CHAPTER 6

Alternate Kalman filter formulations

Our experiences with estimation and control applications engineers, however, indicates that they generally prefer the seemingly simpler Kalman filter algorithms for computer implementation and they dismiss reported instances of numerical failure.
—Gerald Bierman and Catherine Thornton [Bie77a]

In this chapter, we will look at some alternate ways of writing the Kalman filter equations. There are a number of mathematically equivalent ways of writing the Kalman filter equations. This can be confusing. You might read two different papers or books that present the Kalman filter equations, and the equations might look completely different. You may not know if one of the equations has a typographical error, or if they are mathematically equivalent. So you try to prove the equivalence of the two sets of equations only to arrive at a mathematical dead end, because it is not always easy to prove the equivalence of two sets of equations. This chapter derives some Kalman filter formulations that are different than (but mathematically equivalent to) the equations we derived in Chapter 5. This chapter also illustrates their advantages and disadvantages.

The first alternate formulation that we discuss is called the sequential Kalman filter, derived in Section 6.1. Sequential Kalman filtering allows for the implementation of the Kalman filter without matrix inversion. This can be a great benefit, especially in an embedded system that does not have matrix libraries, but it only makes sense if certain conditions are satisfied. The second formulation that we discuss is called information filtering, derived in Section 6.2. Information filtering propagates the inverse of the covariance matrix (i.e., P^{-1}) instead of P, and is computationally cheaper than Kalman filtering under certain conditions. The third formulation that we discuss is called square root filtering, derived in Section 6.3. Square root filtering effectively increases the precision of the Kalman filter, which can help prevent divergence and instability. However, this is at the cost of increased computational effort. The final formulation that we discuss is called U-D filtering, derived in Section 6.4. This is another way to implement square root filtering, which helps to prevent numerical difficulties in the implementation of the Kalman filter.

6.1 SEQUENTIAL KALMAN FILTERING

In this section, we derive the sequential Kalman filter. This is a way of implementing the Kalman filter without matrix inversion. This can be a great advantage, especially in an embedded system that may not have matrix routines. However, the use of sequential Kalman filtering only makes sense if certain conditions are satisfied, which we will discuss in this section.

Recall the Kalman filter measurement update formulas from Equation (5.16):

$$\begin{aligned}
y_k &= H_k x_k + v_k \\
K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\
\hat{x}_k^+ &= \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^-) \\
P_k^+ &= (I - K_k H_k) P_k^-
\end{aligned} \tag{6.1}$$

The computation of K_k requires the inversion of an $r \times r$ matrix, where r is the number of measurements. This is depicted in Figure 6.1.

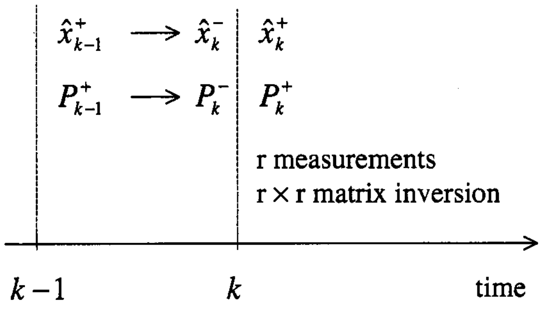
$\hat{x}_{k-1}^+ \longrightarrow \hat{x}_k^-$
$\hat{x}_k^+$
$P_{k-1}^+ \longrightarrow P_k^-$
P_k^+
r measurements
r × r matrix inversion
$k-1$
k
time

Figure 6.1 The measurement-update equation of the standard Kalman filter requires an $r \times r$ matrix inversion, where r is the number of measurements.

Suppose that instead of measuring y_k at time k, we obtain r separate measurements at time k. That is, we first measure $y_k(1)$, then $y_k(2)$, $\cdots$, and finally $y_k(r)$. We will use the shorthand notation y_{ik} for the ith element of the measurement vector y_k. Assume for now that R_k (the covariance of measurement y_k) is diagonal;

that is, R_k is given as

$$R_k = \begin{bmatrix} R_{1k} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R_{rk} \end{bmatrix} \tag{6.2}$$

We will also use the notation that H_{ik} is the ith row of H_k, and v_{ik} is the ith element of v_k. Then we obtain

$$\begin{aligned} y_{ik} &= H_{ik}x_k + v_{ik} \\ v_{ik} &\sim (0, R_{ik}) \end{aligned} \tag{6.3}$$

So instead of processing the measurements at time k as a vector, we will implement the Kalman filter measurement-update equation one measurement at a time. We use the notation that K_{ik} is the Kalman gain that is used to process the ith measurement at time k, $\hat{x}_{ik}^+$ is the optimal estimate after the ith measurement has been processed at time k, and P_{ik}^+ is the estimation-error covariance after the ith measurement at time k has been processed. We can see from these definitions that

$$\begin{aligned} \hat{x}_{0k}^+ &= \hat{x}_k^- \\ P_{0k}^+ &= P_k^- \end{aligned} \tag{6.4}$$

That is, $\hat{x}_{0k}^+$ is the estimate after zero measurements have been processed, so it is equal to the *a priori* estimate. Similarly, P_{0k}^+ is the estimation-error covariance after zero measurements have been processed, so it is equal to the *a priori* estimation-error covariance. The gain K_{ik} and covariance P_{ik}^+ are obtained from the normal Kalman filter measurement-update equations, with the understanding that they apply to the scalar measurement y_{ik}. For $i = 1, \cdots, r$ we have

$$\begin{aligned} K_{ik} &= P_{i-1,k}^+ H_{ik}^T (H_{ik} P_{i-1,k}^+ H_{ik}^T + R_{ik})^{-1} \\ \hat{x}_{ik}^+ &= \hat{x}_{i-1,k}^+ + K_{ik}(y_{ik} - H_{ik}\hat{x}_{i-1,k}^+) \\ P_{ik}^+ &= (I - K_{ik}H_{ik})P_{i-1,k}^+ \end{aligned} \tag{6.5}$$

After all r measurements are processed, we set $\hat{x}_k^+ = \hat{x}_{rk}^+$, and $P_k^+ = P_{rk}^+$, and we have our *a posteriori* estimate and error covariance at time k. The sequential Kalman filter does not require any matrix inversions because all of the expressions in Equation (6.5) are scalar operations. This process is depicted in Figure 6.2. The sequential Kalman filter can be summarized as follows.

The sequential Kalman filter

1. The system and measurement equations are given as

$$\begin{aligned} x_k &= F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \\ y_k &= H_k x_k + v_k \\ w_k &\sim (0, Q_k) \\ v_k &\sim (0, R_k) \end{aligned} \tag{6.6}$$

where w_k and v_k are uncorrelated white noise sequences. The measurement covariance R_k is a diagonal matrix given as

$$R_k = \text{diag}(R_{1k}, \cdots, R_{rk}) \tag{6.7}$$

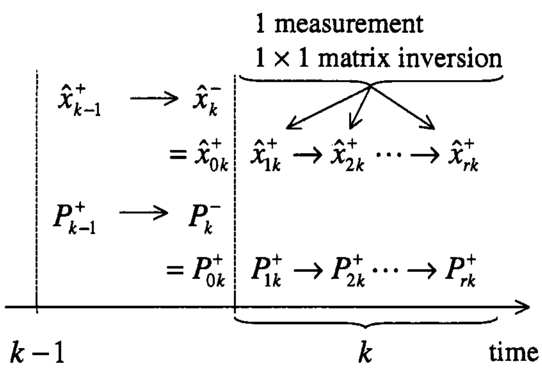
1 measurement
1×1 matrix inversion
$\hat{x}_{k-1}^{+} \longrightarrow \hat{x}_k^{-} = \hat{x}_{0k}^{+}$
$\hat{x}_{1k}^{+} \to \hat{x}_{2k}^{+} \cdots \to \hat{x}_{rk}^{+}$
$P_{k-1}^{+} \longrightarrow P_k^{-} = P_{0k}^{+}$
$P_{1k}^{+} \to P_{2k}^{+} \cdots \to P_{rk}^{+}$
$k-1$
k
time

Figure 6.2 The measurement update equation of the sequential Kalman filter requires r scalar divisions (where r is the number of measurements) because the measurements at each time step are processed sequentially. This is in contrast to the standard Kalman filter processing that is depicted in Figure 6.1.

2. The filter is initialized as

$$\begin{aligned} \hat{x}_0^+ &= E(x_0) \\ P_0^+ &= E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \end{aligned} \tag{6.8}$$

3. At each time step k, the time-update equations are given as

$$\begin{aligned} P_k^- &= F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} \\ \hat{x}_k^- &= F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1} \end{aligned} \tag{6.9}$$

   This is the same as the standard Kalman filter.

4. At each time step k, the measurement-update equations are given as follows.

   (a) Initialize the *a posteriori* estimate and covariance as

$$\begin{aligned} \hat{x}_{0k}^+ &= \hat{x}_k^- \\ P_{0k}^+ &= P_k^- \end{aligned} \tag{6.10}$$

   These are the *a posteriori* estimate and covariance at time k after zero measurements have been processed; that is, they are equal to the *a priori* estimate and covariance.

   (b) For $i = 1, \cdots, r$ (where r is the number of measurements), perform the following:

$$\begin{aligned} K_{ik} &= \frac{P_{i-1,k}^+H_{ik}^T}{H_{ik}P_{i-1,k}^+H_{ik}^T + R_{ik}} \\ &= \frac{P_{ik}^+H_{ik}^T}{R_{ik}} \\ \hat{x}_{ik}^+ &= \hat{x}_{i-1,k}^+ + K_{ik}(y_{ik} - H_{ik}\hat{x}_{i-1,k}^+) \\ P_{ik}^+ &= (I - K_{ik}H_{ik})P_{i-1,k}^+(I - K_{ik}H_{ik})^T + K_{ik}R_{ik}K_{ik}^T \end{aligned}$$

$$\begin{aligned} &= \left[(P_{i-1,k}^{+})^{-1} + H_{ik}^{T} H_{ik}/R_{ik}\right]^{-1} \\ &= (I - K_{ik} H_{ik}) P_{i-1,k}^{+} \end{aligned} \tag{6.11}$$

(c) Assign the *a posteriori* estimate and covariance as

$$\begin{aligned} \hat{x}_k^+ &= \hat{x}_{rk}^+ \\ P_k^+ &= P_{rk}^+ \end{aligned} \tag{6.12}$$

The development above assumes that the measurement-noise covariance R_k is diagonal. What if R_k is not diagonal? Suppose that $R_k = R$ is not diagonal, but it is a constant matrix. We perform a Jordan form decomposition of R by finding a matrix S such that

$$R = S\hat{R}S^{-1} \tag{6.13}$$

$\hat{R}$ is a diagonal matrix containing the eigenvalues of R, and S is an orthogonal matrix (i.e., $S^{-1} = S^T$) containing the eigenvectors of R. This decomposition is always possible if R is symmetric positive definite, as discussed in most linear systems books [Bay99, Che99, Kai00]. Now define a new measurement $\tilde{y}_k$ as

$$\begin{aligned} \tilde{y}_k &= S^{-1} y_k \\ &= S^{-1}(H_k x_k + v_k) \\ &= \tilde{H}_k x_k + \tilde{v}_k \end{aligned} \tag{6.14}$$

where $\tilde{H}_k$ and $\tilde{v}_k$ are defined by the above equation. The covariance of $\tilde{v}_k$ can be obtained as

$$\begin{aligned} E(\tilde{v}_k \tilde{v}_k^T) &= E(S^{-1} v_k v_k^T S^{-T}) \\ &= E(S^{-1} v_k v_k^T S) \\ &= S^{-1} E(v_k v_k^T) S \\ &= S^{-1} R S \\ &= \hat{R} \end{aligned} \tag{6.15}$$

So we have introduced a normalized measurement $\tilde{y}_k$ that has a diagonal noise covariance. Now we can implement the sequential Kalman filter equations, except that we use the measurement $\tilde{y}_k$ instead of y_k, the measurement matrix $\tilde{H}_k$ instead of H_k, and the measurement noise covariance $\hat{R}$.

