ON THE AUTOMATIC GENERATION OF RECURSIVE ATTITUDE DETERMINATION ALGORITHMS

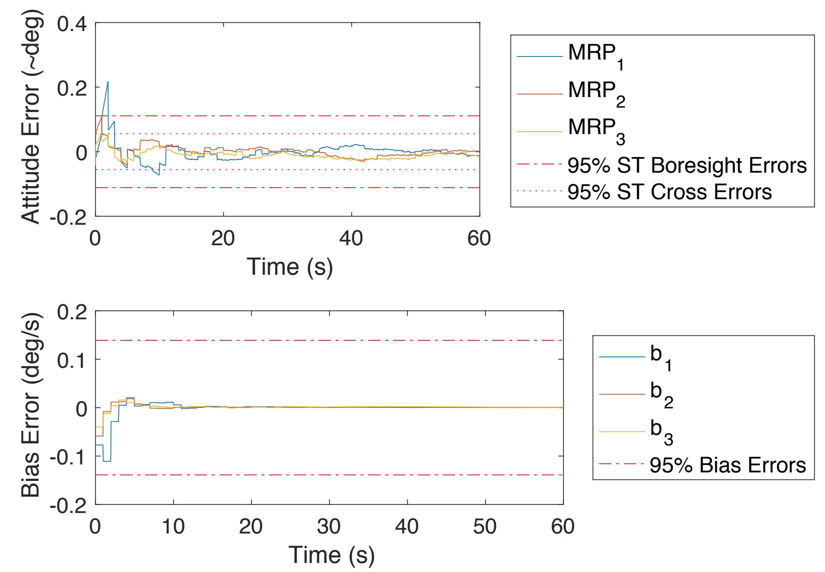
Presented at the AAS GN&C Conference in Breckenridge, CO, on February the 7th, 2017

by Tucker McClure @ An Uncommon Lab

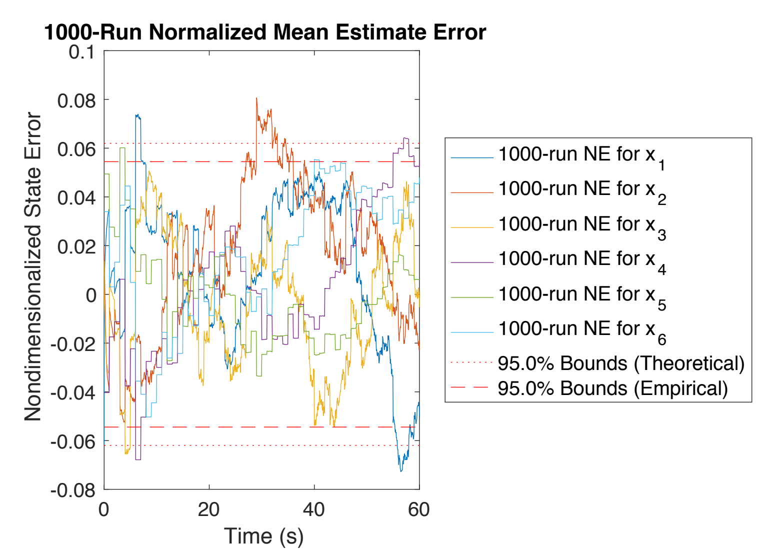
EXAMINE METHODOLOGY

- Create a filter early with very little wasted effort.
- Create a more mature filter later.
- Try variations on the filter.

Early Single Run



Later Mature Results



CLASSIC ATTITUDE ERROR & BIAS ESTIMATOR

- Filter state is attitude error and gyro bias
- Gyro for propagation
- Star tracker for attitude measurements

This example is kept simple to focus on the process, but the process works well when the problem is more complicated.

WHAT ARE THE DEVELOPMENT PATHS FOR DIFFERENT FILTERS?

<table>
<tr><th>UKF</th><th>EKF</th><th>UDKF</th></tr>
<tr><td colspan="3">Spacecraft Simulator</td></tr>
<tr><td colspan="3">Filter Wrapper (Attitude Propagation)</td></tr>
<tr><td>Propagation & Observation Functions</td><td colspan="2">Propagation Jacobian & Effective Process Noise</td></tr>
<tr><td>UKF Implementation</td><td>EKF Implementation</td><td>UDKF Implementation</td></tr>
<tr><td>Unit Testing</td><td>Unit Testing</td><td>Unit Testing</td></tr>
<tr><td colspan="3">Integration & Testing in Simulation</td></tr>
</table>

WHAT ARE THE DEVELOPMENT PATHS FOR DIFFERENT FILTERS?

UKF	EKF	UDKF
Spacecraft Simulator		
Filter Wrapper (Attitude Propagation)		
Propagation & Observation Functions	Propagation Jacobian & Effective Process Noise	
UKF Implementation	EKF Implementation	UDKF Implementation
Unit Testing	Unit Testing	Unit Testing
Integration & Testing in Simulation		

SUNK COSTS

WHAT ARE THE DEVELOPMENT PATHS FOR DIFFERENT FILTERS?

UKF	EKF	UDKF
Spacecraft Simulator		
Filter Wrapper (Attitude Propagation)		
Propagation & Observation Functions	Propagation Jacobian & Effective Process Noise	
UKF Implementation	EKF Implementation	UDKF Implementation
Unit Testing	Unit Testing	Unit Testing
Integration & Testing in Simulation		

SUNK COSTS

AUTOMATED

WHAT ARE THE DEVELOPMENT PATHS FOR DIFFERENT FILTERS?

UKF	EKF	UDKF
Spacecraft Simulator		
Filter Wrapper (Attitude Propagation)		
Propagation & Observation Functions	Propagation Jacobian & Effective Process Noise	
UKF Implementation	EKF Implementation	UDKF Implementation
Unit Testing	Unit Testing	Unit Testing
Integration & Testing in Simulation		

SUNK COSTS

MANUAL

AUTOMATED

POTENTIAL FILTER GENERATORS

SELECTED

*kf	AUTOFILTER
Generates custom code; has run in embedded environment.	
Filter architecture is emergent, not specified.	Uses template for architecture.
Pieces together best "snippets" that fit user's "assumptions".	Fills in architecture with best components for user's functions.
Integrates user's functions as black boxes.	Manipulates fully symbolic user functions.
Currently supported; has flight heritage	Not funded; Dr. Johann Schumann may be able to provide code.

