3-Parameter Kalman Filter for Antenna Motion Model

G. R. Harp
June 16, 2002

As part of the antenna motion feedback system, we need a predictive model of the antenna behavior. Here we describe a simple 3-parameter Kalman filter employed in the motion model. The three state parameters are the position, velocity and acceleration of each drive as a function of time. Only one observable is fed back to this model: the measured position. High reliability position measurements come from the encoder. As a backup for times when the velocity is low (and there are long periods between encoder steps), low reliability position measurements are obtained by integrating the motor velocity.

The azimuth and elevation motors are driven by frequency-controlled stepper motors, and monitored by incremental encoders. We wish to interpolate the antenna position between encoder counts, so we need a model of the system to do this. Such a model also helps us recognize discrepancies between the drive parameters and antenna motion (e.g. if we are driving at one velocity and observe another), for the identification of error conditions in the system.

We assume that the azimuth and elevation drives are completely independent and model them separately. We choose a 3-parameter Kalman filter to model each drive. A Kalman filter minimizes the mean square error (between model and system) under some strict mathematical assumptions: i.e. gaussian error statistics, purely linear system, etc. We don’t claim that the antenna fulfills all these conditions, but the Kalman filter is known to be stable and forgiving to small nonlinear deviations, and provides a well-behaved approach when the state variables are highly correlated. In the present case, we need to model the antenna position and velocity for sure, and acceleration is desired. Clearly, these variables are highly correlated (e.g. position is the integral of velocity).

Model Parameters

Our model defines a state vector (\( \mathbf{x} \)) for the system consisting of the position (\( x \)), velocity (\( \dot{x} \)), and acceleration (\( \ddot{x} \))

---

\( \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} \).

We feed back only one measurement into the model, the measured position, so our measurement vector is
\( \tilde{z} \equiv (x_{\text{meas}}) \).

We shall need a matrix (\( \tilde{H} \)) to project out an estimate of the measurement based on a given state vector. In this case,
\[ \tilde{H} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \quad \text{with} \quad \tilde{z} \sim \tilde{H} \tilde{x}. \]

Next, we need to define the 3x3 error covariance matrix \( \tilde{P} \). The diagonal elements of this matrix correspond to the mean square errors in each of the state parameters. The off-diagonal elements capture the correlations between these errors.

\[
\tilde{P} = \begin{pmatrix}
< \delta x, \delta x > & < \delta x, \delta \dot{x} > & < \delta x, \delta \ddot{x} > \\
< \delta \dot{x}, \delta x > & < \delta \dot{x}, \delta \dot{x} > & < \delta \dot{x}, \delta \ddot{x} > \\
< \delta \ddot{x}, \delta x > & < \delta \ddot{x}, \delta \dot{x} > & < \delta \ddot{x}, \delta \ddot{x} >
\end{pmatrix}.
\]

Another bit of the recipe is the time update matrix \( \tilde{\Phi} \). This is the matrix that takes \( \tilde{x}(t) \) and propagates it forward in time to \( \tilde{x}(t + \Delta t) \):

\[
\tilde{\Phi} = \begin{pmatrix}
1 & \Delta t & \frac{1}{2} \Delta t^2 \\
0 & 1 & \Delta t \\
0 & 0 & 1
\end{pmatrix} \quad \text{with} \quad \tilde{x}(t + \Delta t) = \tilde{\Phi} \tilde{x}(t).
\]

Then comes the process variance matrix \( \tilde{Q} \), which is the way that we inform the model of its own limits. If the model were perfect, then it would rapidly converge to the real system, at which time system measurements would increasingly be ignored. But our model doesn’t have any jerk (\( \dddot{x} \)) or higher derivatives in it, so we know that it is flawed. We model the flaws by saying that there is an error in the acceleration that grows with time in between measurement updates. Errors in acceleration are fed back into velocity and position in a natural way. Thus,

\[
\tilde{Q} = \begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & \xi \Delta t
\end{pmatrix}, \quad \text{where} \quad \xi \quad \text{is a parameter to be determined.}
\]
Finally, we define the measurement error covariance $\tilde{R}$, which says how much we trust our measurements. Since $\tilde{z}$ has only one element, $\tilde{R}$ is a 1x1 matrix. Encoder measurements have a very small associated error (a fraction of one encoder tick) and $\tilde{R}$ is equal to the square of this error. However, there may be times when the drive is moving extremely slowly (e.g. elevation drive at transit). In this case, we can integrate the stepper motor velocity to get a more timely estimate of position. In the latter case $\tilde{R}$ will be larger.

