Dynamic data processing

Master's Degree in Geomatics Engineering and Geoinformation

Academic Year 2017-2018

E.T.S.I.Geodesica, Cartografíca y Topográfica

Outline

- ✓Introduction
- ✓Least squares estimation (LS)
- ✓Recursive LS
- ✓Kalman filtering

Introduction

Most engineering problems involves the estimation of unknown parameters from a set of redundant measurements.

Reasons for collecting redundant measurements:

✓To be able to check for mistakes or errors.
✓The wish to increase the accuracy of the results computed.

Exact measurements does not exist! $\rightarrow$ the redundant data are usually inconsistent $\rightarrow$ parameters have to be determined uniquely $\rightarrow$ adjustment

Computational process of making the measurement data consistent with the model such that the unknown parameters can be determined uniquely.

Introduction

Geodetic/Surveying applications $\rightarrow$ Generally, superabundant measurements are collected to determine parameters that are constant over time $\rightarrow$ static parameters are derived from a whole set of measurements in a batch processing once the field campaign has finished (post-processing)

Nonetheless, parameters can be constant (e.g. geodetic network coordinates) or time-varying (e.g. coordinates of a moving object).

Time-varying parameters can be

Geometric (position, attitude, shape, …)
Physical (temperature, humidity, atmospheric delays, solar radiation, etc…)
Instrumental (clock drift and bias, scale, calibration parameters,…).

Introduction

Navigation $\rightarrow$ the estimated parameters $\vec{x}$ include at least a position vector, a velocity vector, and possibly three additional parameters to describe attitude.

That set of parameters are usually called state vector, and it is time-varying.

In navigation, the estimation of the state vector is usually required to be in real-time, or near/quasi real-time with a low latency so that navigation and guidance can be possible.

Least squares formulation has to be adapted in order to:

✓ Yield a new parameter vector $\vec{x}$ as soon as a new measurement vector $\vec{y}$ is available.
✓ Reduce the computational load (e.g. by reducing data storing)

Introduction (RLS)

Recursive least-squares (RLS) adjustment

✓A parameter solution is said to be recursive when the method of determination enables sequential, rather than batch processing of the measurement data.

✓The essence is that it enables one to update the parameter estimates for new measurements $\vec{y}_t$ without the need to store and use all past measurements $\vec{y}_{t-1}$,…

✓Reduced computational load.

✓There is no need to store measurements.

Introduction (Kalman filtering)

