# Question #4

## Implementing Linearized Kalman Filter (LKF)

To implement the LKF for this Cooperative Localization problem we define an LKF function which performs the prediction [Equation ( 1 )] and correction [Equation ( 2 )] step.

The function takes in the ground truth values (state dynamics and measurements), the nominal state and measurement trajectories without process or measurement noise, the process noise covariance () for tuning, and initial values for and . The function outputs the state estimates (), measurement estimates and estimation errors ( and ).

|  |  |  |
| --- | --- | --- |
|  |  | ( ) |

|  |  |  |
| --- | --- | --- |
|  |  | ( ) |

This function is included in a Monte Carlo run which feeds the ground truth models, nominal state and measurement trajectories, and the DT state space matrices linearized around the nominal trajectories () according to the methods from Question #2 and Question #3.

The truth model is a simulated run of the nonlinear model with the process and measurement noise generated using the covariances uploaded on Canvas. We seed for the ground truth values for every Monte Carlo run from and then and then for every subsequent values the process noise is obtained from covariance matrix ‘Qtrue’.

## Truth Model Testing (TMT) for LKF

For the TMT we do 50 Monte Carlo () runs. This is an appropriate number of Monte Carlo runs as we will have enough data sets for performing an unbiased NEES and NIS test.

Additionally, for the multiple runs in the TMT we feed the LKF the following initial values:

|  |  |  |
| --- | --- | --- |
|  |  | ( ) |

These initial conditions were selected for the following reasons: we are given the nominal and truth state trajectories, and hence the initial perturbation can be derived from those values; furthermore, we have a sufficient degree of certainty of our initial state perturbation such that the state perturbation covariance is finite ().

In addition to this, in the LKF, the covariance matrix for the measurement noise () is set to be the one uploaded on canvas, i.e. Rtrue. As that information is generally known to us from the sensors being used.

## Tuning

There are two aspects that we need to keep in mind while tuning the Kalman filter. First, we need to make sure that the Kalman filter ‘works properly’, i.e. the estimate error averages at 0, and the bounds are converging. Second, we also look at the NEES and NIS chi-square tests and make sure they are within the confidence intervals.

We calculate the confidence intervals on MATLAB with significance level . We chose this significance level to provide a less stringent condition for proving whether the LKF is doing its job (i.e. having a low enough false-alarm probability). This was because we found that the LKF is not good enough for estimating this nonlinear system. The bounds are calculated using the MATLAB function ‘chi2inv’. As a result, for Monte Carlo runs, states, measured values, the chi-square confidence bounds are:

|  |  |  |
| --- | --- | --- |
|  |  | ( ) |

While tuning the LKF we vary our predicted process noise covariance matrix () until the two conditions in the beginning of this subsection are satisfied. We first start with and run the truth model tests and check if the KF works and the NEES and NIS tests are satisfied. The result showed that the neither of the conditions were met, hence we moved to then to and tweaked the main diagonal parameters in this way until the conditions were sufficiently satisfied.

In the end, the process noise covariance matrix for the LKF that best satisfied the conditions was found to be:

|  |  |  |
| --- | --- | --- |
|  |  | ( ) |

Before we move forward, there are some points that need to be highlighted regarding the result. First, the NEES test COULD NOT be satisfied for any selection of regardless of the magnitude of the covariance matrix or selection of initial conditions. The KF was debugged thoroughly, and we could not find any method to alleviate this problem. Second, the NIS test could be satisfied, however periodically at every ~25 seconds the chi-square values exceed the bounds. Although, we managed to bring most of the points for within the bounds in Equation ( 5 ). Finally, it was also found that regardless of the tuning methods employed, the state estimate errors corresponding to North position and East position of the UGV (i.e. and ) are unstable and their bounds do not converge.

The best results obtained after tuning the LKF are shown in the following plots. Figure 1 is a plot showing the NEES and NIS chi-square tests for the Monte Carlo runs with the LKF. Figure 2 is a plot of the state estimate errors from a sample Monte Carlo run with LKF under the same conditions and Figure 3 shows the actual state estimates and the state ground truths for the very same sample run.

