Non-Centralized Kalman Filters for Position, Navigation, and Timing

January 31, 2020

Jeffrey Won, Andrew J. Binder, Jaime Y. Cruz
Navigation and Geopositioning Systems Department
Systems Analysis and Simulation Subdivision

Prepared for:

Space and Missile Systems Center
Air Force Space Command
483 N. Aviation Blvd.
El Segundo, CA 90245-2808

Contract No. FA8802-19-C-0001

Authorized by: Corporate Chief Engineer's Office

Abstract

Kalman filters are a fundamental state estimation tool, but they traditionally require access to the full state, covariance, and measurement information for the system being estimated. For large, widely distributed systems, this condition can make communication requirements prohibitively complex and expensive. This study explores an alternative class of Kalman filter which uses a collection of local filters to estimate the global state, allowing the estimation process to be scalable with the size of the system. Some promising algorithms were identified for crosslink-based satellite navigation in proliferated constellations. A method was developed to formally evaluate the performance of various distributed algorithms, and simulation results are presented for the most promising algorithm.

Non-Centralized Kalman Filters for Position, Navigation, and Timing

Andrew J. Binder, Jeffrey Won, Jaime Y. Cruz
Navigation & Geopositioning Systems Dept.

January 31, 2020

Outline

- Introduction
- Distributed filter algorithm descriptions
  - *Primitive distributed Kalman filter (PDKF)*
  - *Decentralized collaborative localization (DCL)*
  - *Distributed covariance intersection filter (DCIF)*
- Formal evaluation of filter performance
- Simulation results
- Conclusion

Introduction

Background

- Kalman filters are a fundamental tool for the state estimation of dynamical systems
  - *State and uncertainty are propagated using the dynamic model of the system*
  - *Measurement information is absorbed to correct the predicted state and covariance*
  - *Dynamics model error can be accounted for through injection of process noise*
  - *For certain formulations, the estimate produced can be shown to be optimal (in the sense of minimizing the mean square error)*
- Classical Kalman filters imply an estimator with access to the full-system state, covariance, and measurement information
  - *From here on, this will be referred to as a **centralized Kalman filter** (CKF)*
- Another class of Kalman filters produces a global state estimate using a collection of local Kalman filters with some means of communicating state or measurement information
  - *Decentralized, distributed, or federated Kalman filters*
  - *Consensus-based Kalman filters*
  - *Collaborative localization (in robotics)*
  - *Parallel Information filters*

Motivation

- Not always simple or desirable to collect all pertinent information into a centralized location
  - *Complexity of communication of estimator information*
  - *Large system size and numerical difficulties*
- Examples
  - *Large sensor networks widely distributed in time and space*
  - *Networks of semi-autonomous systems (e.g., self-driving cars or robots)*
  - *Proliferated low Earth-orbit constellations*
- What are good choices for an estimation architecture that balance optimality of the estimator with processing and communication requirements on a large networked system?
  - *Survey and compare Kalman filtering techniques*

Overview of Non-Centralized Kalman Filters

- Decentralized Kalman filter
  - *State estimation using a set of local Kalman filters that communicate with all nodes*
  - *Approximates optimal estimate for entire system*
- Distributed Kalman filter (sometimes referred to as scalable Kalman filter)
  - *State estimation using a set of local Kalman filters that communicate with a strict subset of the nodes in the system*
  - *Approximates optimal estimate for entire system*
- Consensus-based Kalman filters
  - *May be used in conjunction with other non-centralized filters*
  - *Often does not approximate optimal estimate for entire system*
  - *Uses average consensus of state or measurement information to produce uniform global estimate at nodes*
    - Simplifies implementation by not attempting to de-correlate common information communicated over the network when averaging
- Federated Kalman filters
  - *A centralized filter manages a collection of local filters*
  - *Information is exchanged in both directions*

Necessary Decorrelation of Information

- Distributed Kalman filters and consensus-based Kalman filters may pass around either state or measurement information
- If state information is communicated, the dependency of information must be considered
  - *There are multiple paths that information from a single measurement may travel to reach a given node*
    - "Double-counting" of information
  - *Requires complicated information graphs to account for paths information may travel to add and remove information correctly*
    - Consensus-based filters do not attempt to capture this effect when they perform their averaging
- Measurement information sharing does not require the same level of tracking

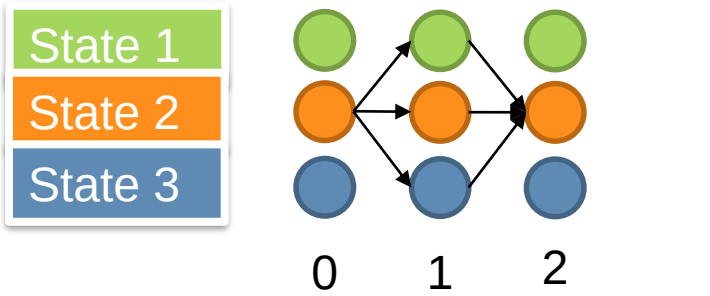


Figure: Information about state 2 at initialization is communicated to adjacent states at iteration 1 and then back to state 2 at iteration 2. The initialization information for state 2 then is weighted more heavily as a result