Note that this procedure would not make sense if R were time-varying, because in that case we would have to perform a Jordan form decomposition at each step of the Kalman filter. That would be a lot of computational effort in order to avoid a matrix inversion. However, if R is constant and it is known before the implementation of the Kalman filter, then we can perform the Jordan form decomposition offline and use the sequential Kalman filter to our advantage.

In summary, it only makes sense to use the sequential Kalman filter if one of the following two conditions holds:

1. The measurement noise covariance R_k is diagonal
2. The measurement noise covariance R is a constant.

Finally, note that the term sequential filtering is sometimes used synonymously with the Kalman filter. That is, sequential is often used as a synonym for recursive [Buc68, Chapter 13], [Bro96]. This can cause some confusion in terminology. However, sequential filtering is usually used in the literature as we use it in this section; that is, sequential filtering is a filtering method that processes measurements one at a time (rather than processing the measurements as a whole vector). Sometimes, the standard Kalman filter is called the batch Kalman filter to distinguish it from the sequential Kalman filter.

■ EXAMPLE 6.1

The change x_k from one week to the next of an American football team's ranking is related to the team's performance against that week's opponent. The expected relationships between various normalized game measures y_{ik} and the team's ranking change at the kth week are given as

$$\begin{aligned} y_{1k} &= x_k + v_{1k} = \text{ point differential} \\ y_{2k} &= \frac{1}{5}x_k + v_{2k} = \text{ turnover differential} \\ y_{3k} &= \frac{1}{50}x_k + v_{3k} = \text{ yardage differential} \end{aligned} \tag{6.16}$$

where $v_{1k} \sim (0,2)$, $v_{2k} \sim (0,1)$, and $v_{3k} \sim (0,50)$. Before the first game of the season is played, it is expected that the team ranking will increase by one due to certain players having returned from injuries. The variance of this *a priori* estimate is 4. Uncertainty in ownership conditions is expected to decrease the team's ranking by 5% each week, with a variance of 2. The system can therefore be modeled as

$$\begin{aligned} x_{k+1} &= 0.95x_k + w_k \\ y_k &= \begin{bmatrix} 1 & 1/5 & 1/50 \end{bmatrix}^T x_k + v_k \\ w_k &\sim (0,Q) \quad Q = 2 \\ v_k &\sim (0,R) \quad R = \text{diag}(2,1,50) \\ \hat{x}_0^+ &= 1 \\ P_0^+ &= 4 \end{aligned} \tag{6.17}$$

Suppose that the team plays its first game and wins by six points, gains three more turnovers than its opponent, and is outgained by 100 yards. That is, $y_1 = \begin{bmatrix} 6 & 3 & -100 \end{bmatrix}^T$. The standard Kalman filter adjusts the team's ranking as follows:

$$\begin{aligned} P_1^- &= FP_0^+F^T + Q \\ &= 5.61 \\ \hat{x}_1^- &= 0.95\hat{x}_0^+ \\ &= 0.95 \\ K_1 &= P_1^-H^T(HP_1^-H^T + R)^{-1} \\ &= \begin{bmatrix} 0.6961 & 0.2785 & 0.0006 \end{bmatrix} \\ \hat{x}_1^+ &= \hat{x}_1^- + K_1(y_1 - H\hat{x}_1^-) \end{aligned}$$

$$
\begin{aligned}
&= 5.1922 \\
P_1^+ &= (I - K_1 H)P_1^- \\
&= 1.3923
\end{aligned}
\tag{6.18}
$$

The K_1 calculation requires the inversion of a 3×3 matrix. On the other hand, the sequential Kalman filter could be used to update the estimated team ranking as follows:

$$
\begin{aligned}
P_1^- &= FP_0^+F^T + Q \\
&= 5.61 \\
\hat{x}_1^- &= 0.95\hat{x}_0^+ \\
&= 0.95 \\
\hat{x}_{01}^+ &= \hat{x}_1^- \\
P_{01}^+ &= P_1^-
\end{aligned}
\tag{6.19}
$$

The first measurement is processed as follows:

$$
\begin{aligned}
K_{11} &= P_{01}^+H_1^T(H_1P_{01}^+H_1^T + R_{11})^{-1} \\
&= 0.7372 \\
\hat{x}_{11}^+ &= \hat{x}_{01}^+ + K_{11}(y_{11} - H_1\hat{x}_{01}^+) \\
&= 4.6728 \\
P_{11}^+ &= (I - K_{11}H_1)P_{01}^+ \\
&= 1.4744
\end{aligned}
\tag{6.20}
$$

The second measurement is processed as follows:

$$
\begin{aligned}
K_{21} &= P_{11}^+H_2^T(H_2P_{11}^+H_2^T + R_{22})^{-1} \\
&= 0.2785 \\
\hat{x}_{21}^+ &= \hat{x}_{11}^+ + K_{21}(y_{21} - H_2\hat{x}_{11}^+) \\
&= 5.2479 \\
P_{21}^+ &= (I - K_{21}H_2)P_{11}^+ \\
&= 1.3923
\end{aligned}
\tag{6.21}
$$

The third measurement is processed as follows:

$$
\begin{aligned}
K_{31} &= P_{21}^+H_3^T(H_3P_{21}^+H_3^T + R_{33})^{-1} \\
&= 0.0006 \\
\hat{x}_{31}^+ &= \hat{x}_{21}^+ + K_{31}(y_{33} - H_3\hat{x}_{21}^+) \\
&= 5.1922 \\
P_{31}^+ &= (I - K_{31}H_3)P_{21}^+ \\
&= 1.3923
\end{aligned}
\tag{6.22}
$$

The sequential Kalman filter requires three loops through the measurement update equations, but no matrix inversions are required.

$\nabla\nabla\nabla$

6.2 INFORMATION FILTERING

In this section, we discuss information filtering. This is an implementation of the Kalman filter that propagates the inverse of P rather than propagating P; that is, information filtering propagates the information matrix of the system. Recall that

$$P = E[(x - \hat{x})(x - \hat{x})^T] \tag{6.23}$$

That is, P represents the uncertainty in the state estimate. If P is "large" then we have a lot of uncertainty in our state estimate. In the limit as $P \to 0$ we have perfect knowledge of x, and as $P \to \infty$ we have zero knowledge of x. The information matrix is defined as

$$\mathcal{I} = P^{-1} \tag{6.24}$$

That is, $\mathcal{I}$ represents the certainty in the state estimate. If $\mathcal{I}$ is "large" then we have a lot of confidence in our state estimate. In the limit as $\mathcal{I} \to 0$ we have zero knowledge of x, and as $\mathcal{I} \to \infty$ we have perfect knowledge of x.

Recall from Equation (5.19) that the measurement update equation for P can be written as

$$(P_k^+)^{-1} = (P_k^-)^{-1} + H_k^T R_k^{-1} H_k \tag{6.25}$$

Substituting the definition of $\mathcal{I}$ into this equation gives

$$\mathcal{I}_k^+ = \mathcal{I}_k^- + H_k^T R_k^{-1} H_k \tag{6.26}$$

This gives the measurement-update equation for the information matrix. Recall from Equation (5.19) the time-update equation for P:

$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1} \tag{6.27}$$

This implies that

$$\mathcal{I}_k^- = [F_{k-1}(\mathcal{I}_{k-1}^+)^{-1} F_{k-1}^T + Q_{k-1}]^{-1} \tag{6.28}$$

Now we can use the matrix inversion lemma from Section 1.1.2, which we restate here:

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1} \tag{6.29}$$

If we make the identifications $A = Q_{k-1}$, $B = F_{k-1}$, $C = F_{k-1}^T$, and $D = \mathcal{I}_{k-1}^+$, then we can apply the matrix inversion lemma to Equation (6.28) to obtain

$$\mathcal{I}_k^- = Q_{k-1}^{-1} - Q_{k-1}^{-1} F_{k-1} (\mathcal{I}_{k-1}^+ + F_{k-1}^T Q_{k-1}^{-1} F_{k-1})^{-1} F_{k-1}^T Q_{k-1}^{-1} \tag{6.30}$$

This gives the time-update equation for the information matrix. The information filter can be summarized as follows.