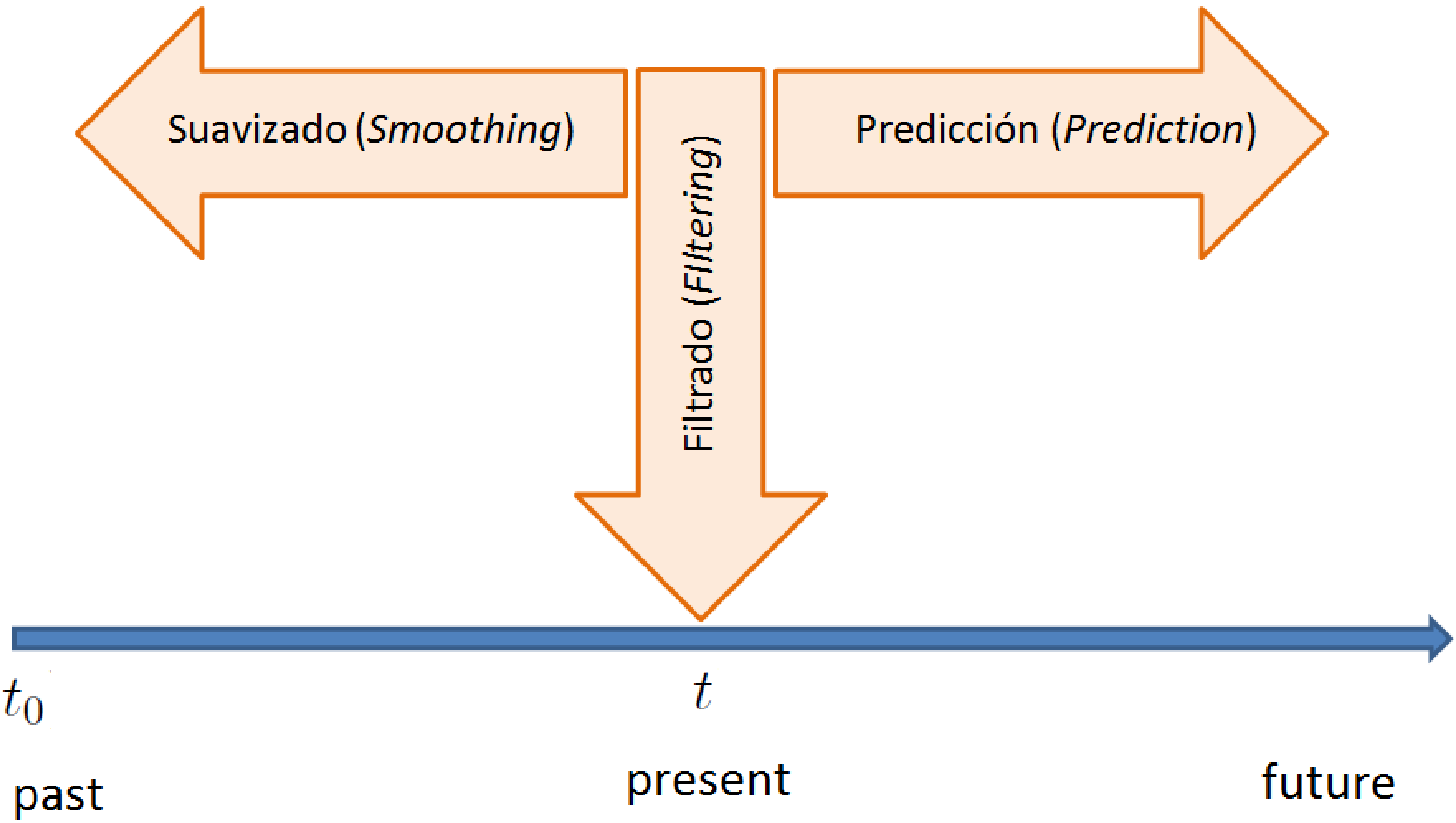
When determining time-varying parameters from sequentially collected measurement data, there are three possible types of estimation problems:

Filtering $\rightarrow$ when the time at which a parameter estimate is required concides with the time the last measurements are collected.

Smoothing $\rightarrow$ when the time at which a parameter estimate is required falls within the time span of available measurement data.

Prediction $\rightarrow$ when the time of interest occurs after the time the last measurements are collected.

Introduction (Kalman filtering)



Introduction (Kalman filtering)

Kalman filter–based estimation techniques have many applications in navigation:

- Fine alignment and calibration of INS
- GNSS navigation
- GNSS signal monitoring
- INS/GNSS integration
- Multisensor integration

For alignment and calibration of an INS:
states estimated → position, velocity, and attitude errors, together with inertial instrument errors, such as accelerometer and gyro biases.
Measurements → position, velocity, and/or attitude differences between the aligning-INS navigation solution and an external reference, such as another INS or GNSS.

For INS/GNSS and multisensor integration:
states estimated → number of errors of the constituent navigation systems, and the navigation solution itself.
Measurements → vary greatly, depending on the type of integration implemented (e.g. INS/GNSS integration)

Least squares estimation (LS)

Standard approach assumes

Stationary random variables $\rightarrow$ the state vector and its stochastic behaviour are not a function of time.

LS estimation is obtained using a whole set of data $\rightarrow$ all the measurement data is available for a post-processing batch-mode solution.

This strategy serves as an introductory step $\rightarrow$ then, recursive LS and Kalman filtering are subsequently derived.

Kalman filtering notation

Standard LS approach

Here we introduce the notation normally used in navigation (KF)

$$\vec{y} = H\vec{x} + \vec{v}$$

$\vec{x}$	Unknown parameter vector ($n \times 1$)
$\vec{y}$	Measurement data vector ($m \times 1$)
$\vec{v}$	Residual vector or observation noise ($m \times 1$)
H	Configuration or design matrix ($m \times n$ partial derivatives)

$$\vec{y} \sim N(0, R) \qquad \vec{v} \sim N(0, R)$$

R Variance-covariance matrix ($m \times m$)