Distributed Kalman Filters

- Distributed Kalman filters are quite applicable for performing orbit determination and clock synchronization for a large, low Earth-orbit constellation
  - *Numerous implementations exist*
- Filter used in initial study [1] is most primitive distributed Kalman filter [5]
  - *Consider natural extensions of this algorithm*
    - Reduces disagreement that may grow between estimates of states and improves robustness of algorithm
    - Minor increase in communication requirements
    - Remains quite localized
- Distributed Kalman filters that use measurement fusion have small communication requirements and do not require the tracking of an information graph [2]
  - *Variable increase in communication requirements*
  - *Less localized but easy to show that it approximates optimal estimate*
- Distributed Kalman filters that only consider influences from neighbors ℓ nodes away [4]

Sources of Non-Optimality

- Non-centralized filters in general produce non-optimal state and covariance estimates
- Two sources of non-optimality:
  - *Non-optimal gain matrix selection due to limitations on which measurements and covariances are "available" (i.e., able to be communicated) to a given node*
  - *Non-optimal covariance update with a given gain matrix due to limitations on how much of the global covariance matrix is available to a given node*
- All non-centralized filter algorithms make some form of approximation relative to a centralized Kalman filter (CKF)
  - *Selection of an appropriate algorithm for a given application should involve evaluating the validity of each approximation for that application*

Filters Analyzed

- Centralized Kalman Filter (CKF)
  - *Standard, fully-correlated Kalman Filter*
  - *All measurements are processed at a central computing node*
- Primitive Distributed Kalman Filter (PDKF)
  - *Simplest possible distributed filter*
  - *No cross-correlations are tracked between nodes*
- Decentralized Collaborative Localization (DCL)
  - *Distributed algorithm developed for robotic systems*
  - *Tracks approximate cross-correlations between nodes*
  - *Assumes only two nodes communicate at a given time*
- Distributed Covariance Intersection Filter (DCIF)
  - *Developed internally as a natural extension of existing literature*
  - *Does not track cross-correlations between nodes, but guarantees a conservative covariance estimate when cross-correlations between nodes are unknown*

Distributed Algorithm: Primitive Distributed Kalman Filter (PDKF)

PDKF Algorithm

- The covariance matrix for the fully correlated system is given by

$$\boldsymbol{P} = \begin{bmatrix} P_{11} & \cdots & P_{1i} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ P_{i1} & \cdots & P_{ii} & \cdots & P_{in} \\ \vdots & & \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{ni} & \cdots & P_{nn} \end{bmatrix}$$

where $i = 1, \ldots n$ are the nodes of the system and each submatrix P_{ii} is $m \times m$, where m is the number of states being estimated at each node

- Each node i stores only its local covariance matrix P_{ii} and its state estimate x_i
- No cross covariances $P_{ij},\ \{j \neq i\}$ are stored
- Each node is responsible for propagating and updating its own state and covariance
- When a measurement is made between nodes, each node sends its local state and covariance estimates along with the measurement

PDKF Algorithm

- When node i absorbs measurements from a set of nodes N_j, the extended local covariance matrix is formed as:

$$P_{ext}^{-} = \begin{bmatrix} P_{ii}^{-} & 0 & 0 & 0 \\ 0 & P_{N_1}^{-} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & P_{N_n}^{-} \end{bmatrix}$$

- The local covariance update for node i is then performed as:

$$S = HP_{ext}^{-}H^T + R$$
$$K = P_{ext}^{-}H^T S^{-1}$$
$$P_{ext}^{+} = (I - KH)P_{ext}^{-}(I - KH)^T + KRK^T$$

- Only the portion of P_{ext}^{+} corresponding to P_{ii} is stored by node i, while the rest of P_{ext}^{+} is discarded

PDKF Algorithm Summary

- Each node stores: x_i, P_{ii}
- Propagate step: each node performs

$$x_{i,t+1} = \phi_{i,t} x_{i,t}$$
$$P_{ii,t+1} = \phi_{i,t} P_{ii,t} \phi_{i,t}^T + Q$$

- Relative measurement update step: each node sends: x_i, P_{ii}, z_{obs}
- Each node i performs:

form P_{ext}^-

$$S = H P_{ext}^- H^T + R$$
$$K = P_{ext}^- H^T S^{-1}$$
$$x_i^+ = x_i^- + K_i (z_{obs} - z_{model})$$
$$P_{ext}^+ = (I - KH) P_{ext}^- (I - KH)^T + KRK^T$$

set P_{ii}^+ to the upper-left submatrix of P_{ext}^+

Non-Optimality of the PDKF

- The PDKF is oblivious to cross-correlations between all nodes, and so makes significant approximations in both the gain selection and covariance update
- Ignoring cross-correlations will tend to over-estimate the amount of information contained in each measurement, causing the filter to generally under-estimate its own covariance
  - *In cases where the cross-correlations between nodes are significant, this can lead to divergent behavior in the filter*

Distributed Algorithm: Decentralized Collaborative Localization (DCL)

Decentralized Collaborative Localization[3] (DCL)

- This is a distributed algorithm derived by approximating the fully-correlated, centralized Extended Kalman Filter (EKF)
- The cross covariances $P_{i,j}$ between nodes i and j are decomposed into $P_{i,j} = \sigma_{i,j}\sigma_{j,i}^{T}$, so that each node can carry and propagate a portion of the cross covariance
  - *Node i carries $\sigma_{i,j}$ and node j carries $\sigma_{j,i}^{T}$, which can be recombined to form $P_{i,j}$ when the nodes communicate again*
- Each node i carries its state x_i, covariance $P_{i,i}$, and cross covariance portions $\sigma_{i,j}$ for each other node j that it has communicated with at some point
- x_i, $P_{i,i}$, and $\sigma_{i,j}$ can be propagated in time rigorously without contact to any other node
- Measurement updates are approximated by not updating the state and auto-covariance of non-participating nodes, and performing an approximate update of the cross covariances between one participating node and one non-participating node
  - *The measurement update of the state, covariance, and cross covariance of the participating nodes is equal to that of the centralized KF*