The information filter

1. The dynamic system is given by the following equations:

$$\begin{aligned} x_k &= F_{k-1} x_{k-1} + G_{k-1} u_{k-1} + w_{k-1} \\ y_k &= H_k x_k + v_k \\ w_k &\sim (0, Q_k) \end{aligned}$$

$$\begin{aligned} v_k &\sim (0, R_k) \\ E(w_k w_j^T) &= Q_k \delta_{k-j} \\ E(v_k v_j^T) &= R_k \delta_{k-j} \\ E(w_k v_k^T) &= 0 \end{aligned} \tag{6.31}$$

2. The Kalman filter is initialized as follows:

$$\begin{aligned} \hat{x}_0^+ &= E(x_0) \\ \mathcal{I}_0^+ &= \left\{E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]\right\}^{-1} \end{aligned} \tag{6.32}$$

3. The information filter is given by the following equations, which are computed for each time step $k = 1, 2, \cdots$:

$$\begin{aligned} \mathcal{I}_k^- &= Q_{k-1}^{-1} - Q_{k-1}^{-1} F_{k-1} (\mathcal{I}_{k-1}^+ + F_{k-1}^T Q_{k-1}^{-1} F_{k-1})^{-1} F_{k-1}^T Q_{k-1}^{-1} \\ \mathcal{I}_k^+ &= \mathcal{I}_k^- + H_k^T R_k^{-1} H_k \\ K_k &= (\mathcal{I}_k^+)^{-1} H_k^T R_k^{-1} \\ \hat{x}_k^- &= F_{k-1} \hat{x}_{k-1}^+ + G_{k-1} u_{k-1} \\ \hat{x}_k^+ &= \hat{x}_k^- + K_k (y_k - H_k \hat{x}_k^-) \end{aligned} \tag{6.33}$$

The standard Kalman filter equations require the inversion of an $r \times r$ matrix, where r is the number of measurements. The information filter equations require at least a couple of $n \times n$ matrix inversions, where n is the number of states. Therefore, if $r \gg n$ (i.e., we have significantly more measurements than states) it may be computationally more efficient to use the information filter. It could be argued that since the Kalman gain is given as

$$K_k = P_k^+ H_k^T R_k^{-1} \tag{6.34}$$

we have to perform and $r \times r$ matrix inversion on R_k anyway, whether we use the standard Kalman filter or the information filter. But if R_k is constant, then we could invert it as part of the initialization process, so the Kalman gain equation may not require this $r \times r$ matrix inversion after all. The same thinking also applies to the inversion of Q_{k-1}.

If the initial uncertainty is infinite, we cannot numerically set $P_0^+ = \infty$, but we can numerically set $\mathcal{I}_0^+ = 0$. This makes the information filter more mathematically precise for the zero initial certainty case. However, if the initial uncertainty is zero (i.e., we have perfect knowledge of x_0), we can numerically set $P_0^+ = 0$, but we cannot numerically set $\mathcal{I}_0^+ = \infty$. This makes the standard Kalman filter more mathematically precise for the zero initial uncertainty case

■ EXAMPLE 6.2

The information filter can be used to solve the American football team ranking problem of Example 6.1. The information filter equations are given as

$$\begin{aligned} \mathcal{I}_1^- &= Q^{-1} - Q^{-1} F (\mathcal{I}_0^+ + F^T Q^{-1} F)^{-1} F^T Q^{-1} \\ &= 0.1783 \end{aligned}$$

$$\begin{aligned}
\mathcal{I}_1^+ &= \mathcal{I}_1^- + H^T R^{-1} H \\
&= 0.7183 \\
K_1 &= (\mathcal{I}_1^+)^{-1} H^T R^{-1} \\
&= \begin{bmatrix} 0.6961 & 0.2785 & 0.0006 \end{bmatrix} \\
\hat{x}_1^- &= F \hat{x}_0^+ \\
&= 0.95 \\
\hat{x}_1^+ &= \hat{x}_1^- + K_1 (y_1 - H \hat{x}_1^-) \\
&= 5.1922
\end{aligned} \tag{6.35}$$

The information filter requires the inversion of Q and R, but in many applications these matrices are constant and can therefore be inverted offline. The only other matrix inversions are in the $\mathcal{I}_k^-$ and K_k equations. These inversions are scalar in this example because there is only one state in this example.

$\nabla\nabla\nabla$

6.3 SQUARE ROOT FILTERING

The early days of Kalman filtering in the 1960s saw a lot of promise and successful applications in the aerospace industry and in NASA's space program, but sometimes problems arose in implementation. Many of the problems that were encountered were due to numerical difficulties. The Riccati equation solution P_k should theoretically always be a symmetric positive semidefinite matrix, but numerical problems in computer implementations sometimes led to P_k matrices that became indefinite or nonsymmetric. This was often because of the short word lengths in the computers of the 1960s [Sch81]. Numerical problems may arise in cases in which some elements of the state-vector x are estimated to much greater precision than other elements of x. This could be because of discrepancies in the units of the state-vector elements. For example, one state might be in units of miles and can be estimated to within 0.01 miles, whereas a second state might be in units of cm/s and can be estimated to within 10 cm/s. The covariance for the first state would be on the order of 10^{-4}, whereas the covariance for the second state would be on the order of 10^2. This led to a lot of research during the 1960s that was related to numerical implementations.

Square root filtering is a way to mathematically increase the precision of the Kalman filter when hardware precision is not available. Perhaps the first square root algorithm was developed by James Potter for NASA's Apollo space program [Bat64]. Although Potter's algorithm was limited to zero process noise and scalar measurements, its success led to a lot of additional square root research in the following years. Potter's algorithm was extended to handle process noise in [And68, Dye69], and was generalized in two different ways to handle vector measurements in [Bel67, And68]. Paul Kaminski gives a good review of square root filtering developments during the first decade of the Kalman filter [Kam71].

Now that computers have become so much more capable, we do not have to worry about numerical problems as often. Nevertheless, numerical issues still arise in finite-word-length implementations of algorithms, especially in embedded systems. In this section, we will discuss the square root filter, which was developed in order to effectively increase the numerical precision of the Kalman filter and hence mitigate numerical difficulties in implementations. However, this improved performance is at the cost of greater computational effort. First, we will review the concept of the condition number of a matrix, then we will derive the square root version of the time update equation, and finally we will derive the square root version of the measurement update equations. Section 8.3.3 contains a discussion of square root filtering for the continuous-time Kalman filter.

6.3.1 Condition number

Recall the definition of the singular values of a matrix. An $n \times n$ matrix P has n singular values σ, given as

$$\begin{aligned} \sigma^2(P) &= \lambda(P^T P) \\ &= \lambda(PP^T) \end{aligned} \tag{6.36}$$

The matrix $P^T P$ is symmetric, and the eigenvalues of a symmetric matrix are always real and nonnegative, so the singular values of a matrix are always real and nonnegative. The matrix P is nonsingular (invertible) if and only if all of its singular values are positive. The condition number of a matrix is defined as

$$\begin{aligned} \kappa(P) &= \frac{\sigma_{\max}(P)}{\sigma_{\min}(P)} \\ &\geq 1 \end{aligned} \tag{6.37}$$

Note that some authors use alternate definitions for condition number; for example, some authors define the condition number of a matrix as the square of the above definition.[1] As $\kappa(P) \to \infty$, the matrix P is said to be poorly conditioned or ill conditioned, and P approaches a singular matrix. In the implementation of a Kalman filter, the error covariance matrix P should always be positive definite because $P = E[(x - \hat{x})(x - \hat{x})^T]$. We use the standard notation

$$P > 0 \tag{6.38}$$

to indicate that P is positive definite. This is equivalent to saying that P is invertible, which is equivalent to saying that all of the eigenvalues of P are greater than zero. But suppose in our Kalman filter that some elements of x are estimated to much greater precision than other elements of x. For example, suppose that

$$P = \begin{bmatrix} 10^6 & 0 \\ 0 & 10^{-6} \end{bmatrix} \tag{6.39}$$

This means that our estimate of x_1 has a standard deviation of 10^3 and our estimate of x_2 has a standard deviation of 10^{-3}. This could be due to drastically different units in x_1 and x_2, or it could be simply that x_1 is much more observable that x_2. The singular values of a diagonal matrix are the magnitudes of the diagonal elements, which are 10^6 and 10^{-6}. In other words,

$$\kappa(P) = 10^{12} \tag{6.40}$$

[1]In MATLAB the COND function can be used to find the condition number of a matrix.

This is a pretty large condition number, which means that the P matrix might look like a singular matrix to a digital computer. For example, if we have a fixed-point computer with 10 decimal digits of precision and the 10^6 term is represented correctly in the computer, then the 10^{-6} term will be represented as a zero in the computer. Mathematically, P is nonsingular, but computationally P is singular.

The square root filter is based on the idea of finding an S matrix such that $P = SS^T$. The S matrix is then called a square root of P. Note that the definition of the square root of P is *not* that $P = S^2$, but that $P = SS^T$. Also note that this definition of the matrix square root is not standard. Some books and papers defined the matrix square root as $P = S^2$, others define it as $P = S^TS$, and others define it as $P = SS^T$. This latter definition is the one that we will use in this book. If P is symmetric positive definite then it always has a square root [Gol89, Moo00]. The square root of a matrix may not be unique; that is, there may be more than one solution for S in the equation $P = SS^T$. (This is analogous to the scalar square root, which is usually not unique. For example, the number 1 has two square roots; $+1$ and -1.) Also note that SS^T will always be symmetric positive semidefinite no matter what the value of the S matrix. Whereas numerical difficulties might cause P to become nonsymmetric or indefinite in the Kalman filter equations, numerical difficulties can never cause SS^T to become nonsymmetric or indefinite.

Matrix square root algorithms were first given by the French military officer Andre Cholesky (1875-1918) and the Polish astronomer Tadeusz Banachiewicz (1882-1954) [Fad59]. An interesting biography of Cholesky is given in the appendix of [Mai84].

The following algorithm computes an S matrix such that $P = SS^T$ for an $n \times n$ matrix P.

```
The Cholesky Matrix Square Root Algorithm {
    For $i = 1, \cdots, n$
    {
        $S_{ii} = \sqrt{P_{ii} - \sum_{j=1}^{i-1} S_{ij}^2}$
        For $j = 1, \cdots, n$
        {
            $S_{ji} = 0 \quad j < i$
            $S_{ji} = \frac{1}{S_{ii}} \left( P_{ji} - \sum_{k=1}^{i-1} S_{jk} S_{ik} \right) \quad j > i$
        }
    }
}
```

This is called Cholesky factorization and results in a matrix S such that $P = SS^T$. The matrix S is referred to as the Cholesky triangle because it is a lower triangular matrix. However, the algorithm only works if P is symmetric positive definite. If P is not symmetric positive definite, then it may or may not have a square root.[2]

In the following example we illustrate the application of Cholesky factorization.

[2]The MATLAB function CHOL outputs the transpose of the Cholesky triangle that is computed above.

EXAMPLE 6.3

This example is taken from [Kam71]. Suppose we have a P matrix given as

$$P = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 2 \\ 3 & 2 & 14 \end{bmatrix} \tag{6.41}$$

The Cholesky factorization algorithm tells us that, for $i = 1$,

$$\begin{aligned} S_{11} &= \sqrt{P_{11}} \\ &= 1 \\ S_{21} &= \frac{1}{S_{11}}(P_{21}) \\ &= 2 \\ S_{31} &= \frac{1}{S_{11}}(P_{31}) \\ &= 3 \end{aligned} \tag{6.42}$$

For $i = 2$, the algorithm tells us that

$$\begin{aligned} S_{22} &= \sqrt{P_{22} - \sum_{j=1}^{1} S_{2j}^2} \\ &= 2 \\ S_{12} &= 0 \\ S_{32} &= \frac{1}{S_{22}}\left(P_{32} - \sum_{k=1}^{1} S_{3k}S_{2k}\right) \\ &= -2 \end{aligned} \tag{6.43}$$

For $i = 3$, the algorithm tells us that

$$\begin{aligned} S_{33} &= \sqrt{P_{33} - \sum_{j=1}^{2} S_{3j}^2} \\ &= 1 \\ S_{13} &= 0 \\ S_{23} &= 0 \end{aligned} \tag{6.44}$$

So we obtain

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & -2 & 1 \end{bmatrix} \tag{6.45}$$

and it can be verified that $P = SS^T$.