First looking at Figure 1, we can see that the NIS test is sufficiently satisfied except for every ~25 seconds where the points are outside the bounds. Upon further investigation, this was found due to the covariance matrix used for the LKF. In the matrix, the elements directly corresponding to the aircraft’s North position and East position values are greater than the other elements that are a combination of the UGV’s and UAV’s together. For instance, elements . And the UAV has a control input of which corresponds to a period of 25 seconds. Furthermore, the NEES test is not satisfied and the problem could not be alleviated regardless of the tuning method. This issue was found to be because the LKF could not accurately ascertain the state evolution of the UGV ( and ).

Also note, that the state estimate errors for and are sufficiently low, except of the time points where the state peaks, and the state estimation covariance () also converges for the UAV’s states. The estimate errors for and also tend to increase at points where the angles are wrapped to within the bounds.

Referring to Figure 2 and Figure 3, we can see that and estimates are barely close in periodic intervals. The LKF fails in estimating these two states. Furthermore, another observation that is that the covariance for the state estimates (i.e. ) for and are oscillating and not converging. Selecting a that alleviates this problem affected the other states adversely, and hence it was left this way. The oscillation also has a period of ~25 seconds and the bounds for is out of phase with .

## Comment on LKF for Estimating Nonlinear System

The tuning procedure was arduous and complicated. However, after multiple attempts, we can conclude that the LKF is not a good enough filter to estimate the nonlinear system for cooperative localization.

We derived this conclusion because we were not able to tune the LKF to satisfy the two conditions: the LKF ‘works properly’ and satisfies the NEES and NIS chi-square tests.

This section went into the methodology used to implement the LKF and the process used to tune it. We also went into the issues faced when tuning and the compromises and assumptions made.

The LKF failed the NEES test and barely passed the NIS test, indicating that the state estimates obtained from this filter are not close enough to the ground truths. This was even after multiple attempts in tuning the filter.

Referring to the UAV’s states, it seems that the LKF starts to mis-estimate its states when the aircraft is moving near the boundaries of its domain in and . However, at all other points, the LKF provides a decent estimate for the UAV’s state. However, the LKF completely fails to successfully estimate the states for UGV. This is credited to the fact that we can directly observe the UAV’s states, while the UGV’s states are only derived indirectly through trigonometric equations.

The LKF fails to estimate the states and for an extended periods and this in and of itself disqualifies the LKF.

Finally, the bounds, or covariance, for and fail to converge over time, which also disqualifies the LKF. We tried to fix this problem by modifying the initial conditions and elements, however in the cases where we got the bounds to converge, the bounds for and exhibit the same oscillating property, and the NEES and NIS tests were worse off. Hence we selected the conditions in Equation ( 3 )( 4 )( 5 ) as a compromise.

Finally, we also tested the filter by checking whether angle wrapping between operations effected or improved the results (i.e. wrapping the nominal and ground truths before feeding into LKF). The results were slightly worse off, because the filter would fail to estimate the states at the point where the angle wrapped back to within the intervals. Hence angles are not wrapped until after the estimate is obtained through the filter.

The MATLAB codes for the LKF implementation, the Monte Carlo runs, and the NEES and NIS tests are attached in Appendix D.

