The Discrete-Time Kalman Filter

Recall: RLS allows us to estimate some unknown constant vector $\bar{x}$ where each subsequent estimate $\hat{x}_k$ incorporates the measurement $\bar{z}_k$

→ Estimating $\hat{x}_k$ requires only $\hat{x}_{k-1}$, $\bar{z}_k$, and P_{k-1}

→ We produce a new $\hat{x}_k$ and P_k to pass forward at every iteration

→ The covariance P_k is always equal to or smaller than P_{k-1}!

Problem: RLS assumes that the quantity to be estimated is a constant $\bar{x}$

→ What if actually want to estimate a time-varying system state $\bar{x}_k$?

Problem statement: We have a DT linear system:

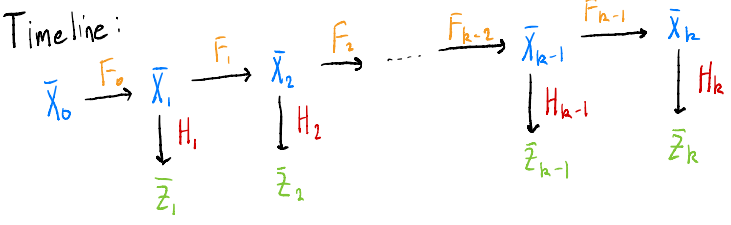
$$\bar{x}_k = F_{k-1}\bar{x}_{k-1} + G_{k-1}\bar{u}_{k-1} + \bar{w}_{k-1}$$

$$\bar{z}_k = H_k\bar{x}_k + \bar{v}_k$$

- $\bar{x}_k$ is an $N \times 1$ vector called the system state at time k
- F_{k-1} is an $N \times N$ matrix called the state transition matrix at time $k-1$
- $\bar{u}_{k-1}$ is an $L \times 1$ vector called the control vector at time $k-1$
- G_{k-1} is an $N \times L$ matrix called the control matrix at time $k-1$
- $\bar{w}_{k-1}$ is an $N \times 1$ vector called process noise at time $k-1$
  - → $\bar{w}_{k-1}$ is zero-mean, white Gaussian noise w/ $N \times N$ process covariance Q_{k-1}
  - → $E\{\bar{w}_k \bar{w}_j^T\} = Q_k \delta[k-j]$ and $\bar{w}_{k-1} \sim N(0, Q_{k-1})$
- $\bar{z}_k$ is an $M \times 1$ vector called the measurement vector at time k
- H_k is an $M \times N$ vector called the measurement/observation matrix at time k

- $\bar{v}_k$ is an $M \times 1$ vector called measurement/observation noise at time k
  - → $\bar{v}_k$ is zero-mean, white Gaussian noise w/ $M \times M$ measurement/observation covariance R_k
  - → $E\{\bar{v}_k \bar{v}_j^T\} = R_k \delta[k-j]$ and $\bar{v}_k \sim N(0, R_k)$
  - → $\bar{w}_k$ and $\bar{v}_j$ are uncorrelated for all k & j! → $E\{\bar{w}_k \bar{v}_j^T\} = 0$

Timeline:



Goal: Given our knowledge of the system dynamics, and our knowledge of all the measurements $\bar{z}_0, \bar{z}_1, \cdots, \bar{z}_k$, we want to determine the best possible estimate of the system state at time k → $\hat{x}_k$.

Starting Point: RLS

Recall: RLS is unbiased
- → $\hat{x}_k$ is a RV w/ $E\{\hat{x}_k\} = \bar{x}$
- → When we fuse a new measurement $\bar{z}_k$ to get $\hat{x}_k$, the assumption is that $E\{\hat{x}_{k-1}\} = \bar{x}$ AND that $E\{\bar{z}_k\} = H_k \bar{x}$

Let's try this approach w/ a time-varying system state now...

As assumed by RLS, suppose we begin iteration k with a prior system state estimate $\hat{x}_{k-1}$ w/ $E\{\hat{x}_{k-1}\} = \bar{x}_{k-1}$, a covariance w/ $\text{cov}\{\hat{x}_{k-1}\} = P_{k-1}$, and a new measurement $\bar{z}_k$ w/ $E\{\bar{z}_k\} = H_k \bar{x}_k$

Q: Can we use the RLS equations directly to get $\hat{x}_k$ and P_k?