$\nabla\nabla\nabla$

After defining S as the square root of P in the Kalman filter, we will propagate S instead of P. This requires more computational effort but it doubles the precision of the filter and helps prevent numerical problems. The singular values σ of P are given as

$$\begin{aligned} \sigma^2(P) &= \lambda(P^TP) \\ &= \lambda(SS^TSS^T) \end{aligned} \tag{6.46}$$

The singular values of S are given as

$$\sigma^2(S) = \lambda(SS^T) \tag{6.47}$$

Recall that for a general matrix A we have $\lambda(A^2) = \lambda^2(A)$. Therefore, we see from the above equations that

$$\begin{aligned} \sigma^2(P) &= \left[\sigma^2(S)\right]^2 \\ \frac{\sigma_{\max}(P)}{\sigma_{\min}(P)} &= \frac{\sigma^2_{\max}(S)}{\sigma^2_{\min}(S)} \\ \kappa(P) &= \kappa^2(S) \end{aligned} \tag{6.48}$$

That is, the condition number of P is the square of the condition number of S. For example, consider the P matrix given earlier in this section:

$$\begin{aligned} P &= \begin{bmatrix} 10^6 & 0 \\ 0 & 10^{-6} \end{bmatrix} \\ \kappa(P) &= 10^{12} \end{aligned} \tag{6.49}$$

The square root of this matrix and its condition number are

$$\begin{aligned} S &= \begin{bmatrix} 10^3 & 0 \\ 0 & 10^{-3} \end{bmatrix} \\ \kappa(S) &= 10^6 \end{aligned} \tag{6.50}$$

The condition number of P is 10^{12}, but the condition number of the square root of P is only 10^6. Square root filtering uses this idea to provide twice the precision of the standard Kalman filter. Instead of propagating P, we propagate the square root of P.

6.3.2 The square root time-update equation

Suppose we have an n-state discrete LTI system given as

$$\begin{aligned} x_k &= F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \\ E(w_kw_k^T) &= Q_k \end{aligned} \tag{6.51}$$

The *a priori* error covariance matrix of the Kalman filter is P_k^-, and its square root is S_k^-. The *a posteriori* error covariance matrix is P_k^+, and its square root is S_k^+. Suppose that we can find an orthogonal $2n \times 2n$ matrix T such that

$$\begin{aligned} \begin{bmatrix} (S_k^-)^T \\ 0 \end{bmatrix} &= T \begin{bmatrix} (S_{k-1}^+)^TF_{k-1}^T \\ Q_{k-1}^{T/2} \end{bmatrix} \\ &= \begin{bmatrix} T_1 & T_2 \end{bmatrix} \begin{bmatrix} (S_{k-1}^+)^TF_{k-1}^T \\ Q_{k-1}^{T/2} \end{bmatrix} \end{aligned} \tag{6.52}$$

Since T is orthogonal we see that

$$\begin{aligned} T^T T &= \begin{bmatrix} T_1^T \\ T_2^T \end{bmatrix} \begin{bmatrix} T_1 & T_2 \end{bmatrix} \\ &= \begin{bmatrix} T_1^T T_1 & T_1^T T_2 \\ T_2^T T_1 & T_2^T T_2 \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \end{aligned} \tag{6.53}$$

where T_1 and T_2 are both $n \times n$ matrices. We see from the above that

$$\begin{aligned} T_1^T T_2 = T_2^T T_1 &= 0 \\ T_1^T T_1 = T_2^T T_2 &= I \end{aligned} \tag{6.54}$$

Now note that we can use Equation (6.52) to write

$$\begin{bmatrix} S_k^- & 0 \end{bmatrix} \begin{bmatrix} (S_k^-)^T \\ 0 \end{bmatrix} = \left[T_1 (S_{k-1}^+)^T F_{k-1}^T + T_2 Q_{k-1}^{T/2} \right]^T \left[\cdots \right] \tag{6.55}$$

We can use this equation, along with Equation (6.54), to write

$$\begin{aligned} S_k^- (S_k^-)^T &= F_{k-1} S_{k-1}^+ T_1^T T_1 (S_{k-1}^+)^T F_{k-1}^T + Q_{k-1}^{1/2} T_2^T T_2 Q_{k-1}^{T/2} \\ &= F_{k-1} S_{k-1}^+ (S_{k-1}^+)^T F_{k-1}^T + Q_{k-1}^{1/2} Q_{k-1}^{T/2} \end{aligned} \tag{6.56}$$

If S_{k-1}^+ is the square root of P_{k-1}^+, this implies that

$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1} \tag{6.57}$$

which is exactly the time-update equation for P_k that is required in the Kalman filter, as shown in Equation (5.19). So if we can find an orthogonal $2n \times 2n$ matrix T such that

$$T \begin{bmatrix} (S_{k-1}^+)^T F_{k-1}^T \\ Q_{k-1}^{T/2} \end{bmatrix} = \begin{bmatrix} n \times n \text{ matrix} \\ 0 \end{bmatrix} \tag{6.58}$$

then the $n \times n$ matrix in the upper half of the matrix on the right side is equal to $(S_k^-)^T$. This assumes that $(S_{k-1}^+)^T$ is available from a square root measurement update equation, which we will discuss in the following two subsections. The square root time update equation above is mathematically equivalent to the original Kalman filter time update equation for P, but the update equation is used to update S instead of P.

As we noted above, the square root of P_k^- is not unique, so different algorithms for solving Equation (6.58) will result in different T and $(S_k^-)^T$ matrices. We can use various methods from numerical linear algebra to find the orthogonal $2n \times 2n$ matrix T and the resulting square root matrix S_k^- (e.g., Householder, Gram–Schmidt, modified Gram–Schmidt, or Givens transformations) [Hor85, Gol89, Str90, Moo00]. A couple of these algorithms are discussed in Section 6.3.5.

■ EXAMPLE 6.4

Suppose that at time $(k-1)$ our Kalman filter has a system matrix, process noise covariance, and *a posteriori* estimation covariance square root equal to

$$\begin{aligned} F_{k-1} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \\ Q_{k-1} &= \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \\ S_{k-1}^+ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{6.59}$$

It can be verified that the square root of Q_{k-1} (so that $Q_{k-1}^{1/2} Q_{k-1}^{T/2} = Q_{k-1}$) is given by

$$Q_{k-1}^{1/2} = \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix} \tag{6.60}$$

Equation (6.58) can be solved as

$$\begin{aligned} T \begin{bmatrix} (S_{k-1}^+)^T F_{k-1}^T \\ Q_{k-1}^{T/2} \end{bmatrix} &= \begin{bmatrix} (S_k^-)^T \\ 0 \end{bmatrix} \\ \frac{1}{\sqrt{10}} \begin{bmatrix} \sqrt{5} & \sqrt{5} & 0 & 0 \\ 1 & -1 & 2 & 2 \\ -2 & -2 & 1 & 1 \\ 0 & 0 & -\sqrt{5} & \sqrt{5} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & -1 \\ 0 & -1 \end{bmatrix} &= \frac{1}{\sqrt{10}} \begin{bmatrix} \sqrt{20} & \sqrt{5} \\ 0 & -5 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{6.61}$$

As mentioned earlier, algorithms for performing this computation will be discussed in Section 6.3.5. The upper-right square matrix on the right side of the above equation is equal to $(S_k^-)^T$, so this shows that the square root of the *a priori* estimation covariance at time k is given as

$$S_k^- = \frac{1}{\sqrt{10}} \begin{bmatrix} \sqrt{20} & 0 \\ \sqrt{5} & -5 \end{bmatrix} \tag{6.62}$$

From this it can be inferred that the *a priori* estimation covariance at time k is given as

$$\begin{aligned} P_k^- &= S_k^- (S_k^-)^T \\ &= \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \end{aligned} \tag{6.63}$$

Indeed, a straightforward implementation of the time-update equation for the estimation-error covariance gives

$$\begin{aligned} P_k^- &= F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1} \\ &= \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \end{aligned} \tag{6.64}$$

which confirms our square root results. However, the square root time update has essentially twice the precision of the standard time-update equation.

∇∇∇

6.3.3 Potter's square root measurement-update equation

The square root measurement-update equation discussed here is based on James Potter's algorithm, which was developed for NASA's Apollo space program [Bat64, Kam71] and modified by Angus Andrews to handle vector measurements [And68]. Recall from Equation (5.19) that the measurement update equation for the estimation covariance is given as

$$P_k^+ = (I - K_k H_k) P_k^- \tag{6.65}$$

We can process the measurements one at a time using the sequential Kalman filter of Section 6.1. That is, first we initialize $P_{0k}^+ = P_k^-$. Then, for $i = 1, \cdots, r$ (where r is the number of measurements), we compute

$$\begin{aligned} K_{ik} &= \frac{P_{i-1,k}^+ H_{ik}^T}{H_{ik} P_{i-1,k}^+ H_{ik}^T + R_{ik}} \\ P_{ik}^+ &= (I - K_{ik} H_{ik}) P_{i-1,k}^+ \end{aligned} \tag{6.66}$$

where H_{ik} is the ith row of H_k and R_{ik} is the variance of the ith measurement. (We are assuming here, as in Section 6.1, that R_k is diagonal.) Suppose we have the square root of $P_{i-1,k}^+$ so that $P_{i-1,k}^+ = S_{i-1,k}^+ S_{i-1,k}^{+T}$. Then K_{ik} can be written as

$$K_{ik} = \frac{S_{i-1,k}^+ S_{i-1,k}^{+T} H_{ik}^T}{H_{ik} S_{i-1,k}^+ S_{i-1,k}^{+T} H_{ik}^T + R_{ik}} \tag{6.67}$$

and P_{ik}^+ can be written as

$$\begin{aligned} P_{ik}^+ &= \left(I - \frac{S_{i-1,k}^+ S_{i-1,k}^{+T} H_{ik}^T H_{ik}}{H_{ik} S_{i-1,k}^+ S_{i-1,k}^{+T} H_{ik}^T + R_{ik}} \right) S_{i-1,k}^+ S_{i-1,k}^{+T} \\ &= S_{i-1,k}^+ (I - a\phi\phi^T) S_{i-1,k}^{+T} \end{aligned} \tag{6.68}$$

where ϕ and a are defined as

$$\begin{aligned} \phi &= S_{i-1,k}^{+T} H_{ik}^T \\ a &= \frac{1}{\phi^T \phi + R_{ik}} \end{aligned} \tag{6.69}$$

It can be shown (see Problem 6.9) that

$$I - a\phi\phi^T = (I - a\gamma\phi\phi^T)^2 \tag{6.70}$$

where γ is given as

$$\gamma = \frac{1}{1 \pm \sqrt{a R_{ik}}} \tag{6.71}$$

Either the plus or minus sign can be used in the computation of γ. Comparing Equations (6.68) and (6.70) shows that

$$S_{ik}^+ = S_{i-1,k}^+ (I - a\gamma\phi\phi^T) \tag{6.72}$$

This results in a square root measurement-update algorithm that can be summarized as follows.