Standard LS approach

Notation normally used in navigation (KF)

$$\hat{\vec{x}} = (H^T R^{-1} H)^{-1} H^T R^{-1} \vec{y}$$

$$P = (H^T R^{-1} H)^{-1}$$

$$\boxed{\hat{\vec{x}} = P H^T R^{-1} \vec{y}}$$

Recursive LS

Let be $\vec{y}_0(m_0 \times 1)$ and $\vec{y}_1(m_1 \times 1)$ two subsets of incorrelated measurements, then the linear model can be witten

$$\begin{pmatrix} \vec{y}_0 \\ \vec{y}_1 \end{pmatrix} = \begin{pmatrix} H_0 \\ H_1 \end{pmatrix} \vec{x} + \begin{pmatrix} \vec{v}_0 \\ \vec{v}_1 \end{pmatrix} \qquad \begin{pmatrix} R_0 & 0 \\ 0 & R_1 \end{pmatrix}$$

Assuming $m_1 > n$, the LS solution for the $\vec{y}_0$ subset of observations is

$$\begin{aligned} P_0 &= (H_0^T R_0^{-1} H_0)^{-1} \\ \hat{\vec{x}}_0 &= P_0 H_0^T R_0^{-1} \vec{y}_0 \end{aligned}$$

Recursive LS

A new covariance matrix P can be obtained in the case that additional observation equations corresponding to $\vec{y}_1$ are included

$$P_1^{-1} = \begin{pmatrix} H_0^T & H_1^T \end{pmatrix} \begin{pmatrix} R_0^{-1} & 0 \\ 0 & R_1^{-1} \end{pmatrix} \begin{pmatrix} H_0 \\ H_1 \end{pmatrix}$$

$$P_1^{-1} = H_0^T R_0^{-1} H_0 + H_1^T R_1^{-1} H_1$$

$$P_1 = (P_0^{-1} + H_1^T R_1^{-1} H_1^{-1})^{-1}$$

Recursive LS

Now, the LS solution $\hat{\vec{x}}_1$ gained under the consideration of the additional measurement group can be obtained

$$\hat{\vec{x}}_1 = P_1 \begin{pmatrix} H_0^T & H_1^T \end{pmatrix} \begin{pmatrix} R_0^{-1} & 0 \\ 0 & R_1^{-1} \end{pmatrix} \begin{pmatrix} \vec{y}_0 \\ \vec{y}_1 \end{pmatrix}$$

$$\hat{\vec{x}}_1 = P_1 \left(H_0^T R_0^{-1} \vec{y}_0 + H_1^T R_1^{-1} \vec{y}_1 \right)$$

Goal: To obtain a LS solution by not using a the old measurements $\vec{y}_0$

Conveniently

$$P_1 = \left(P_0^{-1} + H_1^T R_1^{-1} H_1^{-1} \right)$$

$$P_1 = P_0 - K_1 H_1 P_0$$

Recursive LS

$$P_1 = P_0 - K_1 H_1 P_0$$

$$K_1 = P_0 H_1^T \left(H_1 P_0 H_1^T + R_1 \right)^{-1}$$

$$\boxed{\hat{\vec{x}}_1 = \hat{\vec{x}}_0 + K_1 \left(\vec{y}_1 - H_1 \hat{\vec{x}}_0 \right)}$$

K_1 is called the gain matrix

Assuming a general iteration step for sequentially obtained data

$$\begin{aligned} K_j &= P_{j-1} H_j^T \left(H_j P_{j-1} H_j^T + R_j \right)^{-1} \\ \hat{\vec{x}}_j &= \hat{\vec{x}}_{j-1} + K_j \left(\vec{y}_j - H_j \hat{\vec{x}}_{j-1} \right) \\ P_j &= (I - K_j H_j) P_{j-1} \end{aligned}$$

Discrete Kalman filter

The Kalman filter is an estimation algorithm invented by Rudolf E. Kalman in 1960 and has been developed further by numerous authors since. It maintains real-time estimates of a number of parameters of a system such as its position and velocity [Groves].

Here, the Kalman filter is formulated as an extension of the recursive LS estimation.

The state vector (parameters) normally change over time.

Subscript k is introduced to denote the state update $\vec{x}(t_k) = \vec{x}_k$ by measurements of current epoch t_k that are usually available sequentially.