A: No. We are no longer estimating a constant - we are estimating a state that may have changed between $k-1$ and k → $E\{\hat{x}_{k-1}\} \neq \bar{x}_k$!

Solution: We can propagate the previous state estimate (and covariance) through the system to get an estimate (and covariance) that *do* match the new variable ($\bar{x}_k$) we are trying to estimate!

$$\rightarrow \hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}\bar{u}_{k-1}$$

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

} Time-update equations / Prediction equations

$\rightarrow E\{\hat{x}_k^-\} = \bar{x}_k$ and $cov\{\hat{x}_k^-\} = P_k^-$, so we can now fuse these with $\bar{z}_k$ using RLS equations to get a final estimate $\hat{x}_k^+$ and P_k^+!

What's with the $\hat{x}_k^-/P_k^-$ and $\hat{x}_k^+/P_k^+$?

→ At this point, for any iteration k, we will have *two* separate estimates of $\bar{x}_k$: One before including the measurement $\bar{z}_k$ ($\hat{x}_k^+/P_k^+$) and one before including the measurement ($\hat{x}_k^-/P_k^-$)

→ $\hat{x}_k^-$ and $\hat{x}_k^+$ are both estimates of $\bar{x}_k$, and P_k^- and P_k^+ are the respective covariances of these estimates.

One way to think about $\hat{x}_k^-$ and $\hat{x}_k^+$ is that they are the expected value of the RV $\bar{x}_k$ conditioned on different sets of observed measurements:

$$\hat{x}_k^- = E\{\bar{x}_k \mid \bar{z}_1, \bar{z}_2, \dots, \bar{z}_{k-1}\}$$

$\hat{x}_k^-$ is called the *a priori state estimate*; P_k^- is called the *a priori state covariance*

and

$$\hat{x}_k^+ = E\{\bar{x}_k \mid \bar{z}_1, \bar{z}_2, \dots, \bar{z}_{k-1}, \bar{z}_k\}$$

$\hat{x}_k^+$ is called the *a posteriori state estimate*; P_k^+ is called the *a posteriori state covariance*

Now can apply our RLS equations to get the final state estimate and covariance for iteration k

→ Main difference: in RLS, we fuse $\bar{z}_k$ w/ $\hat{x}_{k-1}$ and P_{k-1} to get $\hat{x}_k$ and P_k

→ $\bar{x}$ is constant, so the "a priori" state / covariance are just the state / covariance from the previous iteration

→ Now, we must fuse $\bar{z}_k$ w/ $\hat{x}_k^-$ and P_k^- to get $\hat{x}_k^+$ and P_k^+

→ Fortunately, we can still use the same RLS equations, we just have to change the variables around. Therefore, we get:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$
$$= P_k^+ H_k^T R_k^{-1}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k (\bar{z}_k - H_k \hat{x}_k^-)$$
$$P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T$$
$$= \left((P_k^-)^{-1} + H_k^T R_k^{-1} H_k \right)^{-1}$$
$$= (I - K_k H_k) P_k^-$$

Measurement-update equations

The time-update and measurement-update steps, when put together, for the core of the DT Kalman filter algorithm.

The Discrete-Time Kalman Filter:

1. Start by formulating the system equations:

$$\bar{x}_k = F_{k-1} \bar{x}_{k-1} + G_{k-1} \bar{u}_{k-1} + \bar{w}_{k-1} \quad \text{w/ } \text{cov}\{\bar{w}_k\} = Q_k$$
$$\bar{z}_k = H_k \bar{x}_k + \bar{v}_k \quad \text{w/ } \text{cov}\{\bar{v}_k\} = R_k$$

→ Need to know F_k, G_k, and H_k (easy), Q_k and R_k (harder), and $\bar{u}_k$ for all k