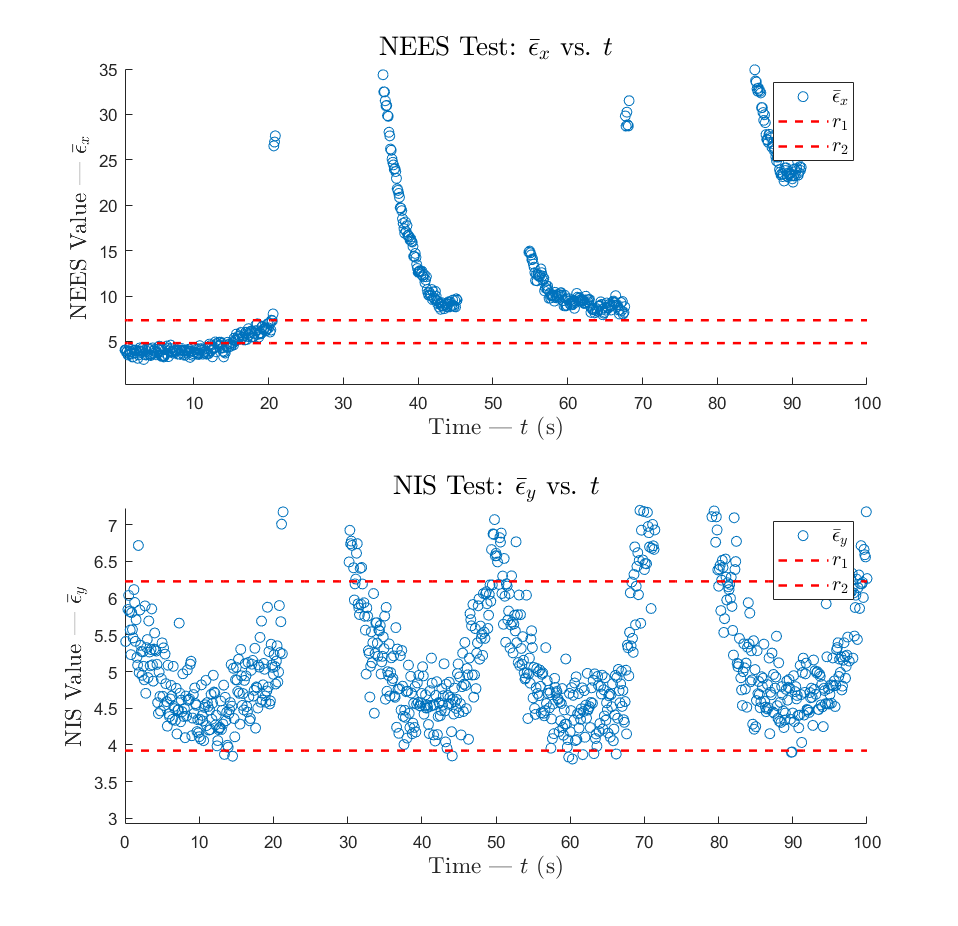


Figure – NEES and NIS chi-square test for LKF Monte Carlo runs with Equations ( 3 )( 4 )( 5 ) applied

