \varepsilon(k+1)). \end{aligned}$$

As in the observer case, the matrix $W(k+1)$ is a *design parameter* to be determined, so why don't we find it *optimally*?

Kalman filter

Measurement

For instance, we can determine the optimal gain $W(k+1)$ so that the *covariance of the error* between the posterior state estimates $\hat{x}(k+1)$ and the actual state $x(k+1)$, i.e. the *posterior covariance matrix*, is minimum! To this end, the *posterior estimation error* $\tilde{x}(k+1)$, i.e. *after* the measurement updates, is by definition:

$$\begin{aligned}\tilde{x}(k+1) &= x(k+1) - \hat{x}(k+1) = x(k+1) - \hat{x}(k+1)^- + \\ &- W(k+1)H(k+1)\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1) = \\ &= \tilde{x}(k+1)^- - W(k+1)H(k+1)\tilde{x}(k+1)^- + \\ &- W(k+1)\varepsilon(k+1) = \\ &= (I - W(k+1)H(k+1))\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1).\end{aligned}$$

Kalman filter

Measurement

The *posterior covariance matrix* $P(k+1)$ of the *posterior estimation error* $\tilde{x}(k+1)$ is then given by

$$\begin{aligned} P(k+1) &= \mathsf{E}\left\{\tilde{x}(k+1)\tilde{x}(k+1)^T\right\} \\ &= \mathsf{E}\Big\{((I - W(k+1)H(k+1))\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1))\cdot \\ &\quad ((I - W(k+1)H(k+1))\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1))^T\Big\}. \end{aligned}$$

As for the *prior covariance matrix* $P(k+1)^-$, if $\tilde{x}(k+1)^-$ and $\varepsilon(k+1)$ are *uncorrelated*, the previous equation simplifies a lot.

Kalman filter

Measurement

To understand what is the condition that should be satisfied in order to have $\tilde{x}(k+1)^-$ and $\varepsilon(k+1)$ *uncorrelated*, we noticed that:

- $\tilde{x}(k+1)^- = A(k)\tilde{x}(k) + G(k)\nu(k)$;
- $\tilde{x}(k)^- = A(k-1)\tilde{x}(k-1) + G(k)\nu(k-1)$;
- $\tilde{x}(k) = (I - W(k)H(k))\tilde{x}(k)^- - W(k)\varepsilon(k)$;
- The *model uncertainties* $\nu(k)$ are supposed to be generated by a *white* stochastic process.

Therefore,

$$\begin{aligned}\tilde{x}(k+1)^- &= A(k)\tilde{x}(k) + G(k)\nu(k) = \\ &= A(k)[(I - W(k)H(k))\tilde{x}(k)^- - W(k)\varepsilon(k)] + G(k)\nu(k) = \\ &= A(k)[(I - W(k)H(k))\{A(k-1)\tilde{x}(k-1) + G(k)\nu(k-1)\}+ \\ &- W(k)\varepsilon(k)] + G(k)\nu(k),\end{aligned}$$

i.e., it is *correlated* with $\nu(k)$, $\nu(k-1)$ and $\varepsilon(k)$.

Kalman filter

Measurement

Therefore, to have $\tilde{x}(k+1)^-$ and $\varepsilon(k+1)$ *uncorrelated*, we need to have:

- $\varepsilon(k)$ generated by a *white* stochastic process;
- $\mathrm{E}\{\varepsilon(k)\nu(k)\} = 0$.

Kalman filter

Measurement

For ease of notation, let us drop the reference to the $(k+1)$-th time instant, i.e. $P(k+1)^- = P^-$.
With the previous conditions on *uncorrelatedness*, the *posterior covariance matrix* $P(k+1)$ simplifies

$$\begin{aligned} P &= \mathsf{E}\left\{((I-WH)\tilde{x}^- - W\varepsilon)((I-WH)\tilde{x}^- - W\varepsilon)^T\right\} = \\ &= (I-WH)P^-(I-WH)^T + WRW^T. \end{aligned}$$

where R is the *covariance matrix* of the *measurement uncertainties* $\varepsilon(k)$.