Potter's square root measurement-update algorithm

1. After the *a priori* covariance square root S_k^- and the *a priori* state estimate $\hat{x}_k^-$ have been computed, initialize

$$\begin{aligned} \hat{x}_{0k}^+ &= \hat{x}_k^- \\ S_{0k}^+ &= S_k^- \end{aligned} \tag{6.73}$$

2. For $i = 1, \cdots, r$ (where r is the number of measurements), perform the following.

   (a) Define H_{ik} as the ith row of H_k, y_{ik} as the ith element of y_k, and R_{ik} as the variance of the ith measurement (assuming that R_k is diagonal).

   (b) Perform the following to find the square root of the covariance after the ith measurement has been processed:

$$\begin{aligned} \phi_i &= S_{i-1,k}^{+T} H_{ik}^T \\ a_i &= \frac{1}{\phi_i^T \phi_i + R_{ik}} \\ \gamma_i &= \frac{1}{1 \pm \sqrt{a_i R_{ik}}} \\ S_{ik}^+ &= S_{i-1,k}^+ (I - a_i \gamma_i \phi_i \phi_i^T) \end{aligned} \tag{6.74}$$

   (c) Compute the Kalman gain for the ith measurement as

$$K_{ik} = a_i S_{ik}^+ \phi_i \tag{6.75}$$

   (d) Compute the state estimate update due to the ith measurement as

$$\hat{x}_{ik}^+ = \hat{x}_{i-1,k}^+ + K_{ik}(y_{ik} - H_{ik}\hat{x}_{i-1,k}^+) \tag{6.76}$$

3. Set the *a posteriori* covariance square root and the *a posteriori* state estimate as

$$\begin{aligned} S_k^+ &= S_{rk}^+ \\ \hat{x}_k^+ &= \hat{x}_{rk}^+ \end{aligned} \tag{6.77}$$

Although square root filtering improves the numerical characteristics of the Kalman filter, it also increases computational requirements. Efforts to make square root filtering more efficient are reported in [Car73, Tho77, Tap80].

■ **EXAMPLE 6.5**

This example is based on [Kam71]. Suppose that we have an LTI system with

$$\begin{aligned} P_k^- &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ H &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\ F &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ Q &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{6.78}$$

If we had an infinite-precision computer, the exact Kalman gain and *a posteriori* covariance at time k would be given by

$$\begin{aligned} K_k &= P_k^- H^T (H P_k^- H^T + R)^{-1} \\ &= \begin{bmatrix} \frac{1}{1+R} \\ 0 \end{bmatrix} \\ P_k^+ &= (I - K_k H) P_k^- \\ &= \begin{bmatrix} \frac{R}{1+R} & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{6.79}$$

The *a priori* covariance and Kalman gain at the next time step $(k+1)$ would be given by

$$\begin{aligned} P_{k+1}^- &= F P_k^+ F^T + Q \\ &= \begin{bmatrix} \frac{R}{1+R} & 0 \\ 0 & 1 \end{bmatrix} \\ K_{k+1} &= P_{k+1}^- H^T (H P_{k+1}^- H^T + R)^{-1} \\ &= \begin{bmatrix} \frac{1}{2+R} \\ 0 \end{bmatrix} \end{aligned} \tag{6.80}$$

Now consider implementation in a finite precision digital computer. Suppose that the measurement covariance $R \ll 1$. The covariance R is such a tiny number that because of rounding in the computer, $1 + R = 1$, but $1 + \sqrt{R} > 1$. The rounded values of the Kalman gain and *a posteriori* covariance at time k would be given by

$$\begin{aligned} K_k &= \begin{bmatrix} \frac{1}{1+R} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ P_k^+ &= (I - K_k H) P_k^- \\ &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{6.81}$$

Note that P_k^+ has become singular because of the numerical limitations of the computer. The rounded values of the *a priori* covariance and Kalman gain at the next time step $(k+1)$ would be given by

$$\begin{aligned} P_{k+1}^- &= F P_k^+ F^T + Q \\ &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ K_{k+1} &= P_{k+1}^- H^T (H P_{k+1}^- H^T + R)^{-1} \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{6.82}$$

The numerical limitations of the computer have resulted in a zero Kalman gain, whereas the infinite-precision Kalman gain as given in Equation (6.80) is about $\begin{bmatrix} 1/2 & 0 \end{bmatrix}^T$.

Now suppose we implement the measurement-update equation using Potter's algorithm. We start out with

$$S_k^- = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.83}$$

We only have to iterate through Equation (6.74) one time since we only have one measurement. The rounded values of the parameters given in Equation (6.74) are

$$\begin{aligned}
\phi &= (S_k^-)^T H^T \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
a &= \frac{1}{\phi^T \phi + R} \\
&= \frac{1}{1+R} \\
&= 1 \\
\gamma &= \frac{1}{1+\sqrt{aR}} \\
&= \frac{1}{1+\sqrt{R}} \\
S_k^+ &= S_k^-(I - a\gamma\phi\phi^T) \\
&= \begin{bmatrix} \frac{\sqrt{R}}{1+\sqrt{R}} & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned} \tag{6.84}$$

Note that $S_k^+ S_k^{+T}$ is nonsingular. The rounded values of the square root of the *a priori* covariance, the parameters of Equation (6.74), and the Kalman gain at the next time step $(k+1)$ would be given by

$$\begin{aligned}
S_{k+1}^- &= S_k^+ \\
\phi &= (S_{k+1}^-)^T H^T \\
&= \begin{bmatrix} \frac{\sqrt{R}}{1+\sqrt{R}} \\ 0 \end{bmatrix} \\
a &= \frac{1}{\phi^T \phi + R} \\
&= \frac{1+R+2\sqrt{R}}{R^2+2R+2R\sqrt{R}} \\
&= \frac{1+2\sqrt{R}}{2R+2R\sqrt{R}} \\
K_{k+1} &= aS_{k+1}^- \phi \\
&= \frac{1+2\sqrt{R}}{2R(1+\sqrt{R})} \begin{bmatrix} \frac{R}{1+R+2\sqrt{R}} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2(1+\sqrt{R})} \\ 0 \end{bmatrix}
\end{aligned} \tag{6.85}$$

Note that the rounded Kalman gain is almost identical to the exact Kalman gain given by Equation (6.80). This shows the benefit that can be gained by using the square root filter.

$\nabla\nabla\nabla$

6.3.4 Square root measurement update via triangularization

The previous section derived a measurement update based on Potter's algorithm that could be performed on the square root of the Kalman filter estimation covariance. This section derives an alternative method for performing the measurement update. Suppose that we want to design a Kalman filter for a system with n states and r measurements. Suppose that we can find an orthogonal matrix $(n+r) \times (n+r)$ matrix $\tilde{T}$ such that

$$\begin{bmatrix} (R_k + H_k P_k^- H_k^T)^{T/2} & \tilde{K}_k^T \\ 0 & (S_k^+)^T \end{bmatrix} = \tilde{T} \begin{bmatrix} R_k^{T/2} & 0 \\ (S_k^-)^T H_k^T & (S_k^-)^T \end{bmatrix} \tag{6.86}$$

S_k^- and S_k^+ are the square roots of the *a priori* and *a posteriori* covariances, and $\tilde{K}_k$ is defined as

$$\tilde{K}_k = K_k (R_k + H_k P_k^- H_k^T)^{T/2} \tag{6.87}$$

where K_k is the normal Kalman gain matrix. Note that S_k^+ in Equation (6.86) is not known until after an orthogonal $\tilde{T}$ is found that forces the left side of Equation (6.86) into the specified form. That is, we need to find a $\tilde{T}$ so that the upper-left $r \times r$ block of the left side of Equation (6.86) is equal to $(R_k + H_k P_k^- H_k^T)^{T/2}$, the upper-right $r \times n$ block is equal to $\tilde{K}_k^T$, and the lower-left $n \times r$ block is equal to 0. After such a $\tilde{T}$ is found, whatever the lower right $n \times n$ block turns out to be is, by definition, equal to $(S_k^+)^T$, which is the transpose of the square root of P_k^+. Now write the $(n+r) \times (n+r)$ matrix $\tilde{T}$ as

$$\tilde{T} = \begin{bmatrix} \tilde{T}_{11} & \tilde{T}_{12} \\ \tilde{T}_{21} & \tilde{T}_{22} \end{bmatrix} \tag{6.88}$$

where $\tilde{T}_{11}$ is an $r \times r$ matrix, $\tilde{T}_{12}$ is an $r \times n$ matrix, $\tilde{T}_{21}$ is an $n \times r$ matrix, and $\tilde{T}_{22}$ is an $n \times n$ matrix. Since $\tilde{T}$ is orthogonal we can write

$$\begin{aligned} \tilde{T}^T \tilde{T} &= \begin{bmatrix} \tilde{T}_{11}^T & \tilde{T}_{21}^T \\ \tilde{T}_{12}^T & \tilde{T}_{22}^T \end{bmatrix} \begin{bmatrix} \tilde{T}_{11} & \tilde{T}_{12} \\ \tilde{T}_{21} & \tilde{T}_{22} \end{bmatrix} \\ &= \begin{bmatrix} \tilde{T}_{11}^T \tilde{T}_{11} + \tilde{T}_{21}^T \tilde{T}_{21} & \tilde{T}_{11}^T \tilde{T}_{12} + \tilde{T}_{21}^T \tilde{T}_{22} \\ \tilde{T}_{12}^T \tilde{T}_{11} + \tilde{T}_{22}^T \tilde{T}_{21} & \tilde{T}_{12}^T \tilde{T}_{12} + \tilde{T}_{22}^T \tilde{T}_{22} \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \end{aligned} \tag{6.89}$$