J. Whittle & J. Schumann, "Automating the Implementation of Kalman Filter Algorithms"

UKF

FILTER WRAPPER: PROPAGATE, RUN FILTER, & CORRECT

- Subtract estimated bias from gyro measurement.
- Propagate the attitude.
- Calculate measurement residual (innovation vector).
- Run the UKF/EKF/UDKF filter.
- Correct the propagated attitude and bias.

TWO FUNCTIONS FOR THE UKF

$$f \qquad\qquad h$$

Follows Crassidis & Markley, "Unscented Filtering for Spacecraft Attitude Estimation".

TWO FUNCTIONS FOR THE UKF

- Propagation function $$\delta x_{i,k} = f(\delta x_{i,k\text{-}1}, \nu_{i,k\text{-}1})$$
  - Given hypothetical attitude error, bias, and gyro noise for last sample, determine current attitude error and bias (~six lines).
- Observation function $$\delta z_{i,k} = h(\delta x_{i,k})$$
  - Given hypothetical attitude error and bias, determine current measurement error (one line).

Follows Crassidis & Markley, "Unscented Filtering for Spacecraft Attitude Estimation"

UKF IMPLEMENTATION

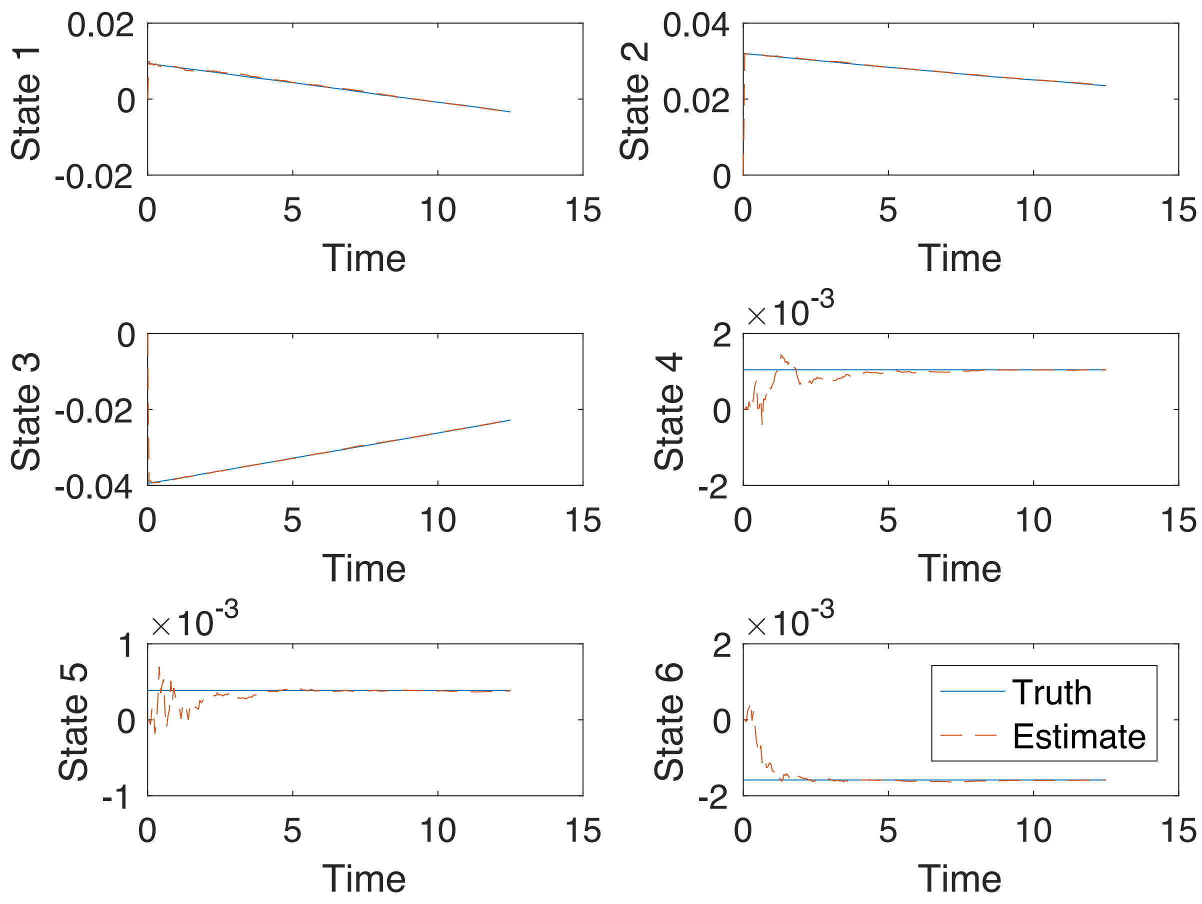
- Sigma point propagation function name: f
- Sigma point observation function name: h
- Process noise covariance: Q (constant in workspace)
- Measurement noise covariance: R (constant in workspace)
- Measurement noise: additive (simplifies calculations)
- Specify when a new measurement is available.
- Output innovation covariance (for analysis).