Discrete Kalman filter

The dynamic behavior of the system is modeled by describing the relationship of two consecutive state vectors. This is achieved by the linear (o linerized) function

$$\hat{\vec{x}}_k = \Phi_{k-1}\vec{x}_{k-1} + G\vec{w}_k$$

Φ_{k-1}	transition matrix $(n \times n)$
G	constant matrix (over the considered interval) $(m \times 1)$
$\vec{w}_k$	system noise vector $(n \times 1)$

$\vec{w}_k$ is assumed to follow a Gaussian distribution with zero mean and an $n \times n$ covariance matrix Q_{k-1}.

$$\vec{w}_k \sim N\left(0, Q_k\right)$$

Discrete Kalman filter

The dynamic model allows to incorporate the time-variant character of the state vector into the recursive LS.

A time update of the state vector (prediction) can be done even though no new measurements are available

$$\begin{aligned}\tilde{\vec{x}}_{k+1} &= \Phi_k \hat{\vec{x}}_k \\ \tilde{P}_{k+1} &= \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T\end{aligned}$$

When new measurements are available, a measurement update (correction) can be done

$$\begin{aligned}\hat{\vec{x}}_k &= \tilde{\vec{x}}_k + K_k \left(\vec{y}_k - H_k \tilde{\vec{x}}_k\right) \\ P_k &= \left(I - K_k H_k\right) \tilde{P}_k\end{aligned}$$

$$K_k = \tilde{P}_k H_k^T \left(H_k \tilde{P}_k H_k^T + R_k\right)^{-1}$$

Kalman filtering steps

Therefore, a standard implementation of the Kalman filtering comprises the following basic steps

1. Initialisation

$$\begin{aligned}\tilde{\vec{x}}_k &= \vec{x}_0 \\ \tilde{P}_k &= P_0\end{aligned}$$

2. Gain computation

$$K_k = \tilde{P}_k H_k^T \left(H_k \tilde{P}_k H_k^T + R_k \right)^{-1}$$

3. Prediction

$$\hat{\vec{x}}_k = \tilde{\vec{x}}_k + K_k \left(\vec{y}_k - H_k \tilde{\vec{x}}_k \right)$$

$$P_k = \left(I - K_k H_k \right) \tilde{P}_k$$

4. Correction

$$\begin{aligned}\tilde{\vec{x}}_{k+1} &= \Phi_k \hat{\vec{x}}_k \\ \tilde{P}_{k+1} &= \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T\end{aligned}$$

Kalman filter loops (Welch et al.)

Time Update ("Predict")

(1) Project the state ahead

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

(2) Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update ("Correct")

(1) Compute the Kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

(3) Update the error covariance

$$P_k = (I - K_k H)P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Kalman filter loops (Farrell)

Initialization	$\hat{\mathbf{x}}_0^- = E\langle \mathbf{x}_0 \rangle$ $\mathbf{P}_0^- = var(\mathbf{x}_0^-)$
Gain Calculation	$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top \left(\mathbf{R}_k + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top \right)^{-1}$
Measurement Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k \right)$
Covariance Update (choose one)	$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k] \mathbf{P}_k^-$ $\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k] \mathbf{P}_k^- [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k]^\top + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^\top$ $\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k \left(\mathbf{R}_k + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top \right) \mathbf{K}_k^\top$ $(\mathbf{P}_k^+)^{-1} = (\mathbf{P}_k^-)^{-1} + \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k$
Time Propagation	$\hat{\mathbf{x}}_{k+1}^- = \mathbf{\Phi}_k \hat{\mathbf{x}}_k^+ + \mathbf{G}_k \mathbf{u}_k$ $\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k \mathbf{P}_k^+ \mathbf{\Phi}_k^\top + \mathbf{Qd}_k$

Table 5.5: Discrete-time Kalman filter equations. Computation of the matrices $\mathbf{\Phi}_k$ and $\mathbf{Qd}_k$ is discussed in Section 4.7.1.