DCL Algorithm

- The covariance matrix for the fully correlated system is given by

$$\boldsymbol{P} = \begin{bmatrix} P_{11} & \cdots & P_{1i} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ P_{i1} & \cdots & P_{ii} & \cdots & P_{in} \\ \vdots & & \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{ni} & \cdots & P_{nn} \end{bmatrix}$$

where $i = 1, \ldots n$ are the nodes of the system and each submatrix P_{ii} is $m \times m$, where m is the number of states being estimated at each node

- The cross covariance submatrices P_{ij} can be expressed as $P_{ij} = \sigma_{ij}\sigma_{ji}^{T}$
- During the propagation step, the cross covariances are propagated as

$$P_{ij,t+1}^{-} = \phi_i P_{ij,t}^{+} \phi_j^{T}$$

which can be expressed as

$$\left(\phi_i \sigma_{ij,t}\right)\left(\phi_j \sigma_{ji,t}\right)^{T} \quad (1)$$

- If σ_{ij} is stored by node i, and σ_{ji} is stored by node j, then the cross covariance can be propagated without communication of the state transition matrices ϕ_i, ϕ_j between the two nodes

DCL Algorithm

- At absorption of measurement between nodes i and j, the covariance and cross covariance of nodes i and j are updated as

$$P_{ii}^{+} = (I - K_i H_i) P_{ii}^{-} - K_i H_j P_{ji}^{-} \quad (1)$$

$$P_{jj}^{+} = \left(I - K_j H_j\right) P_{jj}^{-} - K_j H_i P_{ij}^{-} \quad (2)$$

$$P_{ij}^{+} = (I - K_i H_i) P_{ij}^{-} - K_i H_j P_{jj}^{-} \quad (3)$$

- These equations are exactly the same as in a fully-correlated EKF using the optimal Kalman gain
- The covariances of other nodes not involved in the relative measurement would be propagated in the fully-correlated case as

$$P_{kk}^{+} = P_{kk}^{-} - K_k \begin{bmatrix} H_i & H_j \end{bmatrix} \begin{bmatrix} P_{ik} \\ P_{jk} \end{bmatrix}$$

$$P_{ik/jk}^{+} = \left(I - K_{i/j} H_{i/j}\right) P_{ik/jk}^{-} - K_{i/j} H_{j/i} P_{jk/ik}^{-}$$

- The DCL approximates these updates as follows to eliminate the need for communication with the non-participating nodes:

$$P_{kk}^{+} \approx P_{kk}^{-} \quad (5)$$

$$\sigma_{ik/jk}^{+} \approx P_{ii/jj}^{+} \left(P_{ii/jj}^{-}\right)^{-1} \sigma_{ik/jk}^{-} \quad (6)$$

DCL Algorithm

- The cross-covariance update between a participating node (i, j) and non-participating node k is derived as follows:
- The measurement update for nodes i, j, k can be described in block form as:

$$\begin{bmatrix} P_{ii}^{+} & P_{ij}^{+} & P_{ik}^{+} \\ P_{ji}^{+} & P_{jj}^{+} & P_{jk}^{+} \\ P_{ki}^{+} & P_{kj}^{+} & P_{kk}^{+} \end{bmatrix} = \begin{bmatrix} I - K_i H_i & -K_i H_j & -K_i H_k \\ -K_j H_i & I - K_j H_j & -K_j H_k \\ -K_k H_i & -K_k H_j & I - K_k H_k \end{bmatrix} \begin{bmatrix} P_{ii}^{-} & P_{ij}^{-} & P_{ik}^{-} \\ P_{ji}^{-} & P_{jj}^{-} & P_{jk}^{-} \\ P_{ki}^{-} & P_{kj}^{-} & P_{kk}^{-} \end{bmatrix}$$

- For a relative measurement update between nodes i and j, the submatrices K_k and H_k are 0-filled by the definition of the DCL algorithm
- Using this fact and separating into blocks:

$$\begin{bmatrix} P_{ii}^{+} & P_{ij}^{+} \\ P_{ji}^{+} & P_{jj}^{+} \end{bmatrix} = \begin{bmatrix} I - K_i H_i & -K_i H_j \\ -K_j H_i & I - K_j H_j \end{bmatrix} \begin{bmatrix} P_{ii}^{-} & P_{ij}^{-} \\ P_{ji}^{-} & P_{jj}^{-} \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} P_{ik}^{+} \\ P_{jk}^{+} \end{bmatrix} = \begin{bmatrix} I - K_i H_i & -K_i H_j \\ -K_j H_i & I - K_j H_j \end{bmatrix} \begin{bmatrix} P_{ik}^{-} \\ P_{jk}^{-} \end{bmatrix} = A \begin{bmatrix} P_{ik}^{-} \\ P_{jk}^{-} \end{bmatrix} \tag{8}$$

where

$$A = \begin{bmatrix} I - K_i H_i & -K_i H_j \\ -K_j H_i & I - K_j H_j \end{bmatrix} = \begin{bmatrix} P_{ii}^{+} & P_{ij}^{+} \\ P_{ji}^{+} & P_{jj}^{+} \end{bmatrix} \begin{bmatrix} P_{ii}^{-} & P_{ij}^{-} \\ P_{ji}^{-} & P_{jj}^{-} \end{bmatrix}^{-1} \tag{9}$$