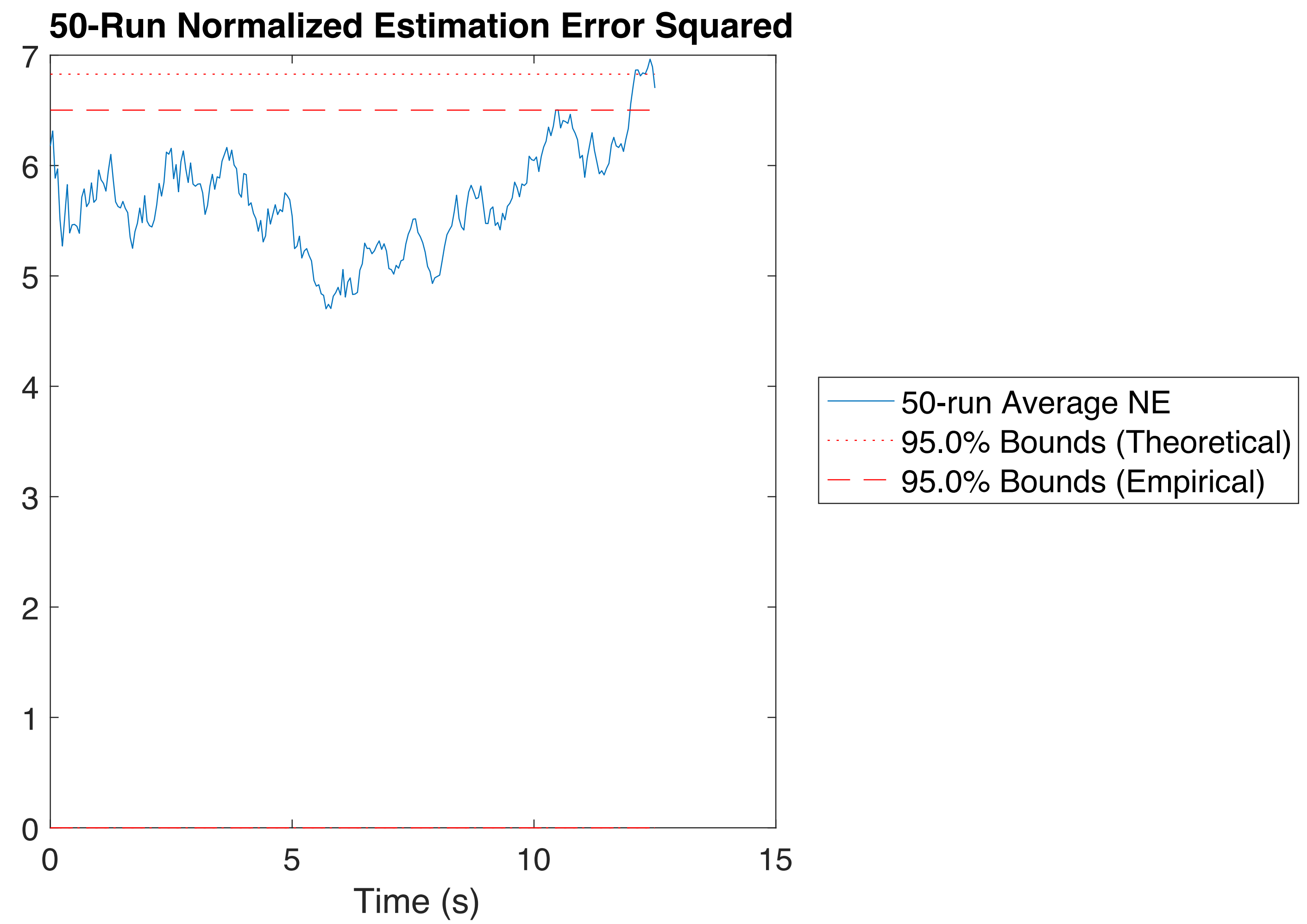
GENERATED FILES

- Initialization function (sets parameters, constants)
- Filter function (performs one step of the filter algorithm)
- Example simulation (used to unit-test filter)
- Example Monte-Carlo wrapper (used to unit-test filter consistency)

UKF UNIT TESTING

- Does filter appear to work?
- Is the covariance matrix consistent with real errors?