2. As with RLS, we must initialize the Kalman filter with an initial guess at the state and the covariance of our estimate:

$$\hat{x}_0^+ = E\{\bar{x}_0\} \quad \text{and} \quad P_0^+ = \text{cov}\{\hat{x}_0^+\}$$

Note: we treat $\hat{x}_0^+$ and P_0^+ as a posteriori to fit it nicely into the recursive nature of the Kalman filter despite having fused no measurements

3. For all $k = 1, 2, \ldots$, do the following:

a.) The time-update (or "prediction") step:

$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}\bar{u}_{k-1}$$

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

b.) The measurement-update (or just "update") step:

$$K_k = P_k^- H_k^T\left(H_k P_k^- H_k^T + R_k\right)^{-1} \quad \rightarrow \text{In the Kalman filter, } K_k \text{ is called the "Kalman gain"}$$

$$= P_k^+ H_k^T R_k^{-1}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k\left(\bar{z}_k - H_k\hat{x}_k^-\right)$$

$$P_k^+ = (I - K_k H_k)P_k^-(I - K_k H_k)^T + K_k R_k K_k^T$$

$$= \left((P_k^-)^{-1} + H_k^T R_k^{-1} H_k\right)^{-1}$$

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

See example PPT notes for some tangible intuition in how each step works in practice

Kalman Gain?

One way is to think of K_k as a "weighting" between the prior state estimate $\hat{x}_k^-$ and the measurement $\bar{z}_k$ based on confidence (i.e., how does P_k^- compare to R_k, or more specifically, how does $H_k P_k^- H_k^T$ compare to R_k)

Scalar example provides extra intuition:

Suppose $\bar{x}_k$ is a scalar, $\bar{z}_k$ is a scalar, and $H_k = 1$ (i.e., $\bar{z}_k = \bar{x}_k + \bar{v}_k$)

→ P_k^- and P_k^+ are scalar variances, as is R_k

→ $K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1} = \dfrac{P_k^- H_k}{P_k^- H_k^2 + R_k} = \dfrac{P_k^-}{P_k^- + R_k}$

Transpose doesn't affect scalar

$H_k = 1$

$H_k = 1$

→ $\hat{x}_k^+ = \hat{x}_k^- + K_k\left(\bar{z}_k - H_k \hat{x}_k^-\right) = \hat{x}_k^- + \dfrac{P_k^-}{P_k^- + R_k}\left(\bar{z}_k - \hat{x}_k^-\right)$

$$= \left(1 - \frac{P_k^-}{P_k^- + R_k}\right)\hat{x}_k^- + \frac{P_k^-}{P_k^- + R_k}\bar{z}_k = \hat{x}_k^+$$

→ In this special case, $\hat{x}_k^+$ is a weighted sum of $\hat{x}_k^-$ and $\bar{z}_k$, where $E\{\hat{x}_k^-\} = E\{\bar{z}_k\} = \bar{x}_k$ in this case only because $H_k = 1$

→ If $P_k^- > R_k$, $\hat{x}_k^+$ will be weighted more toward $\bar{z}_k$

→ If $P_k^- < R_k$, $\hat{x}_k^+$ will be weighted more toward $\hat{x}_k^-$

As $P_k \to 0$, $\dfrac{P_k^-}{P_k^- + R_k} \to 0$ and $\hat{x}_k^+ = \hat{x}_k^-$

As $P_k \to \infty$, $\dfrac{P_k^-}{P_k^- + R_k} \to 1$ and $\hat{x}_k^+ = \bar{z}_k$

As $R_k \to 0$, $\dfrac{P_k^-}{P_k^- + R_k} \to 1$ and $\hat{x}_k^+ = \tilde{z}_k$

As $R_k \to \infty$, $\dfrac{P_k^-}{P_k^- + R_k} \to 0$ and $\hat{x}_k^+ = \hat{x}_k^-$

If $P_k^- = R_k$, $\dfrac{P_k^-}{P_k^- + R_k} = \dfrac{1}{2}$ and $\hat{x}_k^+ = \dfrac{1}{2}\hat{x}_k^- + \dfrac{1}{2}\tilde{z}_k$

How about the posterior covariance P_k^+?