Non-Participating Cross-Covariance Update

- The DCL algorithm assumes that during the relative measurement update between nodes i and j, there is no access to σ_{ki} and σ_{kj} stored at node k, and so the full cross-covariances P_{ik}, P_{jk} cannot be formed
- Two approximations for equation (8) are discussed: a "naive" approximation, and the approximation given in (6)
  - *"Naive" approximation (simply take the block diagonal of A):*

$$A = \begin{bmatrix} I - K_i H_i & 0 \\ 0 & I - K_j H_j \end{bmatrix}$$

$$P_{ik}^{+} = (I - K_i H_i) P_{ik}^{-} \tag{10}$$

$$P_{jk}^{+} = \left(I - K_j H_j\right) P_{jk}^{-} \tag{11}$$

  - *Approximation from (6)*

$$A = \begin{bmatrix} P_{ii}^{+}(P_{ii}^{-})^{-1} & 0 \\ 0 & P_{jj}^{+}\left(P_{jj}^{-}\right)^{-1} \end{bmatrix} = \begin{bmatrix} I - K_i H_i - K_i H_j P_{ji}^{-}(P_{ii}^{-})^{-1} & 0 \\ 0 & I - K_j H_j - K_j H_i P_{ij}^{-}\left(P_{jj}^{-}\right)^{-1} \end{bmatrix}$$

$$P_{ik}^{+} = (I - K_i H_i) P_{ik}^{-} - K_i H_j P_{ji}^{-}(P_{ii}^{-})^{-1} P_{ik}^{-} \tag{12}$$

$$P_{jk}^{+} = \left(I - K_j H_j\right) P_{jk}^{-} - K_j H_i P_{ij}^{-}\left(P_{jj}^{-}\right)^{-1} P_{jk}^{-} \tag{13}$$

- The rigorous update (expanding (8) into its components):

$$P_{ik}^{+} = (I - K_i H_i) P_{ik}^{-} - K_i H_j P_{jk}^{-} \tag{14}$$

$$P_{jk}^{+} = \left(I - K_j H_j\right) P_{jk}^{-} - K_j H_i P_{ik}^{-} \tag{15}$$

Non-Participating Cross-Covariance Update

- Comparing (10) and (12) to (14), we can see that the following approximations are being made in the update for P_{ik} (14):
  - *Naive approximation – comparing (10) to (14):*
$$P_{jk}^{-} = 0$$
  - *Approximation (6) – comparing (12) to (14):*
$$P_{jk}^{-} = P_{ji}^{-}(P_{ii}^{-})^{-1}P_{ik}^{-} \quad (16)$$
- Approximation (6) becomes exact if $P_{ii}^{-} = P_{ij}^{-}\left(P_{jj}^{-}\right)^{-1}P_{ji}^{-}$, in other words, if i and j are highly correlated
- Proof: substitute $P_{ii}^{-} = P_{ij}^{-}\left(P_{jj}^{-}\right)^{-1}P_{ji}^{-}$ into (7). Then:

$$\begin{bmatrix} P_{ii}^{+} & P_{ij}^{+} \\ P_{ji}^{+} & P_{jj}^{+} \end{bmatrix} = \begin{bmatrix} 0 & (I - K_iH_i)P_{ij}^{-}\left(P_{jj}^{-}\right)^{-1} - K_iH_j \\ \left(I - K_jH_j\right)P_{ji}^{-}(P_{ii}^{-})^{-1} - K_jH_i & 0 \end{bmatrix} \begin{bmatrix} P_{ii}^{-} & P_{ij}^{-} \\ P_{ji}^{-} & P_{jj}^{-} \end{bmatrix}$$

- Using this new A matrix, (8) becomes

$$\begin{bmatrix} P_{ik}^{+} \\ P_{jk}^{+} \end{bmatrix} = \begin{bmatrix} 0 & (I - K_iH_i)P_{ij}^{-}\left(P_{jj}^{-}\right)^{-1} - K_iH_j \\ \left(I - K_jH_j\right)P_{ji}^{-}(P_{ii}^{-})^{-1} - K_jH_i & 0 \end{bmatrix} \begin{bmatrix} P_{ik}^{-} \\ P_{jk}^{-} \end{bmatrix} \quad (17)$$

Non-Participating Cross-Covariance Update

- Multiplying out (17) yields the following system of equations:

$$P_{ik}^{+} = (I - K_i H_i) P_{ij}^{-} \left(P_{jj}^{-}\right)^{-1} P_{jk}^{-} - K_i H_j P_{jk}^{-} \quad (18)$$

$$P_{jk}^{+} = \left(I - K_j H_j\right) P_{ji}^{-} (P_{ii}^{-})^{-1} P_{ik}^{-} - K_j H_i P_{ik}^{-} \quad (19)$$

- Comparing (18) and (19) to the rigorous forms (14) and (15), we see that

$$P_{ik}^{-} = P_{ij}^{-} \left(P_{jj}^{-}\right)^{-1} P_{jk}^{-}$$

$$P_{jk} = P_{ji}^{-} (P_{ii}^{-})^{-1} P_{ik}^{-}$$

- These match the approximation in (16) exactly, which means that making the approximation in (16) is equivalent to $P_{ii}^{-} = P_{ij}^{-} \left(P_{jj}^{-}\right)^{-1} P_{ji}^{-}$