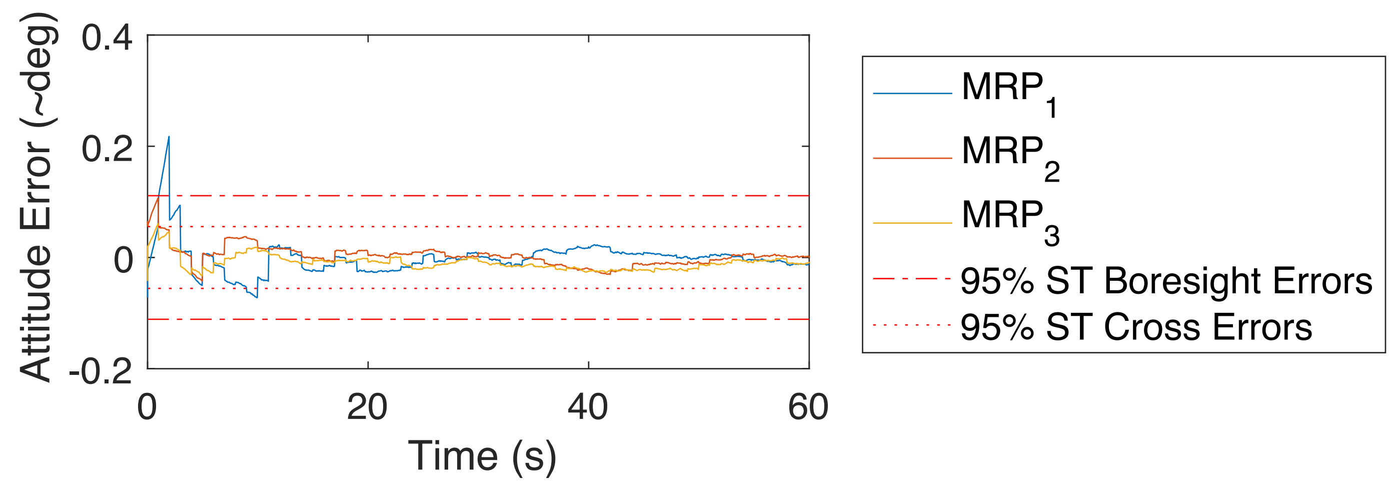


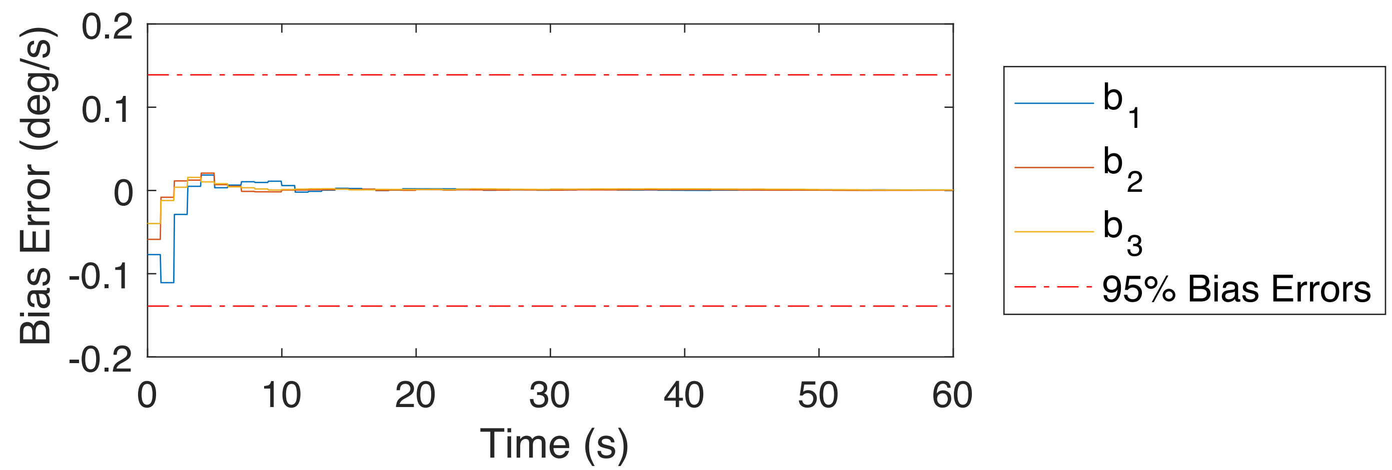


UKF SIMULATION RESULTS

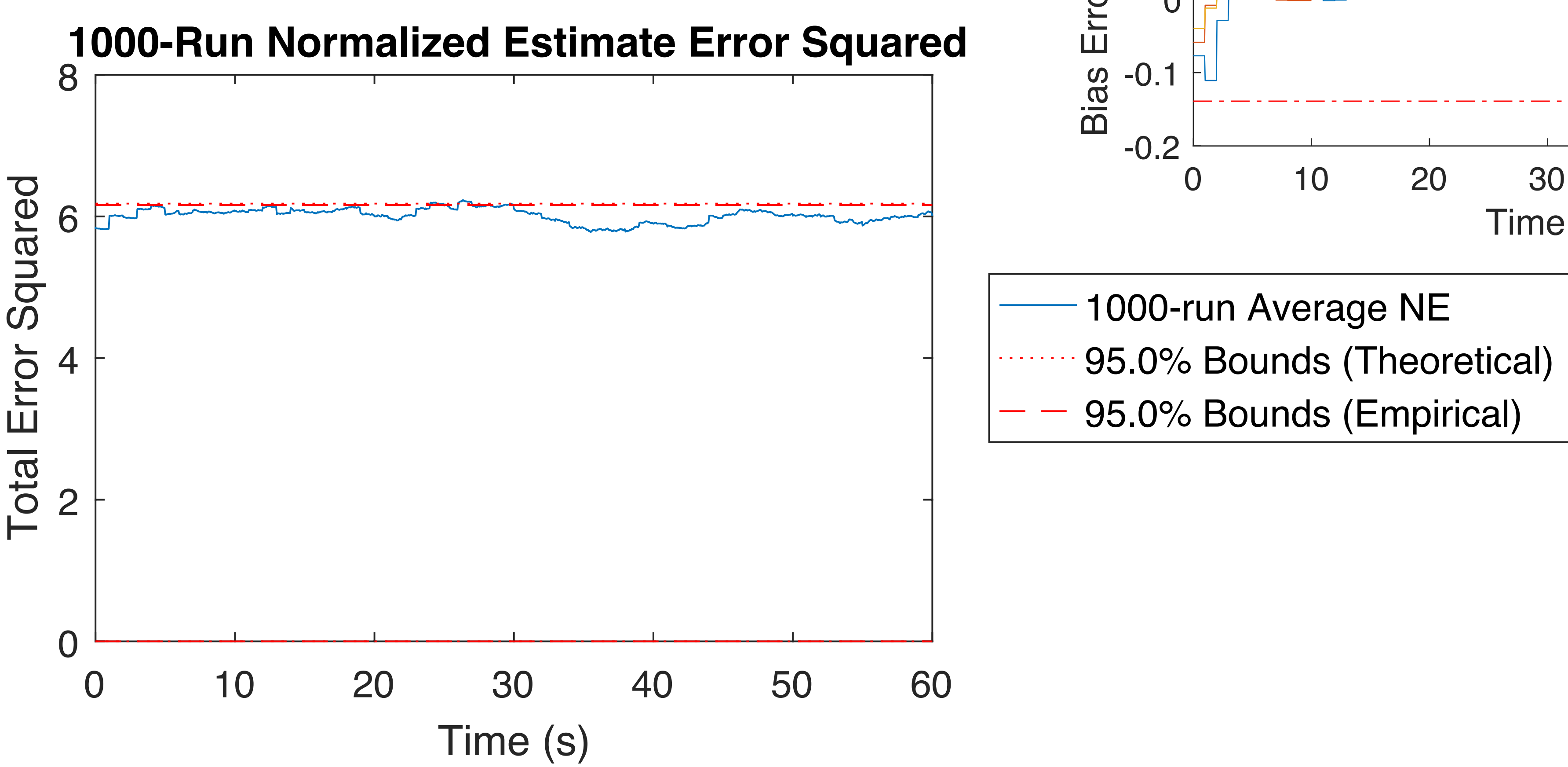
- Results are as expected.
- Filter is consistent.

Single Run





MC Results



EKF

EKF VS. UKF

- Embedded performance
  - EKFs are much faster, especially when using sequential scalar updates.
  - EKFs require less RAM.

TWO MATRICES FOR THE EKF

$$F \qquad\qquad Q_{\text{eff}}$$

Follows Lefferts, Markley, & Shuster, "Kalman Filtering for Spacecraft Attitude Estimation".

TWO MATRICES FOR THE EKF

- Propagation Jacobian Function $\delta x_k \cong F\,\delta x_{k\text{-}1}$
  - Produces Jacobian matrix for given state.
  - Easy for this example problem; more difficult for bigger states.
- Effective process noise $Q_{\mathrm{eff}} = F_q\,Q\,F_q{}^{\mathrm{T}}$
  - Based on gyro's angular random walk and bias random walk.

Follows Lefferts, Markley, & Shuster, "Kalman Filtering for Spacecraft Attitude Estimation"

QUICK VERIFICATION OF JACOBIAN AND PROCESS NOISE

- Can use finite-difference method with the UKF's propagation function to spot check Jacobian and effective process noise covariance matrix – a nice advantage to starting with the UKF.