$$\to P_k^+ = (I - H_k K_k) P_k^- = (1 - K_k) P_k^- = \left(1 - \frac{P_k^-}{P_k^- + R_k}\right) P_k^-$$

This form is convenient

$$= \left(\frac{P_k^- + R_k}{P_k^- + R_k} - \frac{P_k^-}{P_k^- + R_k}\right) P_k^- = \frac{P_k^- R_k}{P_k^- + R_k} = \boxed{\frac{1}{\frac{1}{P_k^-} + \frac{1}{R_k}} = P_k^+}$$

Harmonic Mean Form

→ In this special case, the a posteriori state covariance is the harmonic mean of the a priori state covariance and the measurement covariance (again, only works because $H_k = 1$)

→ P_k^+ is guaranteed to be $\leq P_k^-$ and $\leq R_k$

As $P_k^- \to 0$, $P_k^+ \to 0$ and as $R_k \to 0$, $P_k^+ \to 0$

As $P_k^- \to \infty$, $P_k^+ \to R_k$ and as $R_k \to \infty$, $P_k^+ \to P_k^-$

Kalman Filter Properties

1. If the system is static (i.e., $F_k = I$ and $G_k/\bar{u}_k = 0$), then the Kalman filter reduces to RLS

2. The Kalman filter produces the same results as an optimal estimator acting on all measurements $\bar{z}_1, \ldots, \bar{z}_k$ at once

$$\rightarrow \hat{x}_k^+ = E\{\bar{x}_k \mid \hat{x}_{k-1}^+, \bar{z}_k\} = E\{\bar{x}_k \mid \bar{z}_1, \ldots, \bar{z}_k\}$$

3. Suppose we have an estimate $\hat{x}_k$ of a system state $\bar{x}_k$ w/ error $\bar{\varepsilon}_k = \bar{x}_k - \hat{x}_k$. Suppose we want to minimize the weighted two-norm $E\{\bar{\varepsilon}_k^T S_k \bar{\varepsilon}_k\}$ where S_k is some positive definite matrix.

a.) If the noise sequence $\bar{w}_k$ and $\bar{v}_k$ are Gaussian, zero-mean uncorrelated, and white, then the Kalman filter is the solution to the problem which minimizes the expected value of the weighted error two-norm (i.e., it is the optimal solution)

b.) If $\bar{w}_k$ and $\bar{v}_k$ are zero-mean, uncorrelated, and white, but NOT Gaussian, then the Kalman filter is the best possible linear solution (i.e., the best estimator formulated as a linear combination of the measurements). There may be better nonlinear estimators, but the Kalman filter is the best linear one.

c) The Kalman filter can be modified to solve the problem if $\bar{w}_k$ or $\bar{v}_k$ are correlated or colored.

d.) Various nonlinear filters can approximate the Kalman filter for nonlinear systems, but with no optimality guarantee.

4. K_k / P_k^+ can be pre-computed to save computation → Only in linear Kalman filter!

Divergence Issues:

Kalman Filter assumptions:

- Model is perfectly known (F_k & H_k)
- Process & observation noises perfectly characterized (Q_k & R_k)
- $\bar{w}_k$ and $\bar{v}_k$ are uncorrelated, white, and zero-mean
- Arithmetic and matrix operations are infinite numerical precision

→ Not true in reality! Modeling errors, noise characterization problems, and finite-precision processors

Some simple practical solutions:

1. Increase arithmetic precision
   - Use double precision when you can to reduce numerical errors
   - Not always feasible in embedded systems
2. Use square-root filtering
   - Will cover later on
   - Effectively increases precision, but also increases computation and mathematical complexity
3. Symmetrize $P_k \rightarrow P_k = \frac{P_k + P_k^T}{2}$
   - Some equations for P_k^+ give a mathematically symmetric P_k^+ but not a numerically symmetric P_k^+ in practice
   - Not a silver bullet, but simple enough that you should always do this
4. Use a fading-memory filter
   - Places priority on recent measurements
   - Loses optimality but can restore stability in practice

5. Fictitious process noise
   - Increase Q_k to model uncertainty in system model
   - Allows measurements to drive system performance in the event that our model F_k and G_k are incorrect
   - Will result in suboptimal performance if our model *is* correct

Alternate Kalman Filter Representations (Ch. 6 of book)

There exist several different but mathematically identical formulations of the DT Kalman filter
- You may run into different formulations in the literature
- Some equivalent forms give advantages to precision/stability or computational cost
  - There will always be tradeoffs!