Now we expand Equation (6.86) as

$$\begin{aligned} &\begin{bmatrix} (R_k + H_k P_k^- H_k^T)^{T/2} & \tilde{K}_k^T \\ 0 & (S_k^+)^T \end{bmatrix} = \\ &\begin{bmatrix} \tilde{T}_{11} R_k^{T/2} + \tilde{T}_{12} (S_k^-)^T H_k^T & \tilde{T}_{12} (S_k^-)^T \\ \tilde{T}_{21} R_k^{T/2} + \tilde{T}_{22} (S_k^-)^T H_k^T & \tilde{T}_{22} (S_k^-)^T \end{bmatrix} \end{aligned} \tag{6.90}$$

We will equate the four matrix partitions of this equation to write four separate equalities. We will then take each equality and premultiply each side by its transpose to obtain four new equalities. The first two equalities obtained this way are

$$\begin{aligned}(R_k + H_k P_k^- H_k^T)^{1/2}(\cdots)^{T/2} &= \\ & R_k^{1/2}\tilde{T}_{11}^T\tilde{T}_{11}R_k^{T/2} + H_k S_k^- \tilde{T}_{12}^T \tilde{T}_{11} R_k^{T/2} + \\ & R_k^{1/2}\tilde{T}_{11}^T\tilde{T}_{12}(S_k^-)^T H_k^T + H_k S_k^- \tilde{T}_{12}^T \tilde{T}_{12}(S_k^-)^T H_k^T \\ 0 \quad = \quad & R_k^{1/2}\tilde{T}_{21}^T\tilde{T}_{21}R_k^{T/2} + R_k^{1/2}\tilde{T}_{21}^T\tilde{T}_{22}(S_k^-)^T H_k^T + \\ & H_k S_k^- \tilde{T}_{22}^T \tilde{T}_{21} R_k^{T/2} + H_k S_k^- \tilde{T}_{22}^T \tilde{T}_{22}(S_k^-)^T H_k^T \end{aligned} \tag{6.91}$$

Adding these two equations and using Equations (6.87) and (6.89) to simplify the result gives

$$R_k + H_k P_k^- H_k^T = R_k + H_k S_k^- (S_k^-)^T H_k^T \tag{6.92}$$

This shows that the proposed measurement update of Equation (6.86) is consistent with S_k^- being the square root of P_k^-.

The second two equalities that can be written from Equation (6.90) are

$$\begin{aligned} \tilde{K}_k \tilde{K}_k^T &= S_k^- \tilde{T}_{12}^T \tilde{T}_{12} (S_k^-)^T \\ S_k^+ (S_k^+)^T &= S_k^- \tilde{T}_{22}^T \tilde{T}_{22} (S_k^-)^T \end{aligned} \tag{6.93}$$

Adding these two equations and using Equation (6.89) to simplify the result gives

$$S_k^+ (S_k^+)^T + K_k (R_k + H_k P_k^- H_k^T) K_k^T = S_k^- (S_k^-)^T \tag{6.94}$$

Substituting the standard Kalman gain equation $K_k = P_k^- H_k^T (R_k + H_k P_k^- H_k^T)^{-1}$ into this equation gives

$$\begin{aligned} S_k^+ (S_k^+)^T + P_k^- H_k^T K_k^T &= P_k^- \\ S_k^+ (S_k^+)^T &= P_k^- - P_k^- H_k^T K_k^T \end{aligned} \tag{6.95}$$

Since the left side of the above equation is symmetric and the first term on the right side is symmetric, the last term on the right side must also be symmetric, which means that we can transpose it in the above equation to obtain

$$S_k^+ (S_k^+)^T = P_k^- - K_k H_k P_k^- \tag{6.96}$$

The right side of this equation is the Kalman filter measurement-update equation for P, which means that the left side of the equation must be P_k^+, which means that S_k^+ must be the square root of P_k^+. So if we can find an orthogonal $(n+r) \times (n+r)$ matrix $\tilde{T}$ such that

$$\begin{bmatrix} (R_k + H_k P_k^- H_k^T)^{T/2} & \tilde{K}_k^T \\ 0 & (n \times n \text{ matrix}) \end{bmatrix} = \tilde{T} \begin{bmatrix} R_k^{T/2} & 0 \\ (S_k^-)^T H_k^T & (S_k^-)^T \end{bmatrix} \tag{6.97}$$

then the lower-right $n \times n$ matrix on the left side of the equation is equal to the transpose of the square root of P_k^+, and this equation is mathematically equivalent to the original Kalman filter measurement-update equation for P_k. This measurement-update method results in numerical precision that is effectively twice as much as the standard Kalman filter, which helps to avoid numerical problems. However, the computation of $\tilde{T}$ adds a lot of computational effort to the Kalman filter. In addition, the form of the transformation given in Equation (6.97) makes it of questionable practicality (see Problem 6.10).

6.3.5 Algorithms for orthogonal transformations

Several numerical algorithms are available for performing the orthogonal transformations that are required to solve for the T and S_k^- matrices in Equation (6.58). Some algorithms that can be used are the Householder method, the Givens method, the Gram–Schmidt method, and the modified Gram–Schmidt method. In this section we will present (without derivation) the Householder algorithm and the modified Gram–Schmidt algorithm. Derivations and presentations of the other algorithms can be found in many texts on numerical linear algebra, such as [Hor85, Gol89, Moo00]. A comparison of Gram–Schmidt, modified Gram–Schmidt, and Householder transformations can be found in [Jor68], where it is stated that the modified Gram–Schmidt procedure is best (from a numerical point of view), with the Householder method offering competitive performance.

6.3.5.1 The Householder algorithm The algorithm presented here was developed by Alston Householder [Hou64, Chapter 5], applied to least squares estimation by Gene Golub [Gol65], and summarized for Kalman filtering by Paul Kaminski [Kam71].

1. Suppose that we have a $2n \times n$ matrix $A^{(1)}$, and we want to find an $n \times n$ matrix W such that

$$TA^{(1)} = \begin{bmatrix} W \\ 0 \end{bmatrix} \tag{6.98}$$

   where T is an orthogonal $2n \times 2n$ matrix, and 0 is the $n \times n$ matrix consisting of all zeros. Note that this problem statement is in the same form as Equation (6.58). Also note that we do not necessarily need to find T; our goal is to find W.

2. For $k = 1, \cdots, n$ perform the following:

   (a) Compute the scalar σ_k as

$$\sigma_k = \operatorname{sgn}\left(A_{kk}^{(k)}\right) \sqrt{\sum_{i=k}^{2n} \left(A_{ik}^{(k)}\right)^2} \tag{6.99}$$

   where $A_{ik}^{(k)}$ is the element in the ith row and kth column of $A^{(k)}$. The sgn($\cdot$) function is defined to be equal to +1 if its argument is greater than or equal to zero, and −1 if its argument is less than zero.

   (b) Compute the scalar β_k as

$$\beta_k = \frac{1}{\sigma_k \left(\sigma_k + A_{kk}^{(k)}\right)} \tag{6.100}$$

   (c) For $i = 1, \cdots, 2n$ perform the following:

$$u_i^{(k)} = \begin{cases} 0 & i < k \\ \sigma_k + A_{kk}^{(k)} & i = k \\ A_{ik}^{(k)} & i > k \end{cases} \tag{6.101}$$

This gives a $2n$-element column vector $u^{(k)}$.

(d) For $i = 1, \cdots, n$ perform the following:

$$y_i^{(k)} = \begin{cases} 0 & i < k \\ 1 & i = k \\ \beta_k u^{(k)T} A_i^{(k)} & i > k \end{cases} \tag{6.102}$$

where $A_i^{(k)}$ is the ith column of $A^{(k)}$. This gives an n-element column vector $y^{(k)}$.

(e) Compute the $2n \times n$ matrix $A^{(k+1)}$ as

$$A^{(k+1)} = A^{(k)} - u^{(k)} y^{(k)T} \tag{6.103}$$

3. After the above steps have been executed, $A^{(n+1)}$ has the form

$$A^{(n+1)} = \begin{bmatrix} W \\ 0 \end{bmatrix} \tag{6.104}$$

where W is the $n \times n$ matrix that we are trying to solve for. Note that if $\sigma_k = 0$ at any stage of the algorithm, that means $A^{(1)}$ is rank deficient and the algorithm will fail. Also note that the above algorithm does not compute the T matrix. However, we can find the T matrix as

$$\begin{aligned} T &= T^{(n)} T^{(n-1)} \cdots T^{(1)} \\ T^{(k)} &= I - \beta_k u^{(k)} u^{(k)T} \qquad i = 1, \cdots, n \end{aligned} \tag{6.105}$$

6.3.5.2 The modified Gram-Schmidt algorithm The modified Gram-Schmidt algorithm for orthonormalization that is presented here is discussed in most linear systems books [Kai80, Bay99, Che99]. It was first given in [Bjo67] and was summarized for Kalman filtering in [Kam71].

1. Suppose that we have a $2n \times n$ matrix $A^{(1)}$, and we want to find an $n \times n$ matrix W such that

$$TA^{(1)} = \begin{bmatrix} W \\ 0 \end{bmatrix} \tag{6.106}$$

where T is an orthogonal $2n \times 2n$ matrix, and 0 is the $n \times n$ matrix consisting of all zeros. Note that this problem statement is in the same form as Equation (6.58).

2. For $k = 1, \cdots, n$ perform the following.

(a) Compute the scalar σ_k as

$$\sigma_k = \sqrt{A_k^{(k)T} A_k^{(k)}} \tag{6.107}$$

where $A_i^{(k)}$ is the ith column of $A^{(k)}$.

(b) Compute the kth row of W as

$$W_{kj} = \begin{cases} 0 & j = 1, \cdots, k-1 \\ \sigma_k & j = k \\ A_k^{(k)T} A_j^{(k)} / \sigma_k & j = k+1, \cdots, n \end{cases} \tag{6.108}$$

(c) Compute the kth row of T as

$$T_k = A_k^{(k)T}/\sigma_k \tag{6.109}$$

(d) If $(k < n)$, compute the last $(n-k)$ columns of $A^{(k+1)}$ as

$$A_j^{(k+1)} = A_j^{(k)} - W_{kj}A_k^{(k)}/\sigma_k \qquad j = k+1, \cdots, n \tag{6.110}$$

Note that the first k columns of $A^{(k+1)}$ are not computed in this algorithm.

As with the Householder algorithm, if $\sigma_k = 0$ at any stage of the algorithm, that means $A^{(1)}$ is rank deficient and the algorithm fails. After this algorithm completes, we have the first n rows of T, and T is an $n \times 2n$ matrix. If we want to know the last n rows of T, we can compute them using a regular Gram–Schmidt algorithm as follows [Hor85, Gol89, Moo00].

1. Fill out the T matrix that was begun above by appending a $2n \times 2n$ identity matrix to the bottom of it. This ensures that the rows of T span the entire $2n$-dimensional vector space:

$$T = \begin{bmatrix} T \\ I \end{bmatrix} \tag{6.111}$$

Note that this T is a $3n \times 2n$ matrix.