Kalman filter loops (Jeckeli)

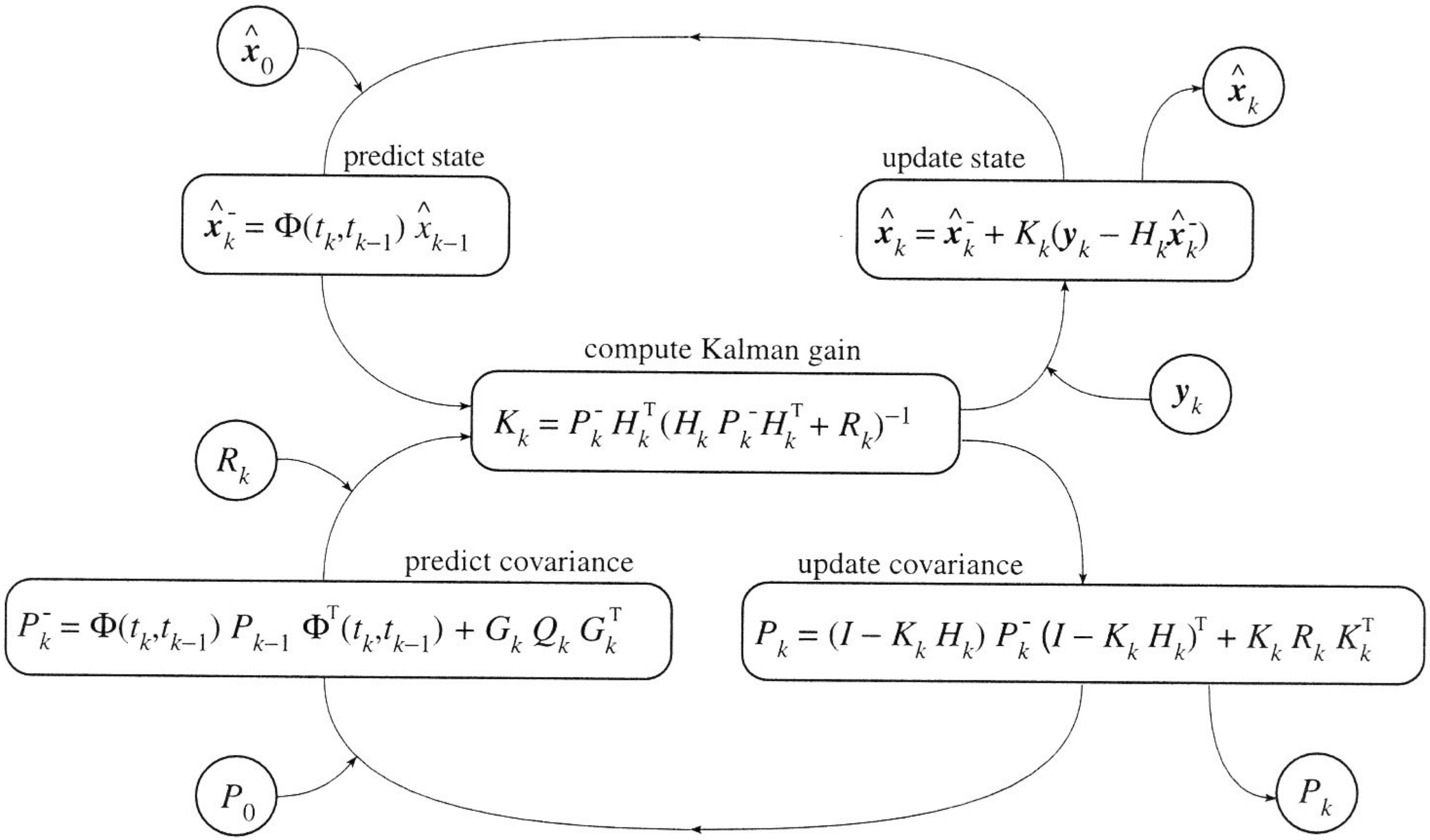


Figure 7.1: Kalman filter loops.

Kalman filter loops (Hofmann)