Kalman filter

Measurement

As a consequence, the *posterior covariance matrix* $P(k+1)$ is finally given by

$$\begin{aligned} P &= (I - WH)P^{-}(I - WH)^T + WRW^T = \\ &= P^{-} - WHP^{-} - P^{-}H^TW^T + WHP^{-}H^TW^T + WRW^T = \\ &= P^{-} - WHP^{-} - P^{-}H^TW^T + W(HP^{-}H^T + R)W^T = \\ &= P^{-} - WHP^{-} - P^{-}H^TW^T + WSW^T. \end{aligned}$$

Kalman filter

Measurement

Let us focus on $S = HP^{-}H^{T} + R$. It is related to the *measurements* $z = Hx + \varepsilon$, since R is the covariance matrix of ε.
Let us consider the *innovations* $e_z(k+1)^{-}$: the *mean value* is

$$\mathsf{E}\left\{e_z^{-}\right\} = \mathsf{E}\left\{(Hx + \varepsilon - H\hat{x}^{-})\right\} = H\mathsf{E}\left\{\tilde{x}^{-}\right\} + \mathsf{E}\left\{\varepsilon\right\} = \mathsf{E}\left\{\varepsilon\right\} = 0.$$

Kalman filter

Measurement

Therefore, the *covariance matrix* of the innovations is

$$\mathsf{E}\left\{e_z^-(e_z^-)^T\right\} = \mathsf{E}\left\{(H\tilde{x}^- + \varepsilon)(H\tilde{x}^- + \varepsilon)^T\right\}.$$

Again, there is the problem of the *cross products*. However, being ε generated by a *white* stochastic process as stated previously, we have

$$\mathsf{E}\left\{e_z^-(e_z^-)^T\right\} = HP^-H^T + R,$$

which is exactly S!

Kalman filter

Measurement

Let us switch back to the *posterior covariance matrix* $P(k+1)$

$$P = P^- - WHP^- - P^-H^TW^T + WSW^T.$$

One way to minimise the *posterior covariance matrix* is to *minimise its trace*, i.e. the sum of the variances on the matrix $P(k+1)$ diagonal, i.e.

$$\frac{\partial \text{Trace}\,(P)}{\partial W} = -2(HP^-)^T + 2WS = 0$$
$$WS = P^-H^T$$
$$W = P^-H^TS^{-1}$$

which is the *optimal Kalman gain* to minimise the *posterior covariance matrix*, i.e. *maximising the estimator precision*.

Kalman filter

Measurement

Therefore, by plugging the *Kalman gain* W in the following

$$P = P^- - WHP^- - P^-H^TW^T + WSW^T,$$

we finally have

$$\begin{aligned} P &= P^- - WHP^- - P^-H^TW^T + P^-H^TS^{-1}SW^T = \\ &= P^- - WHP^- = (I - WH)P^-. \end{aligned}$$

Kalman filter

Remark

The relation $P = (I - WH)P^{-}(I - WH)^{T} + WRW^{T}$ *retrieved previously in the derivation of* W *is called* Joseph form of the covariance update *and can be used in place of* $P = (I - WH)P^{-}$ *because more numerically stable.*

Outline

1. Linear Estimators
   - The Least Squares solution
   - The Kalman filter
2. Nonlinear estimators
   - The Non-linear Least Squares solution
   - The Extended Kalman filter

Outline

1. Linear Estimators
   - The Least Squares solution
   - The Kalman filter
2. Nonlinear estimators
   - The Non-linear Least Squares solution
   - The Extended Kalman filter

Non-linear Least Squares solution

Usually the measurement process is *nonlinear*, i.e.

$$z(k) = h(x) + \varepsilon(k),$$

where $h(x)$ is a generic nonlinear function of the states.
The WLS problem thus becomes

$$\hat{x}^{WLS}(k) = \arg\min_x (Z^k - h(x))^T W (Z^k - h(x)) = \arg\min_x J(x,k)$$

where $W = W^T$ and W p.d.
In such a case it becomes very hard to find the solution to the previous problem, since we have to find

$$\frac{dJ(x,k)}{dx} = 0$$

where $J(x,k)$ is a *generic nonlinear vectorial function*, i.e. the *gradient*.