The Sequential Kalman Filter

Recall: $\bar{z}_k$ is $M \times 1$ $\rightarrow$ R_k is $M \times M$

$\rightarrow K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$ requires inversion of $M \times M$ matrix

$\rightarrow$ Can be a problem if M is large or our processor is small (i.e., embedded)

Recall the scalar RLS problems from previous assignments
$\rightarrow$ Each scalar measurement could be processed individually with any H_k matrix (not required to be constant) and any variances for the different measurements

$\rightarrow$ In theory, we can apply the scalar RLS approach to the Kalman filter measurement update step to process each element of $\bar{z}_k$ individually, which only requires scalar operations!

BUT $\rightarrow$ Remember RLS is based on assumption that sequential measurements are uncorrelated
$\rightarrow$ R_k must be diagonal!

→ If R_k is diagonal, we can process each element of $\bar{z}_k$ sequentially and eliminate all matrix inversions

→ Useful if our processor doesn't handle inversions quickly or at all (eg., low-power embedded)

Sequential Kalman Filter:

1. Model: $\bar{x}_k = F_{k-1}\bar{x}_{k-1} + G_{k-1}\bar{u}_{k-1} + \bar{w}_{k-1}$ w/ $\bar{w}_k \sim (0, Q_k)$

   $\bar{z}_k = H_k\bar{x}_k + \bar{v}_k$ w/ $\bar{v}_k \sim (0, R_k)$ → R_k must be diagonal!

2. Initialize $\hat{x}_0^+$ and P_0^+

3. Time-Update/Prediction:

$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1}$$
$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1}$$

} Identical to regular Kalman filter!

4. Measurement-Update

- Let $\hat{x}_{ik}^+$ be the posterior estimate after the i^{th} element of $\bar{z}_k$ is processed
- Let P_{ik}^+ be the posterior covariance after the i^{th} element of $\bar{z}_k$ is processed
- Let z_{ik} be the i^{th} element of $\bar{z}_k$
- Let R_{ik} be the i^{th} diagonal element of R_k → $\text{var}\{z_{ik}\} = \text{var}\{v_{ik}\} = R_{ik}$
- Let H_{ik} be the i^{th} row of H_k → $z_{ik} = H_{ik}\bar{x}_k + v_{ik}$

a.) Initialize sequential processing steps:

$$\hat{x}_{0k}^+ = \hat{x}_k^- \quad \text{and} \quad P_{0k}^+ = P_k^-$$

b.) For $i = 1, \ldots, M$ do the following to compute K_{ik}, $\hat{x}_{ik}^+$, and P_{ik}^+

$$\underbrace{K_{ik}}_{N\times 1} = \underbrace{P_{i-1,k}^+}_{N\times N}\underbrace{H_{ik}^T}_{N\times 1}\left(\underbrace{H_{ik}}_{1\times N}\underbrace{P_{i-1,k}^+}_{N\times N}\underbrace{H_{ik}^T}_{N\times 1} + \underbrace{R_{ik}}_{1\times 1}\right)^{-1}$$

1×1 → Only a scalar inversion! (i.e., division)

Kalman Gain i

OR $K_{ik} = \frac{1}{R_{ik}} P_{ik}^{+} H_{ik}^{T}$

$$\hat{x}_{ik}^{+} = \hat{x}_{i-1,k}^{+} + K_{ik}\left(z_{ik} - H_{ik}\hat{x}_{i-1,k}^{+}\right)$$

(K_{ik}: $N \times 1$; $z_{ik} - H_{ik}\hat{x}_{i-1,k}^{+}$: 1×1)