DCL Algorithm Summary

- Each node i $(i = 1,2, \ldots n)$ stores: x_i, P_{ii}, σ_{ij} $(j = 1,2, \ldots n; j \neq i)$
- Propagate step: each node performs

$$x_{i,t+1} = \phi_{i,t} x_{i,t}$$
$$P_{ii,t+1} = \phi_{i,t} P_{ii,t} \phi_{i,t}^T + Q$$
$$\sigma_{ij,t+1} = \phi_{i,t} \sigma_{ij,t}$$

- Relative measurement update step: each node sends: $x_i, P_{ii}, \sigma_{ij}, z_{obs}$
- Each node pair (i, j) with node-to-node measurements performs:

$$P_{ij}^- = \sigma_{ij} \sigma_{ji}^T$$
$$S = \begin{bmatrix} H_i & H_j \end{bmatrix} \begin{bmatrix} P_{ii}^- & P_{ij}^- \\ P_{ji}^- & P_{jj}^- \end{bmatrix} \begin{bmatrix} H_i^T \\ H_j^T \end{bmatrix} + R$$
$$\begin{bmatrix} K_i \\ K_j \end{bmatrix} = \begin{bmatrix} P_{ii}^- H_i^T + P_{ij}^- H_j^T \\ P_{jj}^- H_j^T + P_{ji}^- H_i^T \end{bmatrix} S^{-1}$$
$$x_i^+ = x_i^- + K_i (z_{obs} - z_{model})$$
$$x_j^+ = x_j^- + K_j (z_{obs} - z_{model})$$
$$P_{ii}^+ = (I - K_i H_i) P_{ii}^- - K_i H_j P_{ji}^-$$
$$P_{jj}^+ = \left(I - K_j H_j\right) P_{jj}^- - K_j H_i P_{ij}^-$$
$$P_{ij}^+ = (I - K_i H_i) P_{ij}^- - K_i H_j P_{jj}^-$$
$$\sigma_{ik}^+ = P_{ii}^+ (P_{ii}^-)^{-1} \sigma_{ik}^- \quad for\ k \neq \{i, j\}$$

Decompose $\sigma_{ij}^+ \sigma_{ji}^{+T} = P_{ij}^+$

Non-Optimality of the DCL

- The two primary approximations of the DCL algorithm are:
  - *The gain submatrix K_k for non-participating nodes in a measurement update is 0*
  - *The measurement update for cross covariances between a participating and non-participating node, P_{ik} and P_{jk}, are approximate rather than exact*
- The gain matrix approximation works best if non-participating nodes in a measurement update are uncorrelated with the participating nodes
  - *If non-participating nodes are uncorrelated with the participating nodes, then K_k would also be 0 in a CKF*
- The cross covariance update approximation works best if all nodes are highly correlated before the update
  - *As mentioned previously, the approximate update for P_{ik}, P_{jk} becomes exact if nodes i and j are highly correlated*
  - *Any cross covariances between a participating and non-participating node that are 0 before the update will remain 0, despite the fact that they would become non-zero in a CKF*
- These approximations are contradictory in nature: the gain approximation works best on uncorrelated systems, whereas the covariance update approximation works best on highly correlated systems

Distributed Algorithm: Distributed Covariance Intersection Filter

Covariance Intersection

Background

- An assumption implicit in Kalman filtering is that incoming measurements are independent and Gaussian-distributed
- Sometimes, incoming measurements are not independent, i.e. they are highly correlated, but the correlations are unknown
- Covariance Intersection (CI) is a method of conservatively fusing two or more covariance estimates with unknown cross-correlations
- The CI-fused covariance estimate is guaranteed to conservatively bound the true covariance for all possible cross-correlations between the two original estimates

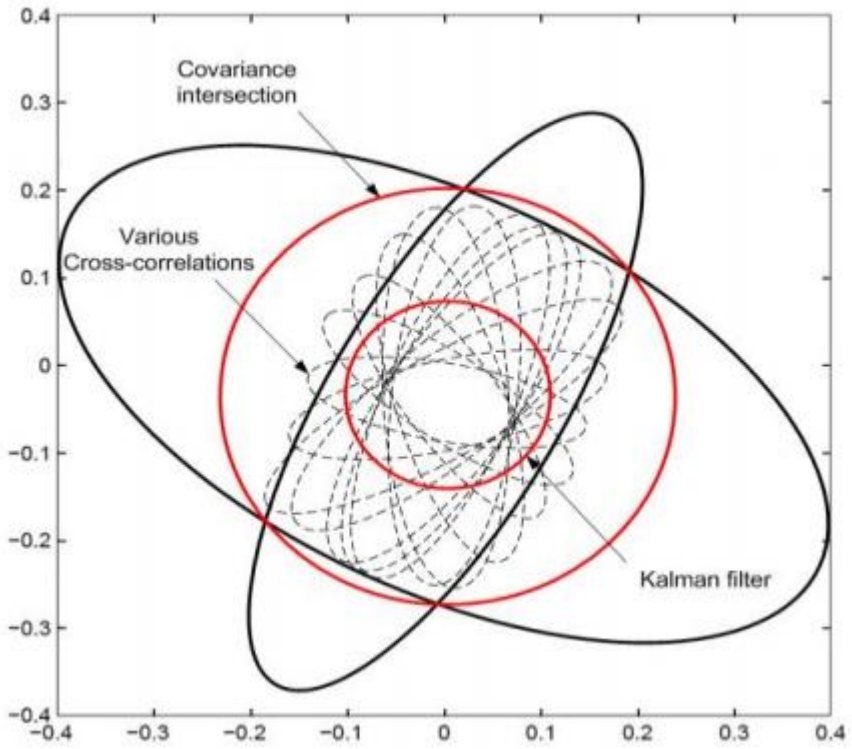


Image from [6]