Non-linear Least Squares solution

Indeed, we recall the derivative of nonlinear functions of x, i.e., given $s(x)$ and $r(x)$ be two *generic nonlinear vectorial functions* of the vector x and A a matrix of suitable dimension, we have

$$\frac{ds(x)^T Ar(x)}{dx} = \left(\frac{ds(x)}{dx}\right)^T Ar(x) + \left(\frac{dr(x)}{dx}\right)^T A^T s(x);$$

We then compute the explicit form of the *gradient* $J(x,k)$ by expanding the products as

$$J(x,k) = {Z^k}^T W Z^k - h(x)^T W Z^k - {Z^k}^T W h(x) + h(x)^T W h(x)$$

It then follows that we are searching for the solution to

$$\frac{dJ(x,k)}{dx} = -H(x)^T W(Z^k - h(x)) = 0,$$

where $H(x) = \frac{dh(x)}{dx}$ is the *Jacobian* of the output forward map $h(x)$.

Non-linear Least Squares solution

To this end, let us first recall that given a generic nonlinear scalar function $g(x)$, without *local minima* and *differentiable*, the problem of finding a solution $\bar{x}$ such that

$$g(\bar{x}) = 0,$$

can be approached with the *Newton-Raphson method*.

Non-linear Least Squares solution

Newton-Raphson method

(1) Select a generic point x_0;

(2) Compute the first order *Taylor expansion* around x_0

$$0 = g(\bar{x}) \approx g(x_0) + \left.\frac{dg(x)}{dx}\right|_{x=x_0} (\bar{x} - x_0) = g(x_0) + g'(x_0)(\bar{x} - x_0);$$

(3) Hence we have

$$\bar{x} = x_0 - \frac{g(x_0)}{g'(x_0)}.$$

Of course, if the function $g(x)$ is linear then the first order *Taylor expansion* holds with an equality, hence $\bar{x}$ is the desired solution.
If the function $g(x)$ is nonlinear, we have to resort to an *iterative solution*.

Non-linear Least Squares solution

Newton-Raphson method

Hence:

(1) Select a generic point x_q, with $q = 0$ (as before);

(2) Compute the first order *Taylor expansion* around x_q (as before);

(3) Compute the solution $\bar{x}$, which now is renamed x_{q+1}, i.e.,

$$x_{q+1} = x_q - \frac{g(x_q)}{g'(x_q)} = x_q + \Delta x_q;$$

(4) If $|x_{q+1} - x_q| = |\Delta x_q|$ is *below a certain threshold*, the algorithm stops and returns x_{q+1} as the point in which $g(x_{q+1}) \approx 0$. Otherwise, switch back to Step 2.

This is an *estimator*, since $x_{q+1} = \hat{\bar{x}}$, i.e. the point in which $g(\bar{x}) = 0$, but it is a *numerical algorithm*.

Non-linear Least Squares solution

Using this idea, we can apply the following algorithm for the nonlinear WLS:

Let's start with the *gradient* function

$$g(x) = \frac{dJ(x,k)}{dx} = -H(x)^T W(Z^k - h(x)) = 0,$$

and let us apply the *Newton-Raphson method*, as follows:

(1) Select a generic state x_q;

(2) Compute the first order *Taylor expansion* around x_q:

$$g(x) \approx g(x_q) + \left.\frac{dg(x)}{dx}\right|_{x=x_q} (x_{q+1} - x_q) = g(x_q) + G(x_q)\Delta x_q = 0,$$

where $G(x_q)$ is the *Hessian matrix* of $J(x,k)$ evaluated in x_q;

(3) Solve with respect to $\Delta x_q = x_{q+1} - x_q$, i.e.,

$$\Delta x_q = -G(x_q)^{-1} g(x_q) \Rightarrow x_{q+1} = x_q - G(x_q)^{-1} g(x_q).$$

Non-linear Least Squares solution

Newton-Raphson method

(4) If $|x_{q+1} - x_q| = |\Delta x_q|$ is *below a certain threshold*, the algorithm stops and returns x_{q+1} as the point in which $g(x_{q+1}) \approx 0$. Otherwise, switch back to Step 2.

In general, the *Newton-Raphson method* can be described as:

$$x_{q+1} = x_q - \mathcal{H}(J(x,k))^{-1} \nabla J(x,k),$$

where $\mathcal{H}(J(x,k))$ and $\nabla J(x,k)$ are the *Hessian* and the *gradient* of $J(x,k)$.

