Introduction

This report describes my experiments on extended Kalman filter behavior, using Dr. Stanley Schmidt's simulation of a 16-state filter from his digital book, Aided Inertial Navigation Systems. I used the simulation to produce Monte Carlo plots for a variety of Kalman filter situations; I overlaid one-sigma covariance boundary plots on the Monte Carlo plots where appropriate.

Kalman filter divergence

To investigate the case in which the extended Kalman filter puts too much weight on its own propagated statistics, I set the simulation's x- and y-axis wind gust error parameters to zero, while setting the true values to 0.02. These settings eliminated wind-gust modeling from the Kalman filter's model, while including substantial wind gust effects in the simulation. I then plotted ten Monte Carlo runs of x-position error vs. time; as seen in Figure 1, removing the estimated effects of wind gusts from the Kalman filter model caused the filter's estimated position to become incorrect. When I overlaid a plot of x-position error standard deviation (also seen in Figure 1) I could see that although the covariance of the error was headed toward zero, the actual estimates were diverging. Thus, the Kalman filter believed erroneously that its measurements were correct, resulting in "Kalman filter incest".
I then attempted to fix the problem by changing the Kalman filter's wind-gust model parameters to conservative values, setting them to 0.04 (twice the true wind gust values of 0.02). Figure 2 shows the results of the Monte Carlo runs and the standard deviation overlay for the new simulation parameters. From the resulting plots, I could see that the standard deviation was large enough to contain most of the x-position error plots, indicating that the conservative modeling of wind gusts kept the filter's covariance large relative to the measurements, thereby preventing divergent behavior. In an actual EKF implementation, the wind-gust model would have to be based on a conservative estimate of the actual wind gusts experienced in the real-world environment.
I next investigated the situation where the extended Kalman filter accepted external updates to its estimates on a regular time interval. To set up the Kalman filter simulation, I set the time update period to one second, and the number of updates per measurement to 50. Additionally, to isolate the desired effect from trajectory variations, I set the velocity azimuth to 90 degrees, the wind gust treatment flag to one, and the velocity treatment flag to zero. Figure 3 shows the results of my Monte Carlo simulations for y-velocity, and an overplot of y-velocity standard deviation. The discontinuities that occur when the external updates are received by the Kalman filter at 50-second intervals are clear.
To look at the effect of “overweighting” valid external updates, I set the wind gust uncertainties (x- and y-wind gust error std) to 0.1. My resulting plot of y-velocity error and standard deviation overlay is shown in Figure 4; the updates are considered disproportionately valid by the Kalman filter, causing it to drive the estimate to the update value every time an update is received. In the cockpit, this behavior would be manifest by sudden, discontinuous changes in the information displayed to the pilot, and potentially erratic commands being sent to the autopilot.

**Figure 4 - y-velocity error with external update every 50 seconds, and excessive update weighting**

**Measurement Rejection**

After setting the outlier treatment flag to 10 (to cause 20% of the simulation's measurements to be incorrect, with errors ten times the normal level), I investigated the effects of different levels of bad measurement rejection on the behavior of the extended Kalman filter. I first performed Monte Carlo simulations without any error rejection; the results are shown in Figure 5. The overlay of the standard deviation shows that the bad measurements are not reflected in the covariance, which decreases as it is propagated over time. An extended Kalman filter such as this would not provide any warning to the pilot that the filter estimates were diverging.
I then changed the measurement rejection ratio to a value of three, causing the filter simulation to reject measurements outside a three-sigma region. From the plot shown in Figure 6, it can be seen that about ten percent of the measurements were rejected by the filter, thereby keeping the estimates much closer to the standard deviation boundaries.
Figure 7 shows the plotted results of changing the rejection ratio to a value of one, causing the filter to reject measurements outside a one-sigma region. As seen in the plot, about 35% of the measurements were rejected. The standard deviation is comparable to that of the 3-sigma rejection level, but a larger percentage of the estimates fall within the borders.

![Figure 7 - y-velocity plot with 10x bad measurements and 1-sigma rejection](image)

Keeping the rejection ratio at one, I changed the outlier treatment flag to 20, thereby doubling the error level of the bad measurements. From the results plotted in Figure 8, it is seen that a greater number of measurements were rejected, but the non-rejected measurements produced essentially the same estimates as before. For comparison, I also plotted the results at this wild outlier level with a 3-sigma rejection level -- the plot is shown in Figure 9. There weren't nearly as many rejected measurements (17 vs. 61) but as seen by the standard deviation plot, the propagated covariance reduced with time.
If the pilot was provided with an override button to force the Kalman filter to accept rejected measurements known to be valid, the Kalman filter’s rejection level could be set tighter than it otherwise could be, and would allow the filter to be updated successfully even if the filter’s estimate drifted off considerably. Since this feature would (hopefully) be used infrequently, but would be essential when actually needed, it would probably be of more practical value than providing the pilot with the ability to...
change the rejection level. Changing the rejection level could actually cause the filter to accept too many bad measurements without the pilot being aware of it.

Effect of Reference Station Geometry on Solution

I changed the extended Kalman filter simulation’s reference station positions to be at zero on the x-axis, and separated on the y- and z-axes. Running Monte Carlo simulations (as shown in Figures 10 and 11) demonstrated that the filter was often divergent in its estimation of x-axis position, while it was correct in the y-axis position estimates.

![Figure 10 - x-position with reference stations aligned on y-axis](http://locus.aero.calpoly.edu/aero551/Chad_Frost/ains-kf/ekf-report.html)
Re-orienting the reference stations to be aligned along the x-axis, I repeated the Monte Carlo simulations. From the plots in Figures 12 and 13, I found that the y-position estimates were good, while the x-position estimates were sometimes divergent. These results demonstrated why GPS receivers look at the DOP values for the combination of satellites they are receiving, and select the combination affording the best DOP. A GPS receiver that does not perform such checking, or that can only see satellites that are in poor alignment, could send very inaccurate updates to a Kalman filter.
Figure 12 - x-position with reference stations aligned on x-axis

Figure 13 - y-position with reference stations aligned on x-axis