Covariance Intersection

Equations

- Given two estimates $\hat{x}_1$ and $\hat{x}_2$ of the same state vector x, the associated covariances $\hat{P}_{11}$ and $\hat{P}_{22}$, and the cross covariance of the estimates $\hat{P}_{12}$, the estimates can be fused linearly as follows for some gains K_1, K_2 (* indicates fused estimate):

$$\hat{x}^* = K_1\hat{x}_1 + K_2\hat{x}_2 \tag{20}$$

$$\hat{P}^* = K_1\hat{P}_{11}K_1^T + K_1\hat{P}_{12}K_2^T + K_2\hat{P}_{12}^TK_1^T + K_2\hat{P}_{22}K_2^T \tag{21}$$

- If $\hat{P}_{12} = 0$, then the optimal covariance update is given by the Kalman filter[9]:

$$\hat{P}^*_{opt} = \left(\hat{P}_{11}^{-1} + \hat{P}_{22}^{-1}\right)^{-1}$$

- If $\hat{P}_{12} \neq 0$ but is unknown, then CI gives the following update rule:

$$\hat{P}^* = \left(\omega\hat{P}_{11}^{-1} + (1-\omega)\hat{P}_{22}^{-1}\right)^{-1} \tag{22}$$

$$\hat{x}^* = \hat{P}\left(\omega\hat{P}_{11}^{-1}\hat{x}_1 + (1-\omega)\hat{P}_{22}^{-1}\hat{x}_2\right) \tag{23}$$

- Here, the weight ω is chosen to minimize the trace of $\hat{P}^*$[8]:

$$\omega = \frac{tr\left(\hat{P}_{11}^{-1}\right)}{tr\left(\hat{P}_{11}^{-1}\right) + tr\left(\hat{P}_{22}^{-1}\right)} \tag{24}$$

Covariance Intersection

Generalization to fusion of N estimates

- The more general form of (22)-(24) for fusing an arbitrary number of estimates N is given by[8]:

$$\hat{P}^* = \left[\sum_{i=1}^{N} \omega_i \hat{P}_{ii}^{-1}\right]^{-1} \tag{25}$$

$$\hat{x}^* = \hat{P} \sum_{i=1}^{N} \omega_i \hat{P}_{ii}^{-1} \hat{x}_i \tag{26}$$

$$\omega_i = \frac{tr\left(\hat{P}_{ii}^{-1}\right)}{\sum_{k=1}^{N} tr\left(\hat{P}_{kk}^{-1}\right)} \tag{27}$$

- Note that $tr\left(\hat{P}_{ii}^{-1}\right) = 1/tr\left(\hat{P}_{ii}\right)$

Covariance Intersection Filtering

- CI allows any two or more estimates of the same state to be fused without knowledge of the cross-covariance between the estimates
- CI can be used to combine estimates from multiple Kalman filters (e.g. in a distributed sensing network where 2 or more sensors detect the same phenomenon)[7]
- CI can be used directly in the measurement absorption step of a filter when the incoming measurements have an unknown cross-correlation with the estimated state
  - *Since the CI equations are derived for fusion of states, measurement fusion requires some manipulation of the equations*

Measurement Fusion Using CI

Analogy to Kalman filter

- The generalized state fusion equation is

$$\hat{x}^* = K_1\hat{x}_1 + K_2\hat{x}_2$$

- Meanwhile, the measurement equation in a Kalman filter is

$$\hat{x}^+ = \hat{x}^- + K(z - H\hat{x}^-) = (I - KH)\hat{x}^- + Kz$$

where K is the Kalman gain, H is the measurement matrix, and z is the measurement vector

- In a linear system, $z = Hx + v$ where v is the measurement noise (assumed zero-mean and Gaussian-distributed) and x is the true state
  - *In a non-linear system, we can approximate* $z \cong h(x_{ref}) + H\Delta x + v$
- The KF measurement equation can then be rewritten as

$$\hat{x}^+ = (I - KH)\hat{x}^- + KHx$$

- Comparing to the state fusion equation, we see the following equivalences:

$$\hat{x}^+ \equiv \hat{x}^*$$
$$\hat{x}^- \equiv \hat{x}_1$$
$$x \equiv \hat{x}_2$$
$$(I - KH) \equiv K_1$$
$$KH \equiv K_2$$

Measurement Fusion Using CI

State update

- The CI state fusion equations are

$$\hat{P}^* = \left(\omega\hat{P}_1^{-1} + (1-\omega)\hat{P}_2^{-1}\right)^{-1} \quad (22)$$

$$\hat{x}^* = \omega\hat{P}^*\hat{P}_1^{-1}\hat{x}_1 + (1-\omega)\hat{P}^*\hat{P}_2^{-1}\hat{x}_2 \quad (23)$$

- Comparison with the generalized state fusion equation (1) gives

$$K_1 = \omega\left(\omega\hat{P}_1^{-1} + (1-\omega)\hat{P}_2^{-1}\right)^{-1}\hat{P}_1^{-1} \quad (28)$$

$$K_2 = (1-\omega)\left(\omega\hat{P}_1^{-1} + (1-\omega)\hat{P}_2^{-1}\right)^{-1}\hat{P}_2^{-1} \quad (29)$$