2. Now we perform a standard Gram–Schmidt orthonormalization procedure on the last $2n$ rows of T (with respect to the already obtained first n rows of T). For $k = n+1, \cdots, 3n$, compute the kth row of T as

$$\begin{aligned} T_k &= T_k - \sum_{i=1}^{k-1}(T_k T_i^T)T_i \\ T_k &= \frac{T_k}{\|T_k\|_2} \end{aligned} \tag{6.112}$$

If T_k is zero then that means that it is a linear combination of the previous rows of T. In that case, the division in the above equation will be a divide by zero, so instead T_k should be discarded. This discard will actually occur exactly n times so that this procedure will compute n additional rows of T and we will end up with an orthogonal $2n \times 2n$ matrix T.

The Gram–Schmidt algorithms are named after the Danish mathematician Jorgen Gram (1850-1916) and the German mathematician Erhard Schmidt (1876-1959). Schmidt received his doctorate in 1905 under David Hilbert's supervision, and in 1929 he was on the doctoral committee of Eberhard Hopf (see Section 3.4.4). However, the Gram–Schmidt algorithm was actually invented by Pierre Laplace (1749-1827).

6.4 U-D FILTERING

U-D filtering was introduced in [Bie76, Bie77a] as another way to increase the numerical precision of the Kalman filter. It is sometimes considered as a type of square root filtering, and sometimes it is considered distinct from square root filtering (depending on the author). It increases the computational cost of the filter but not so severely as the square root filter of the previous section.

The idea of U-D filtering is to factor the $n \times n$ matrix P as UDU^T, where U is an $n \times n$ upper triangular matrix with ones along the diagonal, and D is an $n \times n$ diagonal matrix. This can always be accomplished for a symmetric positive definite matrix P [Gol89, Chapter 4], so it can always be implemented on a Kalman filter. A U-D factorization routine can be implemented without too much difficulty. For example, suppose that we want to compute the U-D factorization of a 3×3 matrix. We can then write

$$\begin{aligned}\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{12} & p_{22} & p_{23} \\ p_{13} & p_{23} & p_{33} \end{bmatrix} &= \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ u_{12} & 1 & 0 \\ u_{13} & u_{23} & 1 \end{bmatrix} \\ &= \begin{bmatrix} d_{11} + d_{22}u_{12}^2 + d_{33}u_{13}^2 & d_{22}u_{12} + d_{33}u_{13}u_{23} & d_{33}u_{13} \\ d_{22}u_{12} + d_{33}u_{13}u_{23} & d_{22} + d_{33}u_{23}^2 & d_{33}u_{23} \\ d_{33}u_{13} & d_{33}u_{23} & d_{33} \end{bmatrix}\end{aligned} \tag{6.113}$$

We need to solve for the u_{ij} and d_{ii} elements. We can begin at the lower-right element of the matrix equality to see that $d_{33} = p_{33}$. Next we can look at the other elements in the third column to see that

$$\begin{aligned} u_{13} &= p_{13}/d_{33} \\ u_{23} &= p_{23}/d_{33} \end{aligned} \tag{6.114}$$

Now look at the $(2,2)$ and $(1,2)$ elements of the equality to see that

$$\begin{aligned} d_{22} &= p_{22} - d_{33}u_{23}^2 \\ u_{12} &= (p_{12} - d_{33}u_{13}u_{23})/d_{22} \end{aligned} \tag{6.115}$$

Finally look at the $(1,1)$ element of the equality to see that

$$d_{11} = p_{11} - d_{22}u_{12}^2 - d_{33}u_{13}^2 \tag{6.116}$$

This gives us the U-D factorization for a 3×3 symmetric matrix, and provides the outline for a general U-D factorization algorithm.

6.4.1 U-D filtering: The measurement-update equation

Recall from Equation (5.19) the measurement update equation for the covariance of the Kalman filter:

$$P^+ = P^- - P^-H^T(HP^-H^T + R)^{-1}HP^- \tag{6.117}$$

We have omitted the time subscripts for ease of notation. Now suppose that we process the measurements sequentially as discussed in Section 6.1. This gives the equation

$$P_i = P_{i-1} - P_{i-1}H_i^T(H_iP_{i-1}H_i^T + R_i)^{-1}H_iP_{i-1} \tag{6.118}$$

where H_i is the ith row of H, R_i is the ith diagonal entry of R, and P_i is the estimation covariance after i measurements have been processed. Now define the scalar $\alpha_i \equiv H_i P_{i-1} H_i^T + R_i$. Suppose that $P_{i-1} = U_{i-1} D_{i-1} U_{i-1}^T$, and $P_i = U_i D_i U_i^T$. With these factorizations we can write the measurement update of Equation (6.118) as

$$\begin{aligned} U_i D_i U_i^T &= U_{i-1} D_{i-1} U_{i-1}^T - \frac{1}{\alpha_i} U_{i-1} D_{i-1} U_{i-1}^T H_i^T H_i U_{i-1} D_{i-1} U_{i-1}^T \\ &= U_{i-1} \left[D_{i-1} - \frac{1}{\alpha_i} (D_{i-1} U_{i-1}^T H_i^T)(D_{i-1} U_{i-1}^T H_i^T)^T \right] U_{i-1}^T \quad (6.119) \end{aligned}$$

The term in brackets in the above equation is symmetric positive definite so it has a U-D factorization that can be written as

$$\bar{U}\bar{D}\bar{U}^T = \left[D_{i-1} - \frac{1}{\alpha_i} (D_{i-1} U_{i-1}^T H_i^T)(D_{i-1} U_{i-1}^T H_i^T)^T \right] \quad (6.120)$$

Combining this with Equation (6.119) gives

$$\begin{aligned} U_i D_i U_i^T &= U_{i-1} \bar{U}\bar{D}\bar{U}^T U_{i-1}^T \\ &= (U_{i-1}\bar{U})\bar{D}(U_{i-1}\bar{U})^T \quad (6.121) \end{aligned}$$

Note that $U_{i-1}\bar{U}$ is upper triangular with diagonal elements equal to 1, and $\bar{D}$ is diagonal. Therefore the above equation means that $U_i = U_{i-1}\bar{U}$, and $D_i = \bar{D}$:

$$\begin{aligned} U_i &= U_{i-1}\bar{U} \\ D_i &= \bar{D} \quad (6.122) \end{aligned}$$

This gives us a way of performing the measurement update of P in terms of its U-D factors. The algorithm can be summarized as follows.

The U-D measurement update

1. We start with the *a priori* estimation covariance P^- at time k. Define $P_0 = P^-$.
2. For $i = 1, \cdots, r$ (where r is the number of measurements), perform the following:
   (a) Define H_i as the ith row of H, R_i as the ith diagonal entry of R, and $\alpha_i = H_i P_{i-1} H_i^T + R_i$.
   (b) Perform a U-D factorization of P_{i-1} to obtain U_{i-1} and D_{i-1}, and then form the matrix on the right side of Equation (6.120).
   (c) Find the U-D factorization of the matrix on the right side of Equation (6.120) and call the factors $\bar{U}$ and $\bar{D}$.
   (d) Compute U_i and D_i from Equation (6.122).
3. The *a posteriori* estimation covariance is given as $P^+ = U_r D_r U_r^T$.

Since the U-D measurement-update equation relies on sequential filtering, the conditions discussed at the end of Section 6.1 apply to U-D filtering. That is, it probably does not make sense to implement U-D filtering unless one of the following two conditions is true.

1. The measurement noise covariance R_k is diagonal
2. The measurement noise covariance R is a constant.

6.4.2 U-D filtering: The time-update equation

Recall from Equation (5.19) the time-update equation for the covariance of the Kalman filter:

$$P^- = FP^+F^T + Q \tag{6.123}$$

We have omitted the time subscripts for ease of notation. If the Kalman filter is being used to estimate the state of an n-state system, then the P matrices will be $n \times n$ matrices. Suppose that P^+ is factored as $U^+D^+U^{+T}$ (from the measurement update equation discussed previously). We need to find the U-D factors of P^- such that $P^- = U^-D^-U^{-T} = FP^+F^T + Q$. Note that U^{-T} in this notation is *not* the transpose of the inverse of U; it is rather the transpose of U^-. The time update of Equation (6.123) can be written as

$$\begin{aligned} P^- &= FP^+F^T + Q \\ &= \begin{bmatrix} FU^+ & I \end{bmatrix} \begin{bmatrix} D^+ & 0 \\ 0 & Q \end{bmatrix} \begin{bmatrix} U^{+T}F^T \\ I \end{bmatrix} \\ &= W\hat{D}W^T \end{aligned} \tag{6.124}$$

where W and $\hat{D}$ are defined by the above equation. Note that W is an $n \times 2n$ matrix, and $\hat{D}$ is a $2n \times 2n$ matrix. From the above equation we see that the U-D factors of P^- need to satisfy

$$U^-D^-U^{-T} = W\hat{D}W^T \tag{6.125}$$

The transpose of W can be written as

$$W^T = \begin{bmatrix} w_1^T & \cdots & w_n^T \end{bmatrix} \tag{6.126}$$

That is, w_i (a $2n$-element row vector) is the ith row of W. Now we find n vectors v_i such that

$$v_k\hat{D}v_j^T = 0 \quad k \neq j \tag{6.127}$$

The v_i vectors ($2n$-element row vectors) can be found with the following Gram–Schmidt orthogonalization procedure [Hor85, Gol89, Moo00]:

$$\begin{aligned} v_n &= w_n \\ v_k &= w_k - \sum_{j=k+1}^{n} \frac{w_k\hat{D}v_j^T}{v_j\hat{D}v_j^T} v_j \qquad k = n-1, \cdots, 1 \end{aligned} \tag{6.128}$$

If we define $u(k, j)$ as

$$u(k, j) = \frac{w_k\hat{D}v_j^T}{v_j\hat{D}v_j^T} \qquad j, k = 1, \cdots, n \tag{6.129}$$

then from Equation (6.128) we see that w_k can be expressed as

$$w_k = v_k + \sum_{j=k+1}^{n} u(k,j)v_j \qquad k = 1, \cdots, n \tag{6.130}$$

or equivalently

$$w_k^T = v_k^T + \sum_{j=k+1}^{n} u(k,j)v_j^T \qquad k = 1, \cdots, n \tag{6.131}$$

These n equations can be written as

$$\begin{aligned} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} &= \begin{bmatrix} 1 & u(1,2) & \cdots & u(1,n) \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u(n-1,n) \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \\ W &= U^- V \end{aligned} \tag{6.132}$$

The $n \times 2n$ matrix W, the $n \times n$ matrix U^-, and the $n \times 2n$ matrix V are defined by the above equation. Note that U^- is a unit upper triangular matrix. The matrix product $W\hat{D}W^T$ can then be written as

$$\begin{aligned} W\hat{D}W^T &= (U^-V)\hat{D}(U^-V)^T \\ &= U^-(V\hat{D}V^T)U^{-T} \\ &= U^-D^-U^{-T} \end{aligned} \tag{6.133}$$

where the D^- matrix is defined by the above equation. From Equation (6.127), we see that the v_i vectors are orthogonal with respect to the $\hat{D}$ inner product. We therefore know that

$$\begin{aligned} D^- &= V\hat{D}V^T = \text{diag}(d_1, \cdots, d_n) \\ d_k &= v_k\hat{D}v_k^T \end{aligned} \tag{6.134}$$

That is, D^- is a diagonal matrix. From Equations (6.124), (6.125), and (6.133) we see that U^- and D^- satisfy the conditions of being the U-D factors of P^-. This gives us a way to perform the Kalman filter time-update equation in U-D factorization form. The algorithm can be summarized as follows.