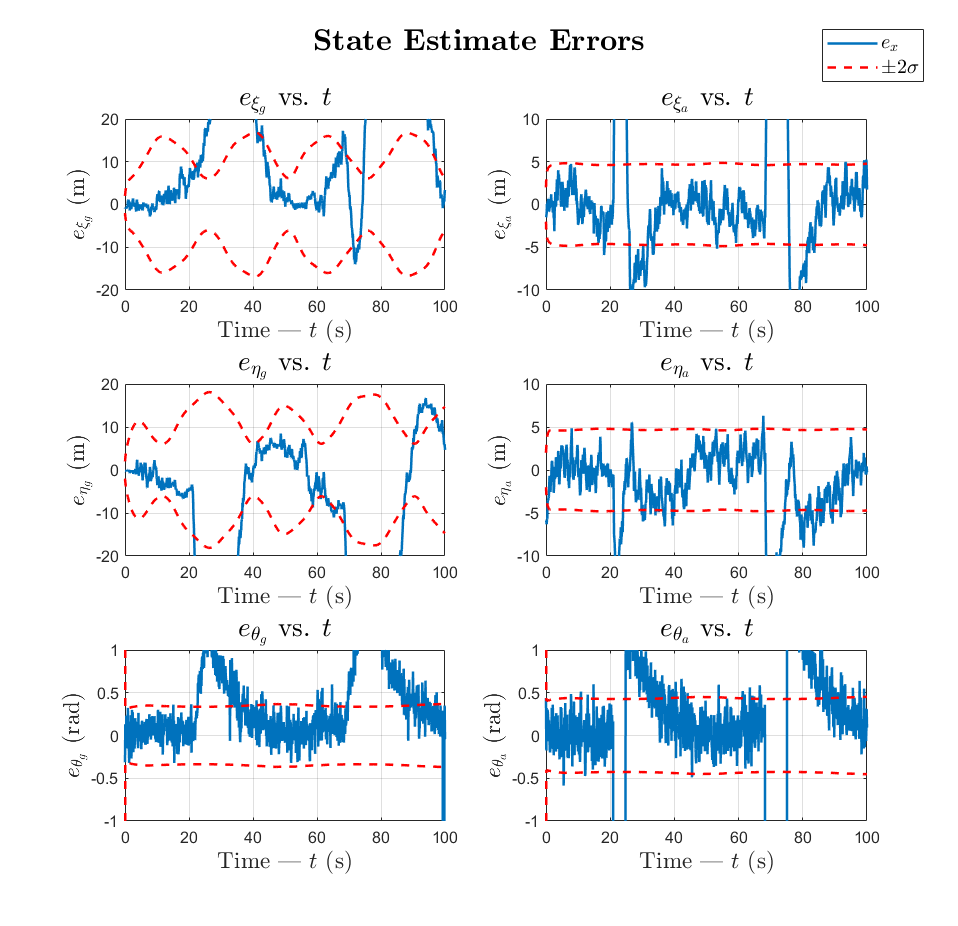


Figure – State estimate errors and bounds of sample LKF Monte Carlo run with Equations ( 3 )( 4 )( 5 ) applied

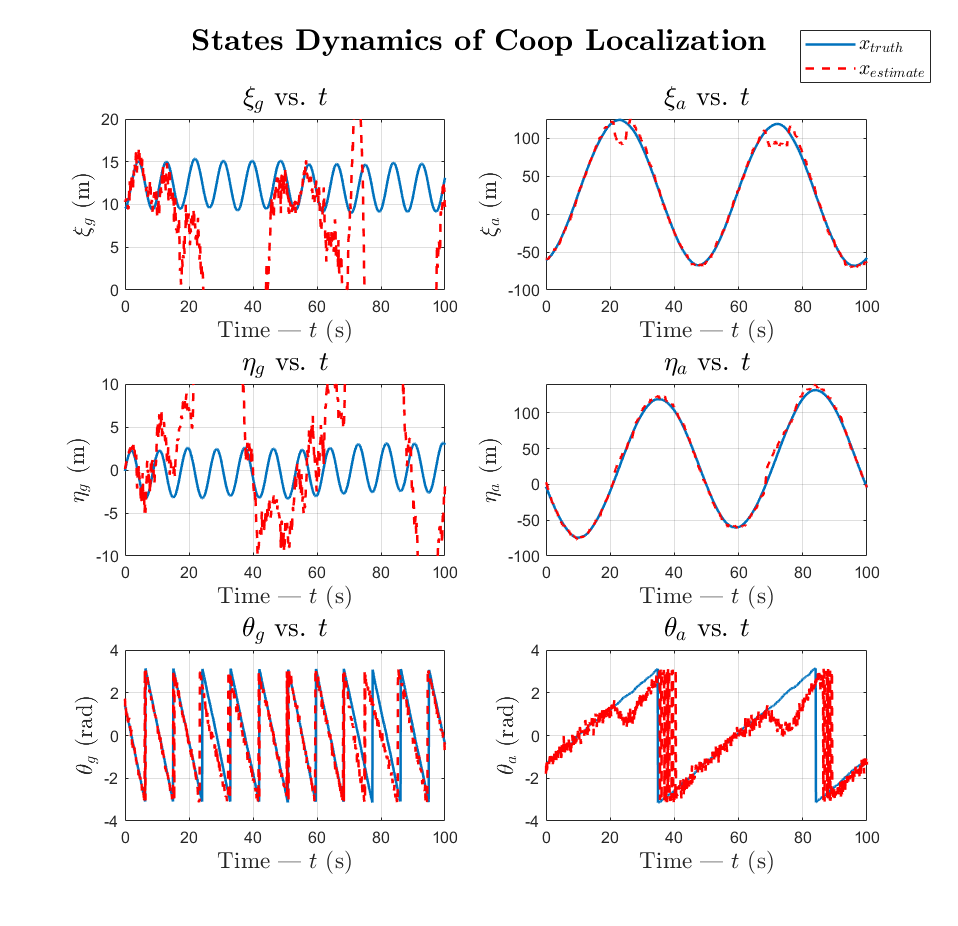


Figure – State estimates and state ground truth of sample LKF Monte Carlo run with Equations ( 3 )( 4 )( 5 ) applied