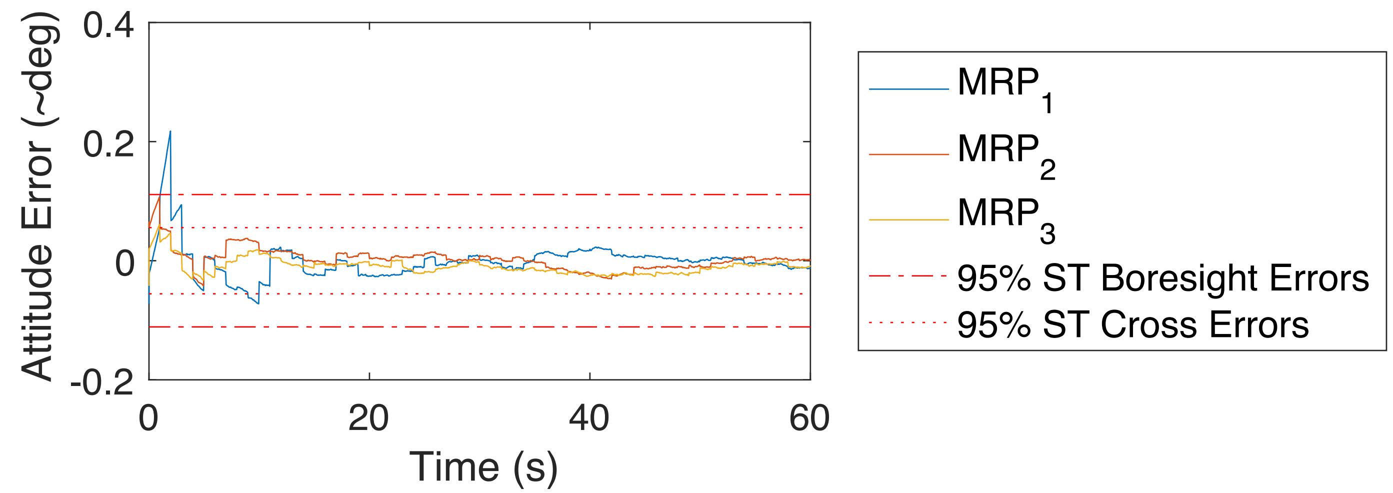
EKF IMPLEMENTATION

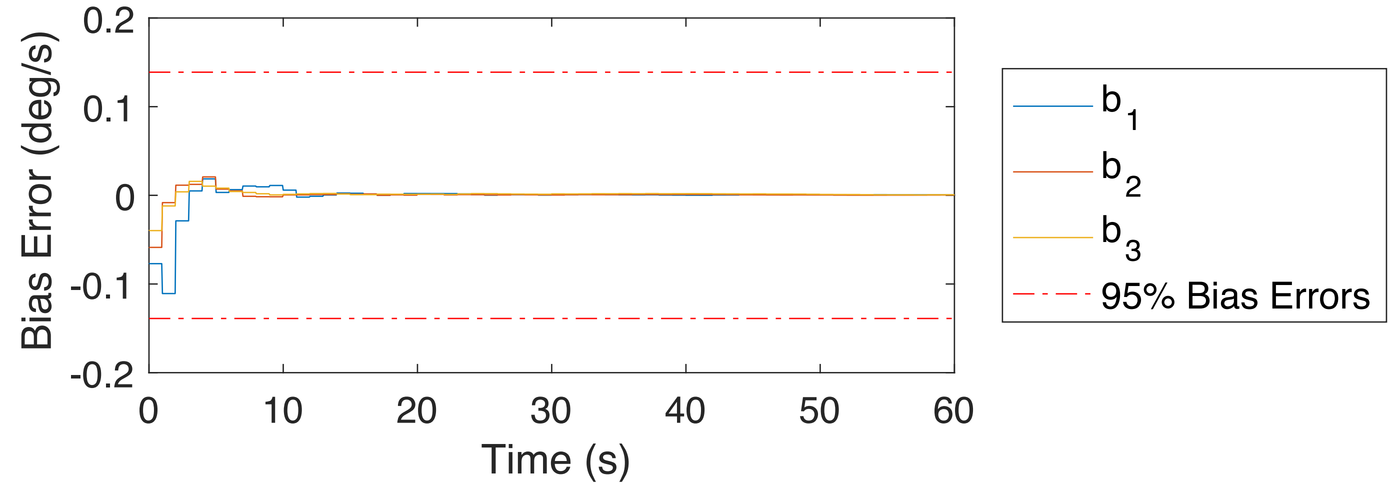
- Propagation function: none (filter wrapper does this)
- Propagation Jacobian function: F (our custom function)
- Process noise covariance: Q_{eff} (constant in workspace)
- Observation function: first 3 indices of error state (simplifies calculation)
- Measurement noise covariance: R (constant in workspace)
- Correction method: sequential scalar updates
- Specify when a new measurement is available.
- Output innovation covariance (for analysis)