- In order to use these equations for measurement fusion instead of state fusion, we would like to use the measurement vector z in place of $\hat{x}_2$
- The covariance of the measurement z transformed to state space is given by:

$$\hat{P}_2 = (H^T S^{-1} H)^{-1}, \qquad S = cov(z)$$

- This allows us to rewrite (23) as

$$\hat{x}^* = K_1\hat{x}_1 + (1-\omega)\left(\omega\hat{P}_1^{-1} + (1-\omega)H^T S^{-1} H\right)^{-1} H^T S^{-1} H\hat{x}_2$$

- Recalling that $z = Hx$, we let $\hat{x}_2 \equiv x$, $\hat{x}^* \equiv \hat{x}^+$, and $\hat{x}_1 \equiv \hat{x}^-$:

$$\hat{x}^+ = K_1\hat{x}^- + \bar{K}_2 z \quad (30)$$

$$K_1 = \omega\left(\omega\left(\hat{P}^-\right)^{-1} + (1-\omega)H^T S^{-1} H\right)^{-1}\left(\hat{P}^-\right)^{-1} \quad (31)$$

$$\bar{K}_2 = (1-\omega)\left(\omega\left(\hat{P}^-\right)^{-1} + (1-\omega)H^T S^{-1} H\right)^{-1} H^T S^{-1} \quad (32)$$

Measurement Fusion Using CI

Covariance update

- The covariance of the fused state is given by

$$\hat{P}^* = \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} \hat{P}_1 & \hat{P}_{12} \\ \hat{P}_{21} & \hat{P}_2 \end{bmatrix} \begin{bmatrix} K_1^T \\ K_2^T \end{bmatrix} = K_1 \hat{P}_1 K_1^T + K_1 \hat{P}_{12} K_2^T + K_2 \hat{P}_{21} K_1^T + K_2 \hat{P}_2 K_2^T$$

- Since the cross covariance $\hat{P}_{12}$ is needed to calculate this, CI approximates the covariance as

$$\hat{P}_{CI}^* \approx K_1 \hat{P}_1 K_1^T + K_2 \hat{P}_2 K_2^T = \left(\omega \hat{P}_1^{-1} + (1-\omega)\hat{P}_2^{-1}\right)^{-1}$$

- This leads to the following equation for measurement fusion:

$$\hat{P}^+ = \left(\omega\left(\hat{P}^-\right)^{-1} + (1-\omega) H^T S^{-1} H\right)^{-1} \qquad (33)$$

Distributing the CI Filter

Motivation

- One of the key features of the CI filter is that it does not track the cross-covariance between states and measurements but still produces a conservative covariance estimate
- In a large distributed system with direct measurements between nodes, tracking the cross-covariances between nodes is one of the most difficult parts of formulating a distributed filter algorithm
  - *CI's agnosticism towards cross-covariance makes it a natural choice to be used in a distributed filter algorithm*
- While there are a number of possible ways to design a CI-based distributed filter algorithm, only one will be presented here

Distributing the CI Filter

Formulation

- Assumptions:
  - *Each node maintains only its own state and covariance*
  - *All relative measurements between nodes have an unknown, non-zero correlation*
  - *When nodes make a relative measurement, they can also exchange state and covariance information*
- State and covariance estimates are propagated through time independently by each node
- When a relative measurement (or set of relative measurements) is made, the node making the measurements only updates itself
  - *If the measurements are bi-directional such that both participating nodes have access to them, then each node updates itself independently using the measurements*
- Relative measurement updates are performed using the CI filter described above
- Measurements to an external reference point can be performed either with CI or with a standard Kalman filter update

Distributing the CI Filter

Equations

- Each node i tracks its state $\hat{x}_i$ and covariance matrix $\hat{P}_i$
- The state and covariance are propagated in time linearly as

$$\hat{x}_{i,t+1|t} = \phi_t \hat{x}_{i,t|t} \tag{34}$$

$$\hat{P}_{i,t+1|t} = \phi_t \hat{P}_{i,t|t} \phi_t^T + Q \tag{35}$$

where ϕ_t is the state transition matrix, Q is the process noise matrix

- The state can alternatively be propagated nonlinearly as

$$\hat{x}_{i,t+1|t} = f\left(\hat{x}_{i,t|t}, t\right)$$

- When node i makes measurements of nodes $j \in \mathcal{N}_i$ (where $\mathcal{N}_i$ is the set of nodes with which node i makes measurements), the measurements are absorbed as

$$\hat{x}_{i,t+1|t+1} = K_i \hat{x}_{i,t+1|t} + \sum_{j \in \mathcal{N}_i} \overline{K}_j z_j \tag{36}$$

$$K_i = \omega_i \left(\omega_i \left(\hat{P}_{i,t+1|t}\right)^{-1} + \sum_{j \in \mathcal{N}_i} \omega_j H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1} \left(\hat{P}_{i,t+1|t}\right)^{-1} \tag{37}$$

$$\overline{K}_j = \omega_j \left(\omega_i \left(\hat{P}_{i,t+1|t}\right)^{-1} + \sum_{j \in \mathcal{N}_i} \omega_j H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1} H_{i,ij}^T S_j^{-1} \tag{38}$$

Distributing the CI Filter

Equations cont.