A posterior state estimate :

$$P_{ik}^{+} = \left(\underline{I} - K_{ik}H_{ik}\right)P_{i-1,k}^{+}\left(\underline{I} - K_{ik}H_{ik}\right)^{T} + K_{ik}R_{ik}K_{ik}^{T}$$

$$= \left(\left(P_{i-1,k}^{+}\right)^{-1} + \frac{1}{R_{ik}}H_{ik}^{T}H_{ik}\right)^{-1} = \left(\underline{I} - K_{ik}H_{ik}\right)P_{i-1,k}^{+}$$

A posterior covariance :

C.) Finalize a posterior state estimate and covariance:

$\hat{x}_{k}^{+} = \hat{x}_{Mk}^{+}$ and $P_{k}^{+} = P_{Mk}^{+}$

What if R_k is not diagonal?

Suppose $R_k = R$ is constant

→ Also, S is orthogonal → $S^{-1} = S^{T}$

→ Jordan form decomposition: Find S such that $R = S\hat{R}S^{-1} \rightarrow \hat{R} = S^{-1}RS$

with $\hat{R}$ a diagonal matrix w/ eigenvalues on diagonal

(note: can use [S, R_hat] = jordan(R) in MATLAB)

→ We can always do this if R is positive definite

Let $\tilde{z}_k = S^{-1}\bar{z}_k = \tilde{H}_k\bar{x}_k + \tilde{v}_k$ w/ $\tilde{H}_k = S^{-1}H_k$ and $\tilde{v}_k = S^{-1}\bar{v}_k$

$\tilde{v}_k$ still zero-mean, so

$$\text{cov}\{\tilde{v}_k\} = E\{\tilde{v}_k\tilde{v}_k^{T}\} = E\{S^{-1}\bar{v}_k\bar{v}_k^{T}(S^{-1})^{T}\} = S^{-1}E\{\bar{v}_k\bar{v}_k^{T}\}S$$

$$= S^{-1}RS = \hat{R}$$

↳ Diagonal!

We can "normalize" $\bar{z}_k$ to get a diagonal covariance!

→ $\tilde{z}_k = S^{-1}\bar{z}_k$ and $\tilde{H}_k = S^{-1}H_k$

Then we can run the sequential Kalman filter using $\tilde{z}_k$ and $\tilde{H}_k$ rather than $\bar{z}_k$ and H_k.

→ But, only useful if R is constant! Jordan form decomposition is computationally taxing and not worth doing every iteration. If R is constant, we can pre-compute S^{-1} and $\tilde{H}_k$.

Summary: Sequential Kalman filter reduces computational load by eliminating matrix inversion from Kalman gain. Only useful if either:
1. R_k is diagonal
2. $R_k = R$ is constant

Note: Standard Kalman filtering in this context is often called "batch filtering"

Information Filtering

Recall: covariance is a measure of uncertainty, where larger covariance implies less certainty → $P = \infty$ means no certainty, $P = 0$ means absolute certainty

The inverse of a covariance matrix (P^{-1}) is the same thing, but in the opposite direction: If P^{-1} is large, we have more certainty, and if P^{-1} is small, we have less certainty

Let $\underline{I = P^{-1}}$ be the information matrix.

→ We can modify the Kalman filter to keep track of I_k instead of P_k

Recall one form of the measurement update to covariance:

$$P_k^+ = \left((P_k^-)^{-1} + H_k^T R_k^{-1} H_k\right)^{-1} \longrightarrow (P_k^+)^{-1} = I_k^+ = (P_k^-)^{-1} + H_k^T R_k^{-1} H_k$$

I_k^+: A posteriori information matrix

$$\rightarrow \boxed{I_k^+ = I_k^- + H_k^T R_k^{-1} H_k}$$

→ Information matrix measurement update

I_k^-: A priori information matrix

Time update?