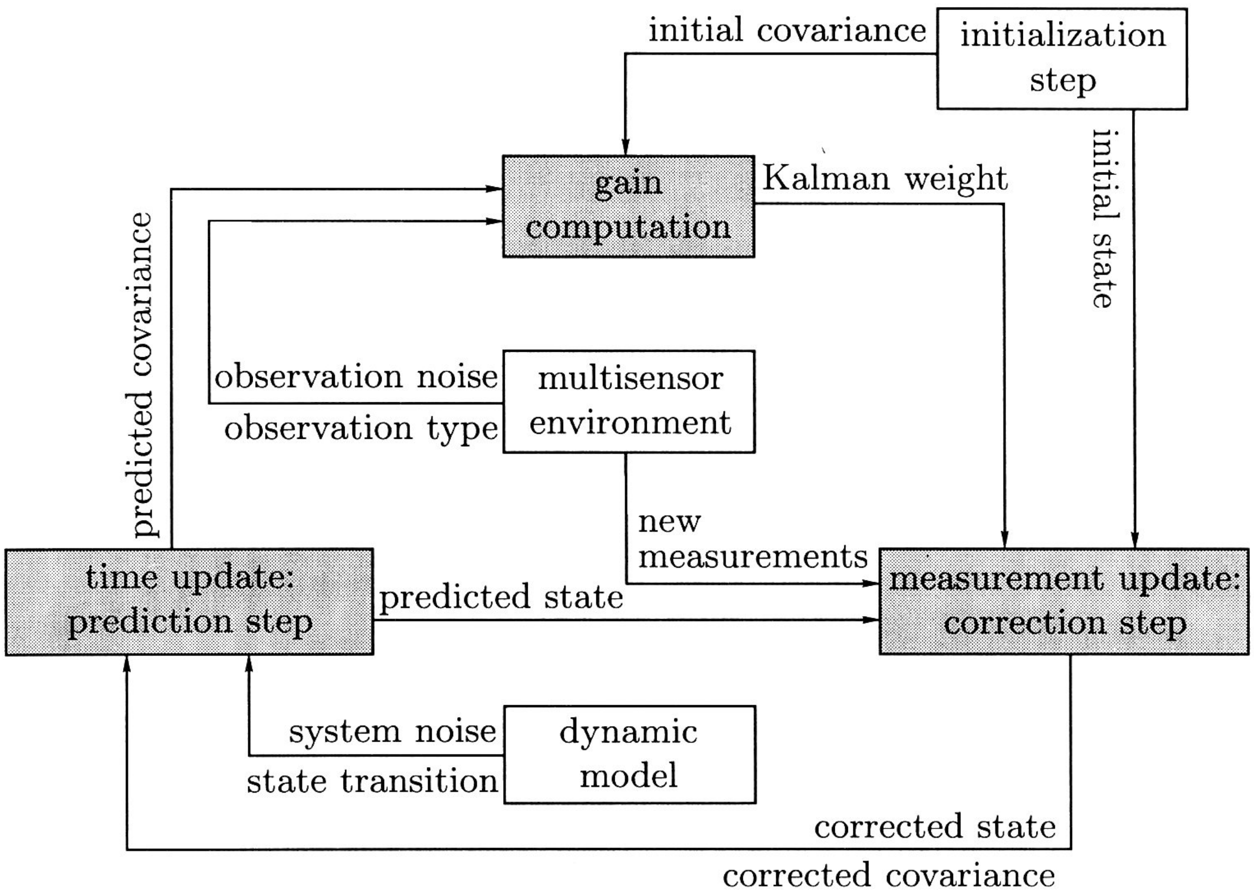


Fig. 3.13. Principle of Kalman filtering

Kalman filter loops (Groves)