EKF RESULTS

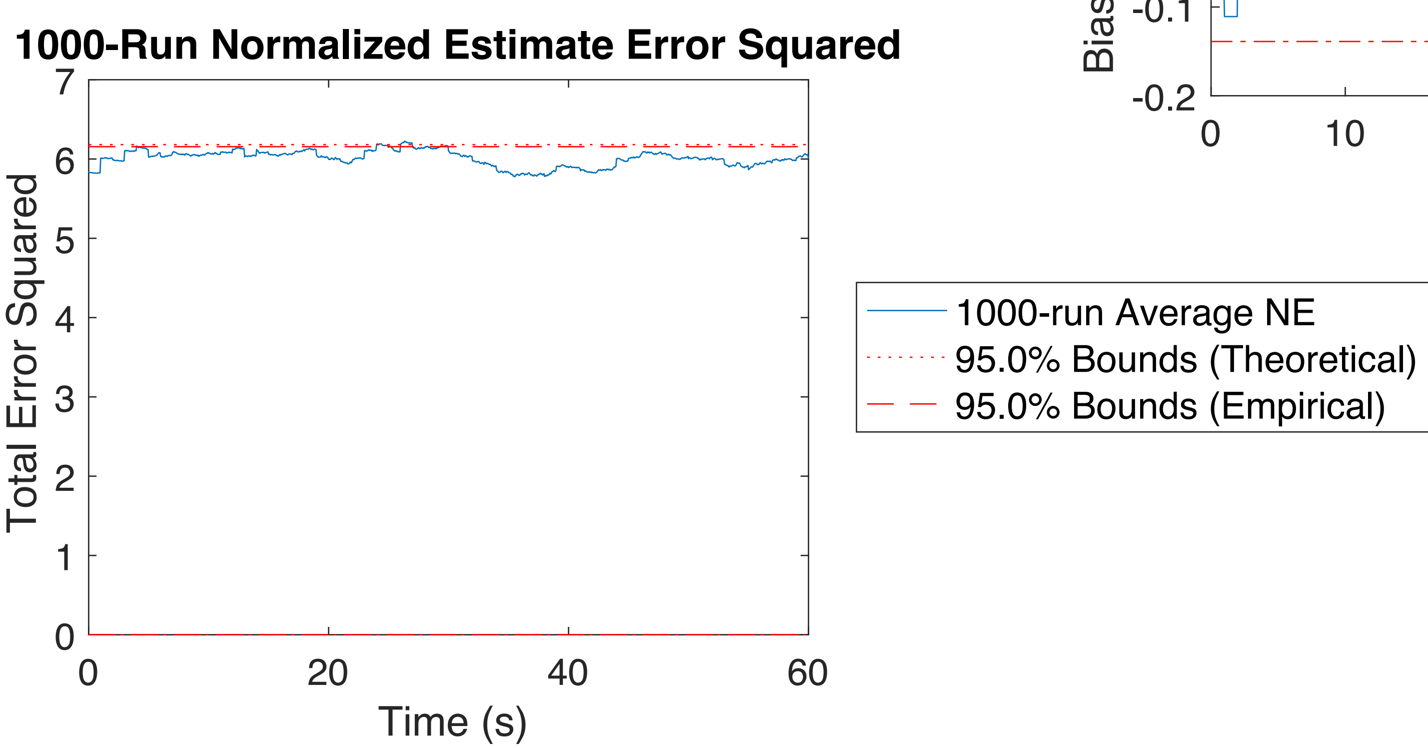
- Virtually identical to UKF.