Non-linear Least Squares solution

Since we are dealing with the *Nonlinear WLS*, recall that the *gradient* is

$$g(x) = \frac{dJ(x,k)}{dx} = -H(x)^T W(Z^k - h(x)),$$

Hence the *Hessian* evaluated in x_q is

$$G(x_q) = \left.\frac{dg(x)}{dx}\right|_{x=x_q} = H(x_q)^T W H(x_q)$$

and the *gradient*

$$g(x_q) = -H(x_q)^T W(Z^k - h(x_q))$$

We can substitute the *explicit* values of the *gradient* and the *Hessian* at step q of the *Newton-Raphson method*, yielding

$$\begin{aligned} x_{q+1} &= x_q + \Delta x_q = x_q - G(x_q)^{-1} g(x_q) = \\ &= x_q + (H(x_q)^T W H(x_q))^{-1} H(x_q)^T W(Z^k - h(x_q)) \end{aligned}$$

where $\Delta y_q = Z^k - h(x_q)$ are the *linearised residuals* (more on this in what

Non-linear Least Squares solution

Again, if $|x_{q+1} - x_q| = |\Delta x_q|$ is below a certain threshold, the algorithm stops and returns x_{q+1} as x. Otherwise, switch back to Step 2.
This solution is called the *Gauss-Newton method*.
Notice that since the *Hessian* is a function of the *Jacobian* of the nonlinear forward map $h(x)$ for the *Nonlinear WLS*, it turns out that *the Hessian is not computed de facto*.

Non-linear Least Squares solution

Notice that for the *Gauss-Newton method*

$$\Delta x_q = x_{q+1} - x_q = (H(x_q)^T W H(x_q))^{-1} H(x_q)^T W (Z^k - h(x_q)),$$

is the WLS solution for a *linearised problem*.
Therefore, the *Gauss-Newton method* can also be interpreted like this: given x_q, compute the first order *Taylor expansion* around x_q of the output function:

$$h(x_{q+1}) \approx h(x_q) + \left.\frac{dh(x)}{dx}\right|_{x=x_q} (x_{q+1} - x_q) = h(x_q) + H(x_q)\Delta x_q,$$

where $H(x_q)$ is the *Jacobian matrix* of the output function $h(x)$ evaluated in x_q, as reported previously.

Non-linear Least Squares solution

Define the following *linear* WLS

$$\Delta x_q^{WLS} = \arg\min_{\Delta x_q}(Z^k - h(x_{q+1}))W(Z^k - h(x_{q+1}))^T \approx$$
$$\approx \arg\min_{\Delta x_q}(Z^k - h(x_q) - H(x_q)\Delta x_q)W(Z^k - h(x_q) - H(x_q)\Delta x_q)^T =$$
$$= \arg\min_{x}(\Delta y_q - H(x_q)\Delta x_q)W(\Delta y_q - H(x_q)\Delta x_q)^T,$$

where we have substituted the *first order Taylor linearised* function $h(x_{q+1})$ and $\Delta y_q = Z^k - h(x_q)$ are, again and more clearly, the *linearised residuals*. The solution is then given as customary

$$\Delta x_q = (H(x_q)^T W H(x_q))^{-1} H(x_q)^T W \Delta y_q,$$

which finally generates $x_{q+1} = x_q + \Delta x_q$.

Non-linear Least Squares solution

Remark

In many practical problems, the Hessian matrix $G(x_q) = H(x_q)^T W H(x_q)$ is quite sparse (i.e., it has a lot of zeros) and it is huge. However, its inverse $G(x_q)^{-1}$ will not be sparse.

Remark

In general, the Hessian matrix $G(x_q)$ is not inverted. Instead, the following normal equation is considered:

$$G(x_q)\Delta x_q = H(x_q)^T W \Delta y_q.$$

To solve the previous equations, the matrix $G(x_q)$ is factorised in its triangular components (Cholesky decomposition) and then the forward/backward substitution algorithm is used.

Non-linear Least Squares solution

Steepest descent

As a final comment, another approach that actually involves just the computation of the *gradient* of the function and follows exactly *the same iterative procedure* is the *steepest descent method*.
In general, the *steepest descent method* can be described as:

$$x_{q+1} = x_q - \alpha \nabla J(x, k),$$

where $\nabla J(x, k)$ is again the *gradient* of $J(x, k)$.
The parameter $\alpha > 0$ is a constant which controls the *convergence rate* and *stability*: the larger is α, the faster is the convergence at the price of a reduced stability.
The main idea underlying this simple approach is to *move* the solution fallowing the *gradient* direction.