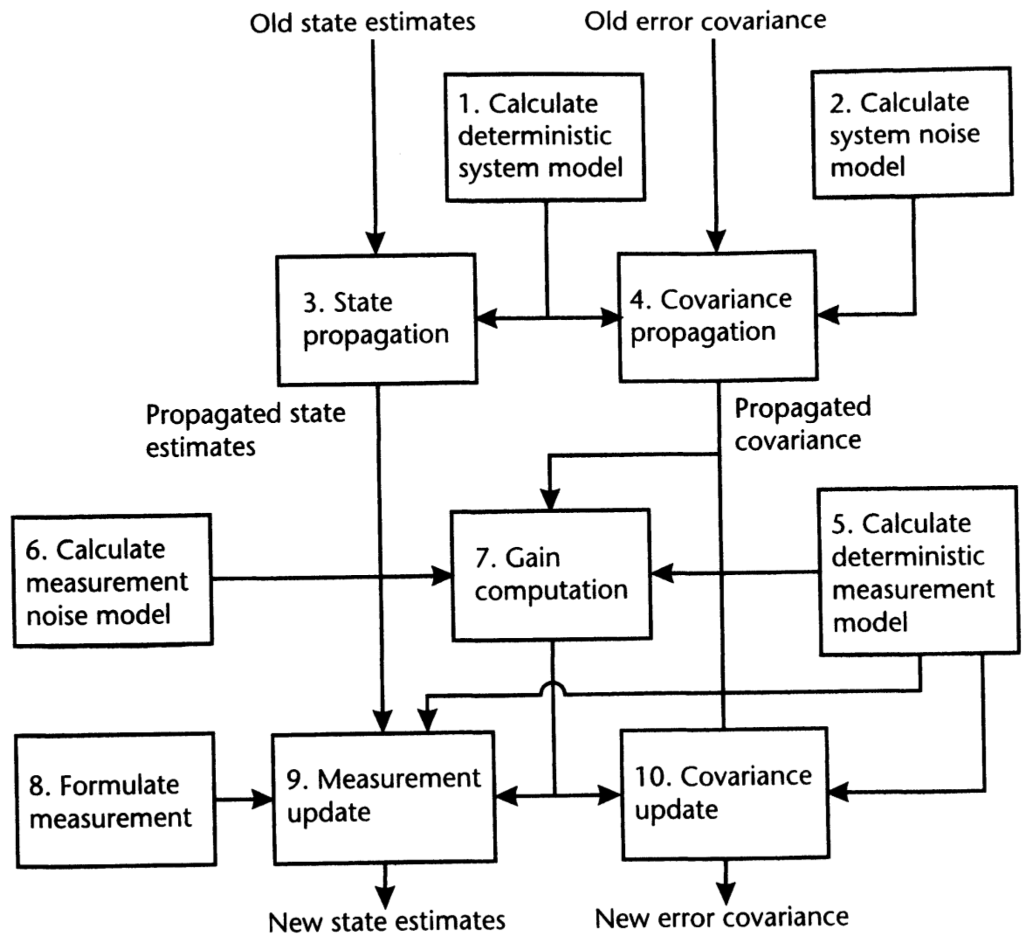


Figure 3.2 Kalman filter algorithm steps.

Numerical issues

The given formulation is a basic version of the Kalman filter.

The used notation has been chosen to be easily related to RLS estimation, but there is a range of possible modifications (augmented state vector, extended/closed-loop KF, continous KF, etc.)

The error covariance matrix P is by its definition symmetric, but when a KF is implemented on a computer, the precision is limited by the number of bits used to store and process each parameter.

Four techniques are common for maintaining the symmetry of P

Numerical issues

1. Instead of using the standard expression

$$P_k = (I - K_k H_k)\, \tilde{P}_k$$

use an equivalent formula such as the *Joseph form*, that is symmetric

$$P_k = (I - K_k H_k)\, \tilde{P}_k\, (I - K_k H_k) + K_k R_k K_k^T$$

2. Resymmetrise the covariance matrices at regular intervals using the equation

$$P_k = \frac{1}{2}\left(P + P^T\right)$$

3. Only calculate the main diagonal and upper triangular portion of P. This approach substantially decreases the memory and computational requirements.

4. Use stable factoring methods such as $P = UDU^T$

Numerical issues

Finally, each application requires specific state transition matrix $\Phi(t)$ as well as specific design matrix H.

Linear, first order systems of differential equations like

$$\dot{\vec{x}}(t) = F(t)\vec{x}(t) + G(t)\vec{w}(t)$$

are solved using state transition matrices $\Phi(t, t')$ that can be approximated by

$$\Phi(t, t') = I + F(t - t') + \frac{1}{2}(F(t - t'))^2 + \frac{1}{3}(F(t - t'))^3 + \cdots$$

as long as the time interval is sufficiently small and F can be considered constant or nearly so.

Example

Jekeli (Pag.155)

$$\boxed{\varepsilon^n = \begin{pmatrix}\psi_1^n & \psi_2^n & \psi_3^n & \delta\dot{\phi} & \delta\dot{\lambda} & \delta\dot{h} & \delta\phi & \delta\lambda & \delta h\end{pmatrix}^T.}$$

$$\delta(\Omega_{in}^n + \Omega_{ie}^n)$$
$$= \begin{pmatrix} 0 & \delta\dot{\lambda}\sin\phi + (\dot{\lambda} + 2\omega_e)\cos\phi\,\delta\phi & -\delta\dot{\phi} \\ -\delta\dot{\lambda}\sin\phi - (\dot{\lambda} + 2\omega_e)\cos\phi\,\delta\phi & 0 & -\delta\dot{\lambda}\cos\phi + (\dot{\lambda} + 2\omega_e)\sin\phi\,\delta\phi \\ \delta\dot{\phi} & \delta\dot{\lambda}\cos\phi - (\dot{\lambda} + 2\omega_e)\sin\phi\,\delta\phi & 0 \end{pmatrix}. \tag{5.60}$$

