Engineer's Guide to Debugging an Indirect Extended Kalman Filter

Dr. Randall Christensen

August 22, 2019

1 Introduction

This document is an attempt at capturing my methods of debugging an indirect, extended Kalman filter. My objective is to provide a procedure that systematically verifies correct implementation of each piece of the filter, so that when you glue everything together, there are no surprises. Two overriding principles are "divide and conquer" and "never swing for the fence".

2 Indirect Extended Kalman Filter Equations

In the following discussion, it is assumed that the model of the truth state is identical to the model used to design the Kalman filter. If this is not the case, a mapping from truth states to navigation states must be developed and used throughout the validation steps when comparing estimated states to truth states.

The design model for the extended Kalman filter consists of nonlinear models of the *continuous* dynamics and *discrete* measurements.

$$\dot{\underline{\mathrm{x}}} = \underline{\mathrm{f}}\left(\underline{\mathrm{x}}, \underline{\mathrm{u}}\right) + B\underline{\mathrm{w}} \tag{1}$$

$$\tilde{\underline{z}} = \underline{\mathrm{h}}\left(\underline{\mathrm{x}}\right) + G\underline{\nu} \tag{2}$$

where dependence on time is implied.

In the extended Kalman filter, the nonlinear models are linearized about the current state estimate to develop linear perturbation models of the state dynamics and measurements

$$\delta\dot{\underline{\mathrm{x}}} = F\left(\hat{\underline{\mathrm{x}}}\right)\delta\underline{\mathrm{x}} + B\underline{\mathrm{w}} \tag{3}$$

$$\delta\tilde{\underline{z}} = H\left(\hat{\underline{\mathrm{x}}}\right)\delta\underline{\mathrm{x}} + G\underline{\nu} \tag{4}$$

where the process noise $\underline{\mathrm{w}}$ and the measurement noise $\underline{\nu}$ are zero mean, white noise sources with power spectral density and variance defined by the following

$$E\left[\underline{\mathrm{w}}\left(t\right)\underline{\mathrm{w}}^{T}\left(t'\right)\right] = Q\delta\left(t - t'\right) \tag{5}$$

$$E\left[\underline{\nu}_{l}\underline{\nu}_{m}^{T}\right] = R\delta\left[l - m\right] \tag{6}$$

The equations used for propagation of the estimated state $\hat{\underline{\mathrm{x}}}$ and the error state covariance P are

$$\dot{\hat{\underline{\mathrm{x}}}} = \underline{\mathrm{f}}\left(\hat{\underline{\mathrm{x}}}, \underline{\mathrm{u}}\right) \tag{7}$$

$$\dot{P} = F\left(\hat{\underline{\mathrm{x}}}\right)P + PF^{T}\left(\hat{\underline{\mathrm{x}}}\right) + BQB^{T} \tag{8}$$

The updated error state vector $\delta\underline{\mathrm{x}}$ and the associated covariance are

$$\delta\hat{\underline{\mathrm{x}}}^{+} = K\left[\tilde{\underline{z}} - \underline{\mathrm{h}}\left(\hat{\underline{\mathrm{x}}}^{-}\right)\right] \tag{9}$$

$$P^{+} = \left[I - KH\left(\hat{\underline{\mathrm{x}}}^{-}\right)\right]P^{-}\left[I - KH\left(\hat{\underline{\mathrm{x}}}^{-}\right)\right]^{T} + KGRG^{T}K^{T} \tag{10}$$

where the Kalman gain is defined as

$$K = P^{-}H^{T}\left(\hat{\underline{\mathrm{x}}}^{-}\right)\left[H\left(\hat{\underline{\mathrm{x}}}^{-}\right)P^{-}H^{T}\left(\hat{\underline{\mathrm{x}}}^{-}\right) + GRG^{T}\right]^{-1} \tag{11}$$

In addition to the preceding equations which define the Kalman filter, three useful mappings can be defined between the true state $\underline{x}$, the estimated state $\hat{\underline{x}}$, and the error state $\delta\underline{x}$.