Outline

1. Linear Estimators
   - The Least Squares solution
   - The Kalman filter
2. Nonlinear estimators
   - The Non-linear Least Squares solution
   - The Extended Kalman filter

Extended Kalman filter

Let us consider the following *time varying nonlinear discrete time* dynamic system:

$$\begin{cases} x(k+1) &= f_k(x(k), u(k), \nu(k)) \\ z(k) &= h_k(x(k), \varepsilon(k)) \end{cases}$$

Extended Kalman filter

We have assumed a *wide sense whiteness* property on the noises, i.e.,

$$\begin{aligned}
&\mathsf{E}\left\{\nu(k)\right\} = \mu_\nu \\
&\mathsf{C}\left\{\nu(k)\right\} = \mathsf{E}\left\{(\nu(k)-\mu_\nu)(\nu(j)-\mu_\nu)^T\right\} = Q(k)\delta_{kj} \\
&\mathsf{E}\left\{\varepsilon(k)\right\} = \mu_\varepsilon \\
&\mathsf{C}\left\{\varepsilon(k)\right\} = \mathsf{E}\left\{(\varepsilon(k)-\mu_\varepsilon)(\varepsilon(j)-\mu_\varepsilon)^T\right\} = R(k)\delta_{kj}
\end{aligned}$$

Moreover, it is usually assumed that

$$\mathsf{E}\left\{(\nu(k)-\mu_\nu)(\varepsilon(j)-\mu_\varepsilon)^T\right\} = 0, \quad \forall k, j,$$

i.e., *uncorrelated noises*.

Extended Kalman filter

Since the system is nonlinear, the *Kalman filter* (KF) cannot be applied as is.
To *extend* the application of the KF to this class of systems, we first notice that the process prediction and the measurement equations *can be used as they are* in the formulation of the KF.
What changes is the way in which the *covariances* are propagated and the way in which the *Kalman gain* $W(k)$ is computed, which indeed subsumes a *linear process*.

Extended Kalman filter

We recall here that in order to compute an approximated propagation of covariances for a *nonlinear combination* of **rv**s, we can make use of the *law of propagation of errors*, aka *law of propagation of the uncertainties*.
In that case, a *Taylor linearised* model can be used.
Therefore, the *Extended Kalman Filter* (EKF) is just a filter that makes use of linearised models for covariances propagation and gain computation.
The linearisation involves the knowledge of the actual estimates $\hat{x}(k)$ at time k.

Extended Kalman filter

For the *Prediction step* (based on the model, i.e., in *open loop*), we have:

$$\hat{x}(k+1)^- = f_k(\hat{x}(k), u(k))$$
$$P(k+1)^- = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T$$

where the *linearised model* for the covariances is given by

$$A(k) = \left.\frac{f_k(x,u,\nu)}{dx}\right|_{\substack{x=\hat{x}(k)\\ u=u(k)\\ \nu=0}}$$

$$G(k) = \left.\frac{f_k(x,u,\nu)}{d\nu}\right|_{\substack{x=\hat{x}(k)\\ u=u(k)\\ \nu=0}}$$

Extended Kalman filter

Similarly, for the *Update step* (based on the measurements, i.e., in *closed loop*), we have:

$$
\begin{aligned}
S(k+1) &= H(k+1)P(k+1)^{-}H(k+1)^{T} + R(k+1) \\
W(k+1) &= P(k+1)^{-}H(k+1)^{T}S(k+1)^{-1} \\
\hat{x}(k+1) &= \hat{x}(k+1)^{-} + W(k+1)\left(z(k+1) - h_{k+1}(\hat{x}(k+1)^{-})\right) \\
P(k+1) &= (I - W(k+1)H(k+1))P(k+1)^{-}
\end{aligned}
$$

where the *linearised model* for the covariances and the gain is given by

$$
H(k+1) = \left.\frac{h_{k+1}(x,\varepsilon)}{dx}\right|_{\substack{x=\hat{x}(k+1)^{-} \\ \varepsilon=0}}
$$

Extended Kalman filter

Remark

There are other solutions provided in the literature for the KF, which can be efficiently used in particular cases. For example, the Unscented Kalman Filter, that makes use of the unscented transformation, is a possible choice in the nonlinear case.
The unscented transformation is explicitly conceived for the nonlinear propagation of the uncertainties.

Remark

Other approaches, instead, consider the model as a blended combination of elementary models, among which the process may switch. These are called Interactive Multiple Models.