Single Run

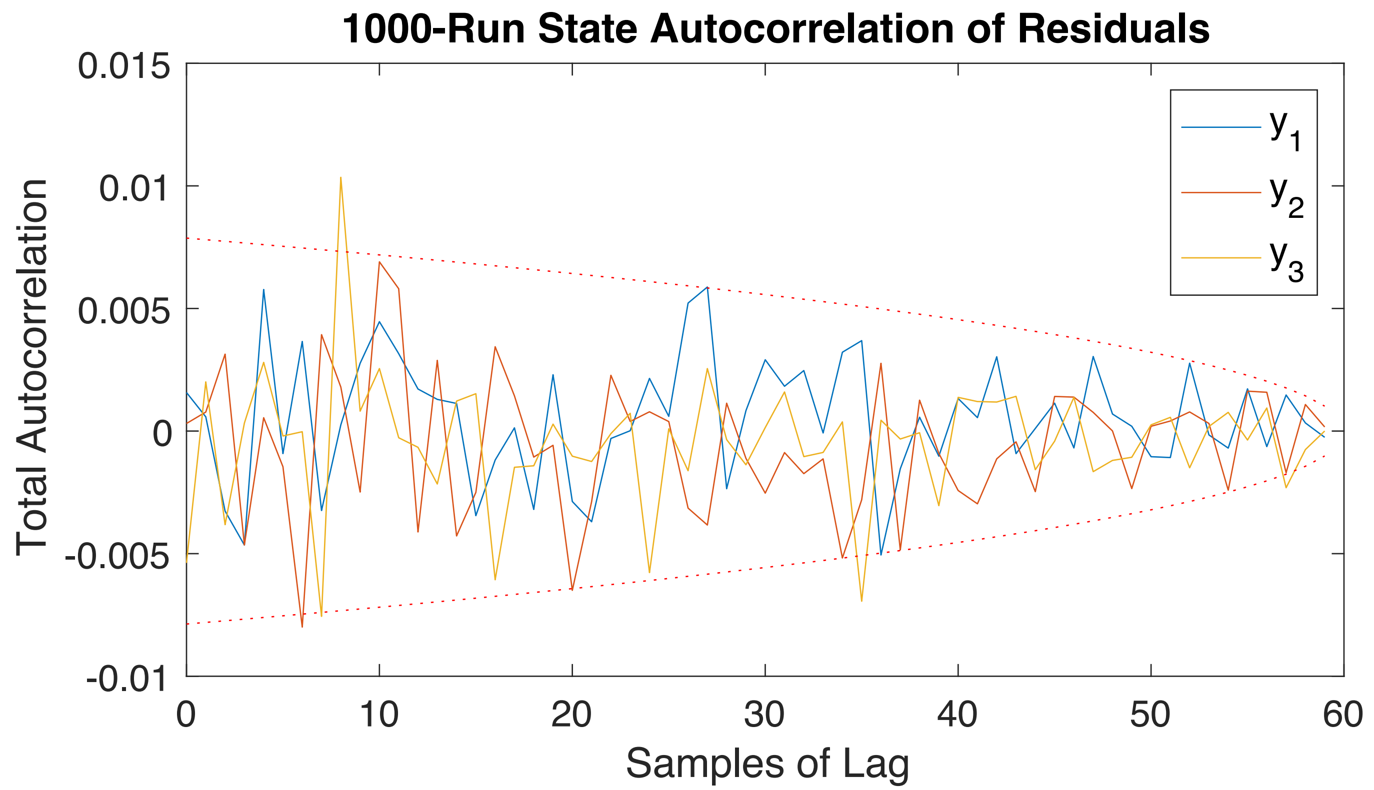


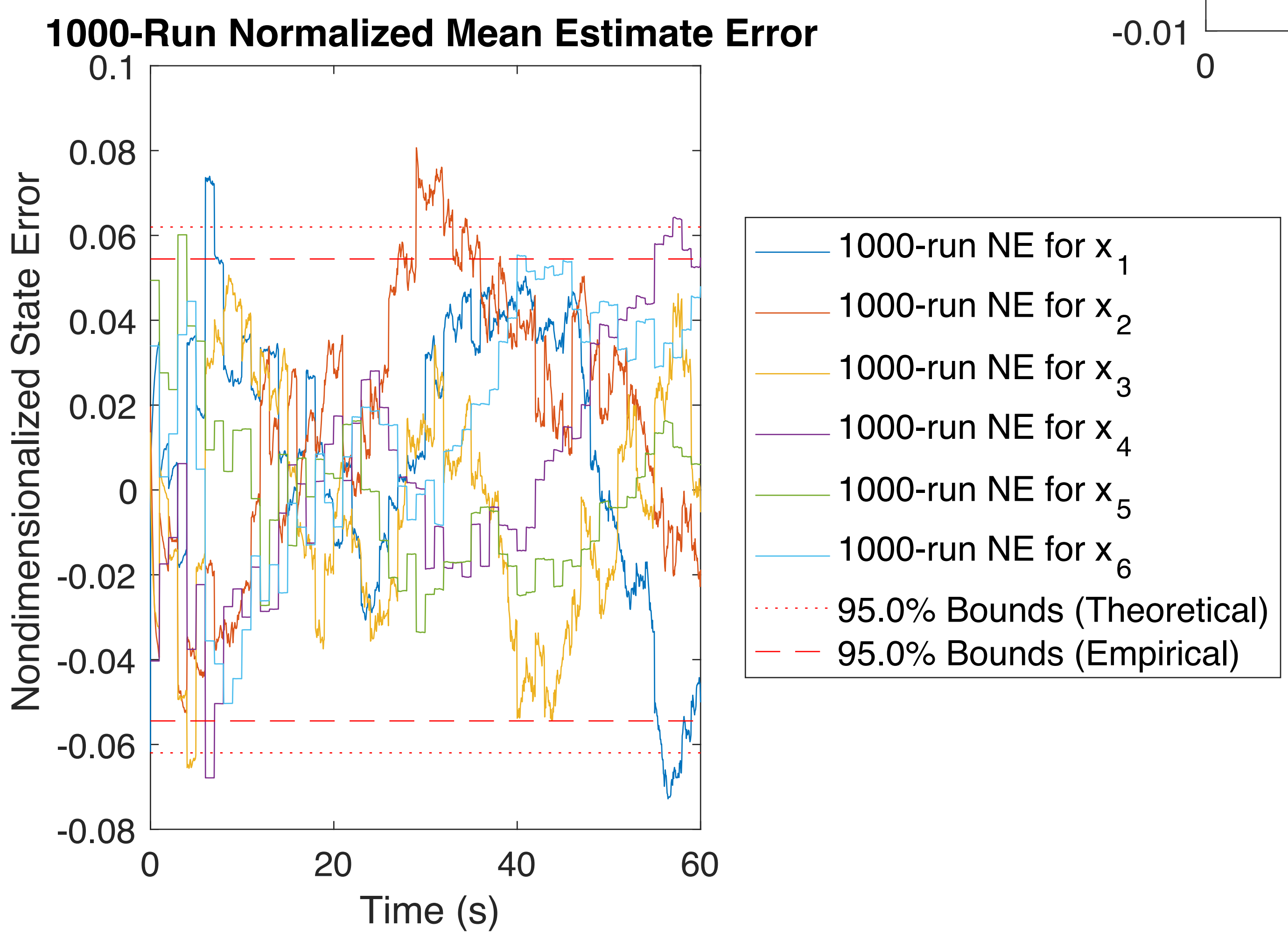


MC Results



EKF RESULTS


1000-Run State Autocorrelation of Residuals
Total Autocorrelation
Samples of Lag
0.015
0.01
0.005
0
-0.005
-0.01
0
10
20
30
40
50
60
y1
y2
y3


1000-Run Normalized Mean Estimate Error
Nondimensionalized State Error
Time (s)
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
0
20
40
60
1000-run NE for x1
1000-run NE for x2
1000-run NE for x3
1000-run NE for x4
1000-run NE for x5
1000-run NE for x6
95.0% Bounds (Theoretical)
95.0% Bounds (Empirical)

UDKF

UDKF VS. EKF

- Operates directly on UD factors of covariance matrix
- Better stability of underlying covariance
- Little additional run-time cost
- Much longer to code by hand

NOTHING ELSE NEEDED FOR UDKF

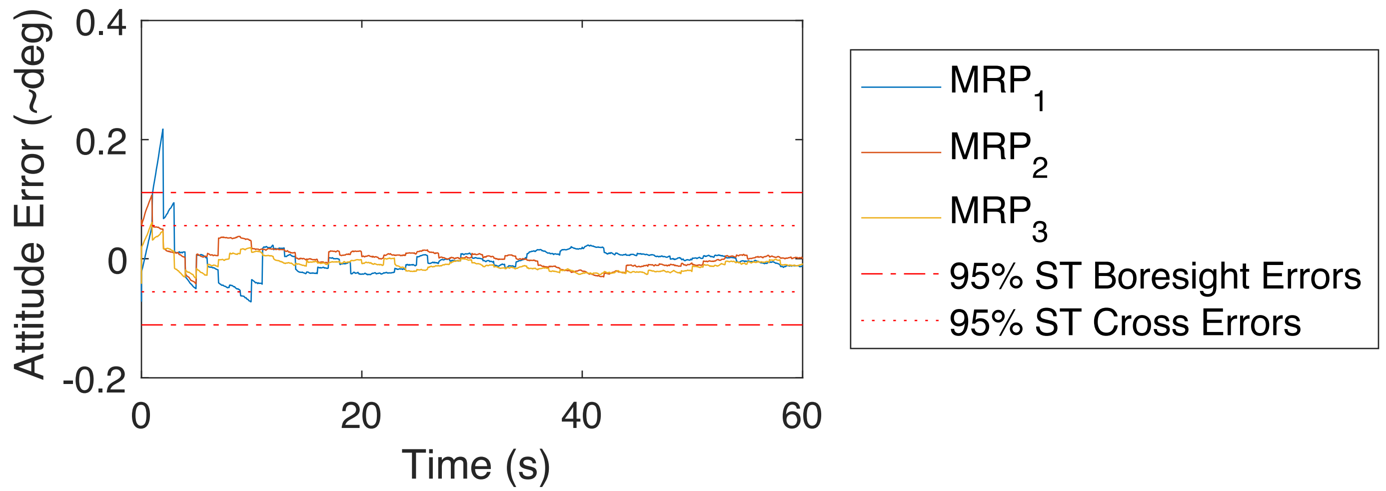
- Just change an option in *kf from "Covariance" to "UDU".