The U-D time update

1. Begin with $P^+ = U^+D^+U^{+T}$ (from the measurement update equation).
2. Define the following matrices.

$$\begin{aligned} W &= \begin{bmatrix} FU^+ & I \end{bmatrix} \\ \hat{D} &= \begin{bmatrix} D^+ & 0 \\ 0 & Q \end{bmatrix} \end{aligned} \tag{6.135}$$

3. Use the rows of W along with the Gram–Schmidt orthogonalization procedure to generate v_i vectors that are orthogonal with respect to the $\hat{D}$ inner product. The algorithm for generating the v_i vectors is given in Equation (6.128).

4. Form the V matrix using the v_i vectors as rows; see Equation (6.132).
5. Use $\hat{D}$ inner products to form the unit upper triangular matrix U^-; see Equations (6.129) and (6.132).
6. Define D^- as $D^- = V\hat{D}V^T$.

The U-D filter results in twice as much precision as the standard Kalman filter, just like the square root filter, but it requires less computation than the square root filter. If some of the states are missing from the measurement vector, a more efficient U-D algorithm can be derived [Bar83].

6.5 SUMMARY

In this chapter, we discussed the sequential Kalman filter, which is mathematically identical to the Kalman filter, but which avoids matrix inversion. This is an attractive formulation for embedded systems in which computational time and memory are at a premium. However, sequential filtering can only be used if the noise covariance is diagonal, or if the noise covariance is constant. Information filtering is also equivalent to the Kalman filter, but it propagates the inverse of the covariance. This can be computationally beneficial in cases in which the number of measurements is much larger than the number of states. Square root filtering and U-D filtering effectively increase the precision of the Kalman filter. Although these approaches require additional computational effort, they can help prevent divergence and instability. Gerald Bierman's book provides an excellent and comprehensive overview of square root and U-D filtering [Bie77b].

We see that we have a number of different choices when implementing a Kalman filter.

- Covariance filtering or information filtering
- Standard filtering, square root filtering, or U-D filtering
- Batch filtering or sequential filtering

Any of these choices can be made independently of the other choices. For instance, we can choose to combine information filtering with square root filtering [Kam71] in much the same way as we combined covariance filtering with square root filtering in this chapter. The choices in the list above gives us a total of 12 different Kalman filter formulations (two choices in the first item, three choices in the second item, and two choices in the third item). There are also other choices that are not listed above, especially other types of square root filtering. A numerical comparison of various Kalman filter formulations (including the standard filter, the square root covariance filter, the square root information filter, and the Chandrasekhar algorithm) is given in [Ver86]. Numerical and computational comparisons of various Kalman filtering approaches are given in [Bie73, Bie77a]. Continuous-time square root filtering is discussed in [Mor78] and in Section 8.3.3 of this book.

PROBLEMS

Written exercises

6.1 In this chapter, we discussed alternatives to the standard Kalman filter formulation. Some of these alternatives include the sequential Kalman filter, the information filter, and the square root filter.

a) What is the advantage of the sequential Kalman filter over the batch Kalman filter? What is the advantage of the batch Kalman filter over the sequential Kalman filter?

b) What is the advantage of the information filter over the standard Kalman filter? What is the advantage of the standard Kalman filter over the information filter?

c) What is the advantage of the square root filter over the standard Kalman filter? What is an advantage of the standard Kalman filter over the square root Kalman filter?

6.2 Suppose that you have a system with the following measurement and measurement noise covariance matrices:

$$\begin{aligned} H &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ R &= \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \end{aligned}$$

You want to use a sequential Kalman filter to estimate the state of the system. Derive the normalized measurement, measurement matrix, and measurement noise covariance matrix that could be used in a sequential Kalman filter.

6.3 Consider the two alternative forms for the information matrix time-update equation. What advantages does Equation (6.28) have? What advantages does Equation (6.30) have?

6.4 A radioactive mass has a half-life of τ seconds. At each time step k the number of emitted particles x is half of what it was one time step ago, but there is some error w_k (zero-mean with variance Q_k) in the number of emitted particles due to background radiation. At each time step the number of emitted particles is counted with two separate and independent instruments. The instruments used to count the number of emitted particles both have a random error at each time step that is zero-mean with a unity variance. The initial uncertainty in the number of radioactive particles is a random variable with zero mean and unity variance.

a) The discrete-time equations that model this system have a one-dimensional state and a two-dimensional measurement. Use the information filter to compute the *a priori* and *a posteriori* information matrix at $k = 1$ and $k = 2$. Assume that $Q_0 = 1$ and $Q_1 = 5/4$.

b) Another way to solve this problem is to realize that the two measurements can be averaged to form a single measurement with a smaller variance than the two independent measurements. What is the variance of the averaged measurement at each time step? Use the standard Kalman filter equations to compute the *a priori* and *a posteriori* covariance matrix at $k = 1$ and $k = 2$, and verify that it is the inverse of the information matrix that you computed in part (a).

6.5 Prove that the singular values of a diagonal matrix are the magnitudes of the diagonal elements.

6.6 Prove that SS^T is symmetric positive semidefinite for any S matrix.

6.7 Find an upper triangular matrix S (using only paper and pencil) such that

$$SS^T = \begin{bmatrix} 1 & 3 \\ 3 & 9 \end{bmatrix}$$

Is your solution unique?

6.8 Find an upper triangular matrix S (using only paper and pencil) such that

$$SS^T = \begin{bmatrix} 5 & 2 & -2 \\ 2 & 2 & -1 \\ -2 & -1 & 1 \end{bmatrix}$$

How many solutions exist to this problem?

6.9 Verify Equation (6.70). Hint: Equate the two sides of the equation, take the trace, and solve for γ. Make sure to explain why taking the trace is valid.

6.10 Suppose that an orthogonal matrix $\tilde{T}$ is desired to satisfy Equation (6.97), where Cholesky factorization is used to compute the matrix square roots on the left side of the equation. This equation can then be written as $U = \tilde{T}A$, where U is an upper triangular matrix. Show that such a transformation cannot be found unless the two-norm of the first column of A happens to be equal to $|U_{11}|$. [Note that this does not necessarily prevent the possibility of the transformation of Equation (6.97), because U could be nontriangular if nontriangular square root matrices are used to form the U matrix.]

6.11 Use the Householder method (using only paper and pencil) to find an orthogonal T such that $TA = \begin{bmatrix} W \\ 0 \end{bmatrix}$ where W is a 2×2 matrix and

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 1 \\ 2 & 2 \end{bmatrix}$$

6.12 Use the modified Gram–Schmidt method (using only paper and pencil) to solve Problem 6.11.

6.13 Compute the U-D factorization (using only paper and pencil) for the matrix

$$P = \begin{bmatrix} 1 & 3 \\ 3 & 9 \end{bmatrix}$$

Computer exercises

6.14 Consider the RLC circuit of Example 1.8 with $R = 100$ and $L = C = 1$. Suppose the applied voltage is continuous-time, zero-mean white noise with a standard deviation of 3. The initial capacitor voltage and inductor current are both zero. Discretize the system with a time step of 0.1. The discrete-time measurements consist of the capacitor voltage and the inductor current, both measurements containing zero-mean unity variance noise. Implement a sequential Kalman filter for the system. Simulate the system for 2 seconds. Let the initial state estimate be equal to the initial state, and the initial estimation covariance be equal to $0.1I$. Hint: Set the discrete-time process noise covariance $Q = Q_c\Delta t$, where Q_c is the covariance of the continuous-time process noise, and Δt is the discretization step size. Q will be nondiagonal, which means you need to use the algorithm in Section 2.7 to simulate the process noise.

a) Generate a plot showing the *a priori* variance of the capacitor voltage estimation error, and the two *a posteriori* variances of the capacitor voltage estimation error.

b) Generate a plot showing a typical trace of the true, *a posteriori* estimated, and measured capacitor voltage. What is the standard deviation of the capacitor voltage measurement error? What is the standard deviation of the capacitor voltage estimation error?

6.15 The pitch motion of an aircraft flying at constant speed can be approximately described by the following equations [Ste94]:

$$\begin{aligned}\dot{x} &= \begin{bmatrix} -0.5680 & 17.9800 \\ 1.0000 & -1.2370 \end{bmatrix} x + \begin{bmatrix} 0.1750 & 0.1750 \\ -0.0010 & -0.0010 \end{bmatrix} u + \begin{bmatrix} 17.9800 \\ -1.2370 \end{bmatrix} w \\ y(t_k) &= x(t_k) + v_k\end{aligned}$$

where x_1 is the pitch rate, x_2 is the angle of attack, u consists of the elevator and flap angles, and w is disturbance due to wind. Suppose that the variance of the wind disturbance is 0.001, and the measurement variances are 0.3. Discretize the system with a step size of 0.01 and simulate the system and a square root Kalman filter for 100 time steps. Use an initial state of zero, an initial state estimate of zero, an initial estimation-error covariance of $0.01I$, and a control input of zero. Hint: Set the discrete-time process noise covariance $Q = Q_c\Delta t$, where Q_c is the covariance of the continuous-time process noise, and Δt is the discretization step size. Q will be nondiagonal, which means you need to use the algorithm in Section 2.7 to simulate the process noise.

a) Generate a plot showing the *a posteriori* variance of the estimation errors of the two states.

b) Generate a plot showing a typical trace of the true, *a posteriori* estimated, and measured pitch rate. What is the standard deviation of the pitch rate measurement error? What is the standard deviation of the pitch rate estimation error?

c) Generate a plot showing a typical trace of the true, *a posteriori* estimated, and measured angle of attack. What is the standard deviation of the angle of attack measurement error? What is the standard deviation of the angle of attack estimation error?