$$\underline{x} = \underline{l}\left(\hat{\underline{x}}, \delta\underline{x}\right) \tag{12}$$

$$\hat{\underline{x}} = \underline{m}\left(\underline{x}, \delta\underline{x}\right) \tag{13}$$

$$\delta\underline{x} = \underline{n}\left(\hat{\underline{x}}, \underline{x}\right) \tag{14}$$

Equation 12 is used in the process of linearizing the design model. It can also be utilized to correct the navigation state given the estimated error state $\delta\hat{\underline{x}}^{+}$. Equation 13 is derived from equation 12 and can be used to insert errors into the state estimates in a way that is consistent with equation 12. Equation 14 is also derived from equation 12 and can be used to calculate the estimation errors consistent with equation 12.

3 Verification Steps

This section lists several steps which verify separate components of the Kalman filter. It is assumed that a Monte Carlo simulation has been completely implemented (i.e. initialization, propagation, Kalman update, data saving, etc).

3.1 Error State Vector

Equations 12 to 14 are important tools in the following verification steps, and thus need to be verified themselves. For lack of a better method, the following procedure tests that they are consistent with each other, but does not necessarily verify correct definition of errors. For this verification step, do the following:

1. Begin with zero error in the navigation state vector.
2. Define an error state vector $\delta\underline{x}$.
3. Inject estimation errors into the state estimates using equation 13.
4. Compute the estimation errors using equation 14 and verify that it is equal to the defined error state vector $\delta\underline{x}$.
5. Use the computed estimation errors from the previous step to correct the state estimates using equation 12.
6. Verify that the errors have be removed

3.2 Nonlinear State Propagation and Nonlinear Measurement Modeling

In the absence of process noise and Kalman updates, the state propagation equation 7 should produce the same results as the truth state propagation. Furthermore, in the absence of measurement noise ν, the residual $\tilde{\underline{z}} - \underline{h}\left(\hat{\underline{x}}^{-}\right)$ in equation 9 should be zero. Thus for this verification step, do the following:

1. Set the process noise $\underline{w}$, and the measurement noise ν to zero. Note that the matrices Q and R are not set to zero, because this could cause numerical problems in the Kalman filter.
2. Set the Kalman gain K to zero to prevent state or covariance updates.
3. Set the initial estimation errors to zero.
4. Run a single simulation for a "sufficient" amount of time.
5. Compute estimation errors using equation 14.
6. Plot estimation errors vs. time.
7. Plot residuals vs. time.

This step verifies that the state propagation in equation 1 matches the navigation state propagation in equation 7. In the more general case where the truth model is different than the design model, this step verifies that equation 7 is consistent with the truth models. This step also verifies that model used to synthesize measurements $\tilde{z}$ from the design (or truth) states in equation 2 is consistent with the model used to predict the measurement using the navigation states in equation 9.

3.3 Linear Error State Modeling

The perturbation model in equation 3 is a linear approximation to the nonlinear model in equations 1 and 7. Thus, to first order, errors propagating through equations 1 and 7 should agree with errors propagating through equation 3. Thus for this verification step, do the following:

1. Set the process noise $\underline{\mathrm{w}}$, and the measurement noise ν to zero. Note that the matrices Q and R are not set to zero, because this could cause numerical problems in the Kalman filter.
2. Set the Kalman gain K to zero to prevent state or covariance updates.
3. Set the time constant of first-order Markov process to 1/10 of the simulation time (for higher order systems, set the bandwidth $= 10/T_{sim}$)
4. At the beginning of the simulation, define an error state vector $\delta\underline{\mathrm{x}}$.
5. Inject estimation errors into the state estimates using equation 13.
6. Over one Kalman cycle, propagate the error state vector using equation 3 and the estimated state using equation 7.
7. Compute the estimation error at the end of the Kalman cycle using equation 14 and compare with the propagated error state vector. The difference should be "small".