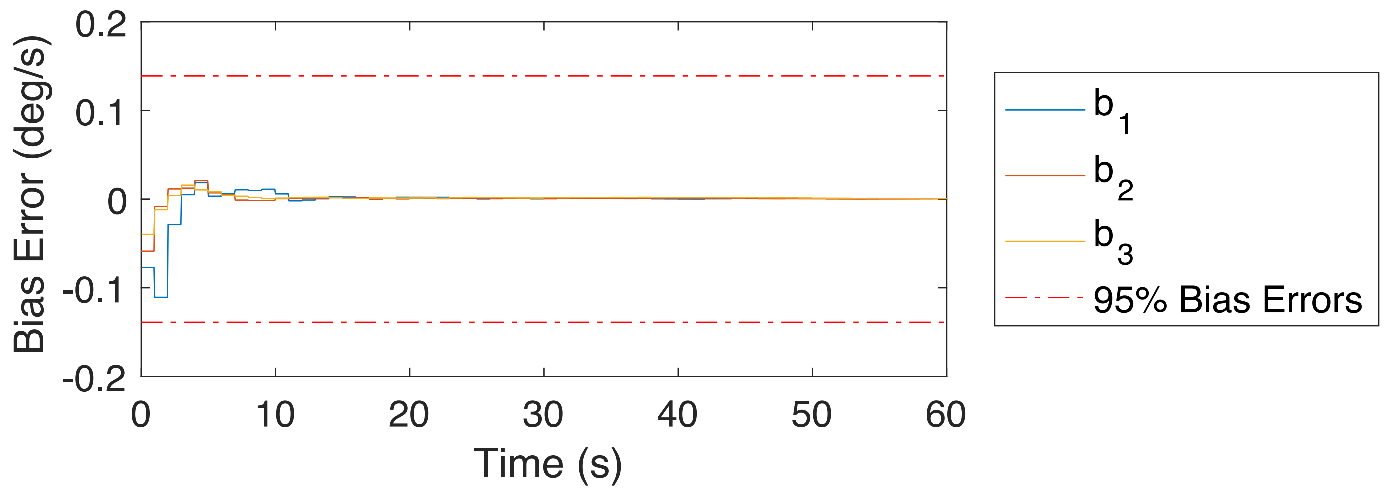
Follows Bierman, *Factorization Methods for Discrete Sequential Estimation*

UDKF RESULTS

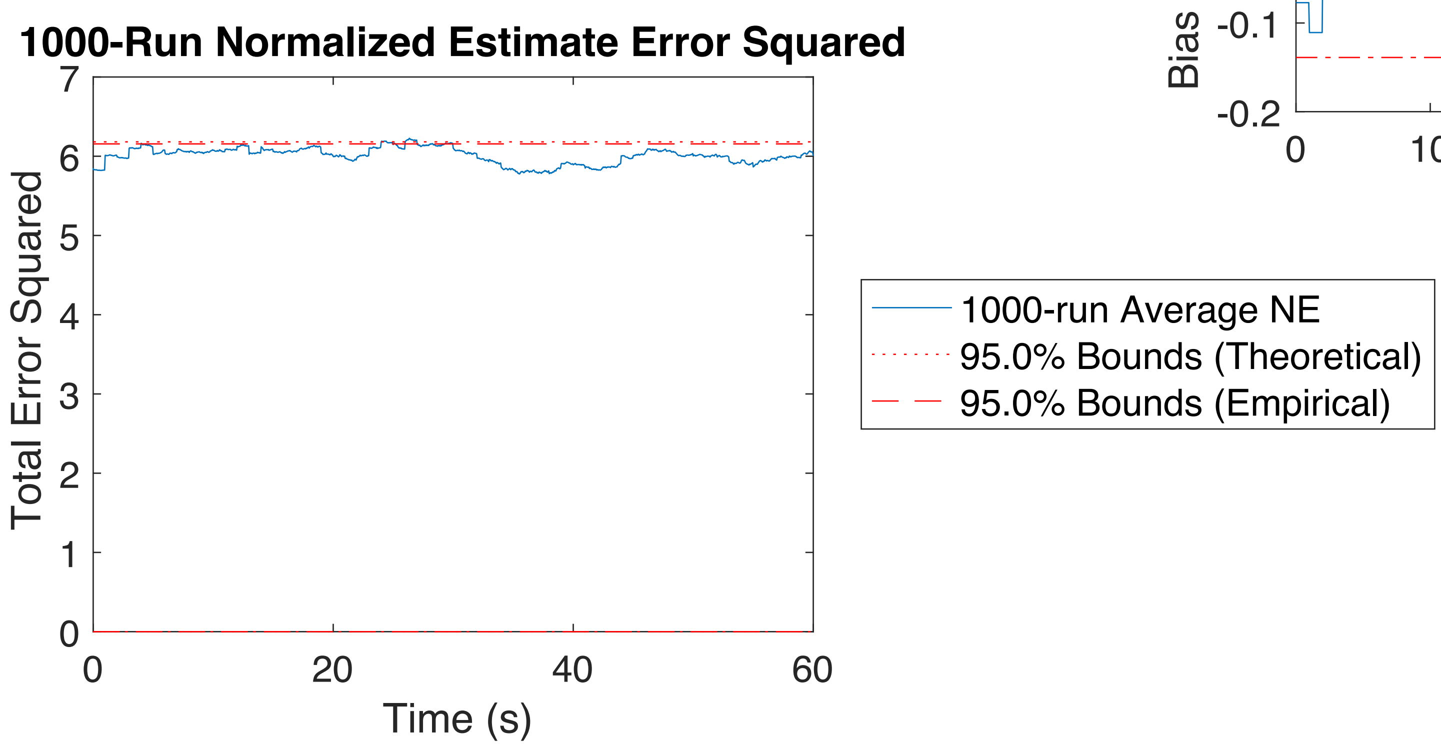
- Identical to EKF, as expected.

Single Run





MC Results



SUMMARY

- Write sim and filter wrapper (necessary anyway)
- Two functions → UKF (sensor trade studies, control development)
- One function and one matrix → EKF (checked against UKF, runs on flight computer!)
- A changed option → UDKF (checked against EKF, more stability with no additional development time)
- Result: estimator available early, little wasted work, mature end product