**Time Update**

Given the definitions above, we notice that a complete specification of the model consists of the state $\tilde{x}$ and its errors $\tilde{P}$. It often happens that we know $(\tilde{x}, \tilde{P})$ at one time, but wish to estimate their values at some later time. Such time-updated values can be found from

$$\tilde{x}(t + \Delta t) = \Phi(\Delta t) \tilde{x}(t) \quad \text{and}$$

$$\tilde{P}(t + \Delta t) = \Phi(\Delta t) \tilde{P}(t) \Phi^T(\Delta t) + \tilde{Q}(\Delta t).$$

**Measurement Update**

When a new measurement comes in, we need to update $(\tilde{x}, \tilde{P})$ with its value. We define the Kalman matrix

$$\tilde{K} = \tilde{P} \tilde{H}^T (\tilde{H} \tilde{P} \tilde{H}^T + \tilde{R})^{-1}.$$ 

Notice that if the filter is set up such that only one variable is measured (as here), the matrix inversion is a simple reciprocal. This is a substantial simplification since we don’t have to worry about matrix singularities and we never do any matrix inversions at all. Even when you measure more than one parameter at a time, if the measurements are uncorrelated then it is desirable to update each parameter independently rather than as a group (this is often done).

Given that definition, the measurement update equations are

$$\tilde{x}^+ = \tilde{x}^- + \tilde{K} (\tilde{z} - \tilde{H} \tilde{x}^-) \quad \text{and}$$

$$\tilde{P}^+ = \tilde{P}^- - \tilde{K} \tilde{H} \tilde{P}^-,$$

where the plus superscripts refer to after the measurement update and the minus subscripts refer to before the measurement update.
To get a better understanding of what is happening, study the equation for the state update. The term in parentheses represents the difference between our measurement and the prediction of the model. Speaking loosely, if our measurements are very noisy, then $\tilde{K}$ is small and the state vector is changed only a little bit by a single measurement. On the other hand, if our measurements are accurate, then $\tilde{K}$ is of order unity and our state follows the measurements closely.

**Appendix I**

Here we show some implementation details. Rather than cranking up a bunch of matrix operation classes to take care of the model, we can hand code them. Because there are few parameters (and no inversions) this is easy. For example, in this case

$$\tilde{K} = \frac{1}{P_{00} + R} \begin{pmatrix} P_{00} \\ P_{10} \\ P_{20} \end{pmatrix}, \text{ and}$$

$$\tilde{K} \tilde{H} = \frac{1}{P_{00} + R} \begin{pmatrix} P_{00} & 0 & 0 \\ P_{10} & 0 & 0 \\ P_{20} & 0 & 0 \end{pmatrix}.$$
\[ \Omega_{01} = (P_{02} + P_{11}) \Delta t + (P_{12} + \frac{1}{2} P_{21}) \Delta t^2 + \frac{1}{2} P_{22} \Delta t^3 \]
\[ \Omega_{11} = (P_{12} + P_{21}) \Delta t + P_{22} \Delta t^2 \]
\[ \Omega_{21} = P_{22} \Delta t \]
\[ \Omega_{02} = P_{12} \Delta t + \frac{1}{2} P_{22} \Delta t^2 \]
\[ \Omega_{12} = P_{22} \Delta t \]
\[ \Omega_{22} = \xi \Delta t \]

Appendix II
Here we show some examples. We have built a software model of a motor drive. It drives the motor along a sinusoidal path. The parameters of the path (i.e. max velocity, max acceleration) were chosen to be similar to those for a real antenna. First lets examine the results.

Fig. 1: Example of how the model follows a sinusoidal input position signal.
Fig. 2: The difference between the “actual” position and the model position. The maximum error occurs at startup time where there is an artificial discontinuity in the velocity. The minimum error occurs when the motor is driving quickly (where it get the maximal number of measurements per second). In all cases, the error is much less than 1°.
Fig. 3: The model keeps track of velocity, which is followed very well.

Fig. 4: The highest derivative the model keeps track of is acceleration. For the chosen set of parameters, the model acceleration is getting a little hairy.
There is a trade off relating to how closely the “noise” in the acceleration as determined by the model, and how closely the position is followed. This trade off comes in the setting of the process variance matrix $\tilde{Q}$ (it has only one nonzero element). In the present example, I have chosen a value equal to the maximum expected jerk that the antenna will see, which is close to the theoretically “correct” value. By making $\tilde{Q}$ smaller, it is possible to significantly tighten the agreement between the model acceleration and the real acceleration\(^2\). However, this comes at the cost of worsening the position agreement. When tweaking the real antenna drives, the $\tilde{Q}$ matrix is the parameter that needs the most attention.

\(^2\) Remember that $\tilde{Q}$ represents the error in our model. If we make $\tilde{Q}$ smaller, this means that we trust our model more. Hence the model acceleration doesn’t jump around as much as it tries to follow tiny fluctuations in the input measurements.