This step verifies that the F matrix used to propagate the covariance in equation 8 is implemented correctly.

3.4 Linear Measurement Modeling

Similar to the previous step, equation 4 is the linear approximation to the residual $\tilde{\underline{z}} - \underline{\mathrm{h}}\left(\hat{\underline{\mathrm{x}}}^{-}\right)$ used in equation 9. Thus error injection can be used to validate the H matrix used in the calculation of the Kalman gain in equation 11. Thus for this verification step, do the following:

1. Set the process noise $\underline{\mathrm{w}}$, and the measurement noise ν to zero. Note that the matrices Q and R are not set to zero, because this could cause numerical problems in the Kalman filter.
2. Set the Kalman gain K to zero to prevent state or covariance updates.
3. Run the simulation to the first Kalman update.
4. Inject estimation errors into the state estimates using equation 13.
5. Compute the residual $\tilde{\underline{z}} - \underline{\mathrm{h}}\left(\hat{\underline{\mathrm{x}}}^{-}\right)$ and compare it to the linear approximation of the same quantity using equation 4. The difference should be "small".

This step verifies correction implementation of the H matrix.

3.5 Covariance Propagation

So far, you have verified several components of the Kalman filter. This has been done with all noise sources turned off. In this step, you will test the propagation of process noise. For small errors, the propagation of the error covariance using equation 8 should match the ensemble statistics of errors propagated through the nonlinear system in equation 1. Thus for this verification step, do the following:

1. Set the measurement noise ν to zero. Note that the R matrix is not set to zero, because this could cause numerical problems in the Kalman filter.
2. Set the Kalman gain K to zero to prevent state or covariance updates.
3. Set the time constant of first-order Markov process to 1/10 of the simulation time (for higher order systems, set the bandwidth $= 10/T_{sim}$)
4. Set the initial estimation errors to be consistent with the initial covariance P.
5. Run a Monte Carlo simulation of ~200 samples for a "sufficient" amount of time.

6. Create a hair plot of the estimation errors for ALL states and verify that the ensemble statistics are consistent with the propagated covariance and zero mean.

This step verifies correct implementation of the covariance propagation in equation 8. It also verifies correct implementation of the process noise coupling matrix B and that you have synthesized process noise $\underline{w}$ consistent with the Q matrix in equation 8.

3.6 Estimation Capability

At this point, you have done enough verification to trust nearly all the terms involved in the Kalman update equations. For a properly-tuned Kalman filter with "small" estimation errors, the error covariance P should accurately represent the ensemble statistics of the state estimation errors (which should be zero mean). In addition, statistics of the residual computed in equation 9 should be zero mean with a covariance of $H\left(\underline{\hat{x}}^{-}\right)P^{-}H^{T}\left(\underline{\hat{x}}^{-}\right)+GRG^{T}$ from equation 11. Thus for this verification step, do the following:

1. Set the initial estimation errors to be consistent with the initial covariance P.
2. Set all parameters to be representative of your system (e.g. time constants, noise values, etc)
3. Run a Monte Carlo simulation of ~200 samples for a "sufficient" amount of time and with a state trajectory that is representative of your application.
4. Create a hair plots of the estimation errors for ALL states and verify that the ensemble covariance is consistent with the propagated covariance.
5. Create residual plots for ALL measurements and verify that the ensemble covariance is consistent with the $H\left(\underline{\hat{x}}^{-}\right)P^{-}H^{T}\left(\underline{\hat{x}}^{-}\right)+GRG^{T}$

This steps verifies correct implementation of equations 9 to 11. It also verifies correct implementation of the measurement noise coupling matrix G and that you have synthesized measurement noise $\underline{\nu}$ consistent with the R matrix in equations 10 and 11.

4 Additional tips

- Overly-frequent Kalman updates can mask bugs in the propagation of the covariance.
- Over-large covariances may violate the assumptions of linearization.
- Others?