$$\boldsymbol{u} = \begin{pmatrix}\delta\boldsymbol{\omega}_{is}^s \\ \delta\boldsymbol{a}^s \\ \delta\bar{\boldsymbol{g}}^n\end{pmatrix} \qquad \delta\boldsymbol{a}^n = C_s^n\,\delta\boldsymbol{a}^s + \boldsymbol{a}^n \times \boldsymbol{\psi}^n. \qquad \delta\omega_{in}^n = \begin{pmatrix}\delta\dot{\lambda}\cos\phi - (\dot{\lambda} + \omega_e)\delta\phi\sin\phi \\ -\delta\dot{\phi} \\ -\delta\dot{\lambda}\sin\phi - (\dot{\lambda} + \omega_e)\delta\phi\cos\phi\end{pmatrix}.$$

$$\frac{d}{dt}\varepsilon^n = F^n\varepsilon^n + G^n\boldsymbol{u},$$

Example

$$F^n =$$

$$\begin{bmatrix}
0 & -\dot{\ell}_1 \sin\phi & \dot{\phi} & 0 & \cos\phi & 0 & -\dot{\ell}_1 \sin\phi & 0 & 0 \\
\dot{\ell}_1 \sin\phi & 0 & \dot{\ell}_1 \cos\phi & -1 & 0 & 0 & 0 & 0 & 0 \\
-\dot{\phi} & -\dot{\ell}_1 \cos\phi & 0 & 0 & -\sin\phi & 0 & -\dot{\ell}_1 \cos\phi & 0 & 0 \\
0 & \dfrac{-a_3^n}{r} & \dfrac{a_2^n}{r} & \dfrac{-2\dot{h}}{r} & -\dot{\ell}_1 \sin 2\phi & \dfrac{-2\dot{\phi}}{r} & \bar{\Gamma}_{11}^n - \dot{\lambda}\dot{\ell}_2 \cos 2\phi & \bar{\Gamma}_{12}^n \cos\phi & \dfrac{\ddot{\phi} + \frac{1}{2}\dot{\lambda}\dot{\ell}_2 \sin 2\phi + \bar{\Gamma}_{13}^n}{-r} \\
\dfrac{a_3^n}{r\cos\phi} & 0 & \dfrac{-a_1^n}{r\cos\phi} & 2\dot{\ell}_1 \tan\phi & 2\left(\dot{\phi}\tan\phi - \dfrac{\dot{h}}{r}\right) & \dfrac{-2\dot{\ell}_1}{r} & \begin{matrix} 2\dot{\ell}_1 \left(\dot{\phi} + \dfrac{\dot{h}\tan\phi}{r}\right) \\ + \ddot{\lambda}\tan\phi + \bar{\Gamma}_{21}^n \sec\phi \end{matrix} & \bar{\Gamma}_{22}^n & \dfrac{2\dot{\phi}\dot{\ell}_1 \tan\phi - \ddot{\lambda} - \dfrac{\bar{\Gamma}_{23}^n}{\cos\phi}}{r} \\
a_2^n & -a_1^n & 0 & 2r\dot{\phi} & 2r\dot{\ell}_1 \cos^2\phi & 0 & -r\dot{\lambda}\dot{\ell}_2 \sin 2\phi - r\bar{\Gamma}_{31}^n & -r\cos\phi\bar{\Gamma}_{32}^n & \dot{\phi}^2 + \dot{\lambda}\dot{\ell}_2 \cos^2\phi + \bar{\Gamma}_{33}^n \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}$$

$$G^n = \begin{pmatrix} -C_s^n & 0 & 0 \\ 0 & D^{-1}C_s^n & D^{-1} \\ 0 & 0 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} M+h & 0 & 0 \\ 0 & (N+h)\cos\phi & 0 \\ 0 & 0 & -1 \end{pmatrix} \qquad \frac{d}{dt}\begin{pmatrix} \delta\phi \\ \delta\lambda \\ \delta h \end{pmatrix} = \begin{pmatrix} \delta\dot{\phi} \\ \delta\dot{\lambda} \\ \delta\dot{h} \end{pmatrix}$$