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

$$\rightarrow \left(P_k^-\right)^{-1} = I_k^- = \left(F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1}\right)^{-1} = \left(F_{k-1} \left(I_{k-1}^+\right)^{-1} F_{k-1}^T + Q_{k-1}\right)^{-1}$$

☹

We can use the matrix inversion lemma to (sort of) simplify. Recall:

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

→ Let $A = Q_{k-1}$, $B = F_{k-1}$, $C = F_{k-1}^T$, and $D = I_{k-1}^+$

$$\rightarrow I_k^- = Q_{k-1}^{-1} - Q_{k-1}^{-1} F_{k-1} \left(I_{k-1}^+ + F_{k-1}^T Q_{k-1}^{-1} F_{k-1}\right)^{-1} F_{k-1}^T Q_{k-1}^{-1}$$

→ Information matrix time update

→ Annoying, but we will discuss advantages shortly...

The Information Filter:

1. Define system model (i.e., determine F_k, G_k, H_k, Q_k, and R_k)
2. Initialize: $\hat{x}_0^+ = E\{\bar{x}_0\}$ and $I_0^+ = \left(E\{(\bar{x}_0 - \hat{x}_0^+)(\bar{x}_0 - \hat{x}_0^+)^T\}\right)^{-1}$
3. For $k = 1, 2, \ldots$

Time-Update:

$$\hat{x}_k^- = F_{k-1} \hat{x}_{k-1}^+ + G_{k-1} \bar{u}_{k-1}$$

$$I_k^- = Q_{k-1}^{-1} - Q_{k-1}^{-1} F_{k-1} \left(I_{k-1}^+ + F_{k-1}^T Q_{k-1}^{-1} F_{k-1}\right)^{-1} F_{k-1}^T Q_{k-1}^{-1}$$

Measurement-Update:

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

$$\hat{x}_k^+ = \hat{x}_k^- - K_k \left(\bar{z}_k - H_k \hat{x}_k^-\right)$$

$$I_k^+ = I_k^- + H_k^T R_k^{-1} H_k$$

Okay, why the matrix inversion lemma for the time-update step?

→ If Q_k is constant, then Q_{k-1}^{-1} can be precomputed. Thus, this form means we only need to do one $N \times N$ matrix inversion at each iteration rather than two.

What is the point of the information filter?

→ In the standard Kalman filter, we must perform one $M \times M$ matrix inversion

→ In the information filter, assuming R_k is constant and thus R_k^{-1} can be pre-computed, we must perform two $N \times N$ matrix inversions

→ If $M >> N$, the information filter provides major computational savings!

Another interesting benefit is that the $P_0^+ = \infty$ case can be implemented exactly in a program since in that case, $I_0^+ = 0$!

> Summary: The information filter can potentially reduce the computational load of filtering, but requires the following conditions:
>
> 1. $M >> N$
> 2. R_k is constant and R_k^{-1} can be pre-computed

Square-Root Filtering

We won't go too in depth with this here, as the derivations will be too time-consuming

Big idea: Suppose $P_k^+ = \begin{bmatrix} 10^6 & 0 \\ 0 & 10^{-6} \end{bmatrix}$

This is fine mathematically. But because of the many orders of magnitude difference, P_k^+ may appear singular (i.e., non-invertible and not a valid covariance matrix) to a computer with limited precision. This will break the Kalman filter.

The idea of square-root filtering is to find an S_k^+ such that $P_k^+ = S_k^+ (S_k^+)^T$

→ S_k^+ is said to be the square root of P_k^+

Then, we keep track of S_k^+ rather than P_k^+.

For $P_k^+ = \begin{bmatrix} 10^6 & 0 \\ 0 & 10^{-6} \end{bmatrix}$, $S_k^+ = \begin{bmatrix} 10^3 & 0 \\ 0 & 10^{-3} \end{bmatrix}$ → We double our precision

→ The square-root filter, and a similar algorithm called the U-D filter, can increase the numerical precision of the Kalman filter when hardware constraints limit the precision. It comes at the cost of increased computational load.

A derivation and more complete discussion of these filters can be found in Chapter 6 of our textbook.

Summary: Kalman filter types

Certainty Representation
1. Covariance Filter
2. Information Filter

Precision Augmentation
1. Standard Kalman filter
2. Square-root filter
3. U-D filter

Measurement Processing
1. Batch filter
2. Sequential filter

→ We can mix and match all these types depending on our application