- The covariance is then updated as

$$\hat{P}_{i,t+1|t+1} = \left(\omega_i \left(\hat{P}_{i,t+1|t} \right)^{-1} + \sum_{j \in \mathcal{N}_i} \omega_j H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1} \quad (39)$$

$$\omega_k = \frac{tr\left(\hat{P}_{k,t+1|t} \right)^{-1}}{\sum_{m \in \{\mathcal{N}_i, i\}} tr\left(\hat{P}_{m,t+1|t} \right)^{-1}} \quad (40)$$

- In the above equations, S_j is the covariance of the measurement from node j, given by

$$S_j = H_{j,ij} \hat{P}_{j,t+1|t} H_{j,ij}^T + R \quad (41)$$

where R is the measurement noise matrix

- Note the subscripts $H_{i,ij}$ versus $H_{j,ij}$. Here, $H_{i,ij}$ is used to refer to the partial derivatives of the measurement from node i to node j with respect to node i's state, while $H_{j,ij}$ refers to the partial derivatives of the measurement from node i to node j with respect to node j's state

Distributing the CI Filter

Algorithm summary

- At each timestep t, for each node i, do:

$$\hat{x}_{i,t+1|t} = f\left(\hat{x}_{i,t|t}, t\right)$$

$$\hat{P}_{i,t+1|t} = \phi_t \hat{P}_{i,t|t} \phi_t^T + Q$$

- At each node i with measurement partners $j \in \mathcal{N}_i$, do:

$$\omega_k = \frac{tr\left(\hat{P}_{k,t+1|t}\right)^{-1}}{\sum_{m \in \{\mathcal{N}_i, i\}} tr\left(\hat{P}_{m,t+1|t}\right)^{-1}}, \qquad k \in \{i, \mathcal{N}_i\}$$

$$S_j = H_{j,ij} \hat{P}_{j,t+1|t} H_{j,ij}^T + R$$

$$K_i = \omega_i \left(\omega_i \left(\hat{P}_{i,t+1|t}\right)^{-1} + \sum_{j \in \mathcal{N}_i} \omega_j H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1} \left(\hat{P}_{i,t+1|t}\right)^{-1}$$

$$\overline{K}_j = \omega_j \left(\omega_i \left(\hat{P}_{1,t+1|t}\right)^{-1} + \sum_{j \in \mathcal{N}_i} \omega_j H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1} H_{i,ij}^T S_j^{-1}$$

$$\hat{x}_{i,t+1|t+1} = K_i \hat{x}_{i,t+1|t} + \sum_{j \in \mathcal{N}_i} \overline{K}_j z_j$$

$$\hat{P}_{i,t+1|t+1} = \left(\omega_i \left(\hat{P}_{i,t+1|t}\right)^{-1} + \sum_{j \in \mathcal{N}_i} \omega_j H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1}$$

- Go to next timestep

Comparison to Kalman Filter

- The covariance update of the CI filter is given by:

$$\hat{P}_{i,t+1|t+1} = \left(\omega_i \left(\hat{P}_{i,t+1|t} \right)^{-1} + \sum_{j \in \mathcal{N}_i} \omega_j H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1}$$

- Meanwhile, the covariance update of the Kalman filter can be expressed in inverse information form as:

$$\hat{P}_{i,t+1|t+1} = \left(\left(\hat{P}_{i,t+1|t} \right)^{-1} + \sum_{j \in \mathcal{N}_i} H_{i,ij}^T S_j^{-1} H_{i,ij} \right)^{-1}$$

- Here, we can see that the Kalman filter update is a special case of the CI filter update where the weights $\omega_i = \omega_j = 1$
- This illuminates the possibility of a “tunable” CI filter, in which the weights are selected empirically ($0 < \omega \leq 1$) to provide the optimal performance on a real system

Non-Optimality of the DCIF

- CI does not track cross-correlations between nodes, and instead approximates its gain selection such that the covariance update is guaranteed to be conservatively bound the optimal Kalman filter update for the actual cross-correlation
- Since the cross-correlations are not tracked, the covariance update for measurements between nodes is also approximated
- In the best case, overly conservative covariance estimates will tend to produce noisier state estimates
- In the worst case, being overly conservative can cause the covariance estimate to grow without bound when there would otherwise have been enough information for the filter to converge

Formal Evaluation of Filter Performance

Formal Covariance Analysis of Distributed Filters

Motivation

- Each distributed filter makes some form of approximation in its covariance update, so the covariance estimates of the filters cannot be taken as the formal covariance of the system
- Formal covariance analysis requires that all cross-correlations between nodes be rigorously tracked and updated without approximation, as is done in a centralized filter
  - *Since the motivation behind using a distributed filter in the first place is to avoid using a centralized filter, this is not an analysis that would be feasible to perform during day-to-day spacecraft operations*
  - *Rather, this analysis is useful for evaluating and comparing distributed filter algorithms during the design phase of a program, so that an appropriate algorithm can be selected*

Formal Covariance Analysis of Distributed Filters

Methodology

- Rigorous tracking of all the covariances of the system simply requires that any operations performed on the state estimate of the system are also performed on the full state covariance (including all cross-correlations between nodes)
  - *This is what a centralized filter naturally does*
- For example, the state update of each node in a distributed system can be expressed as

$$x_i^+ = x_i^- + K_i\left(z_{i_{obs}} - z_{i_{model}}\right)$$

for which the corresponding covariance update (in Joseph's formulation) is:

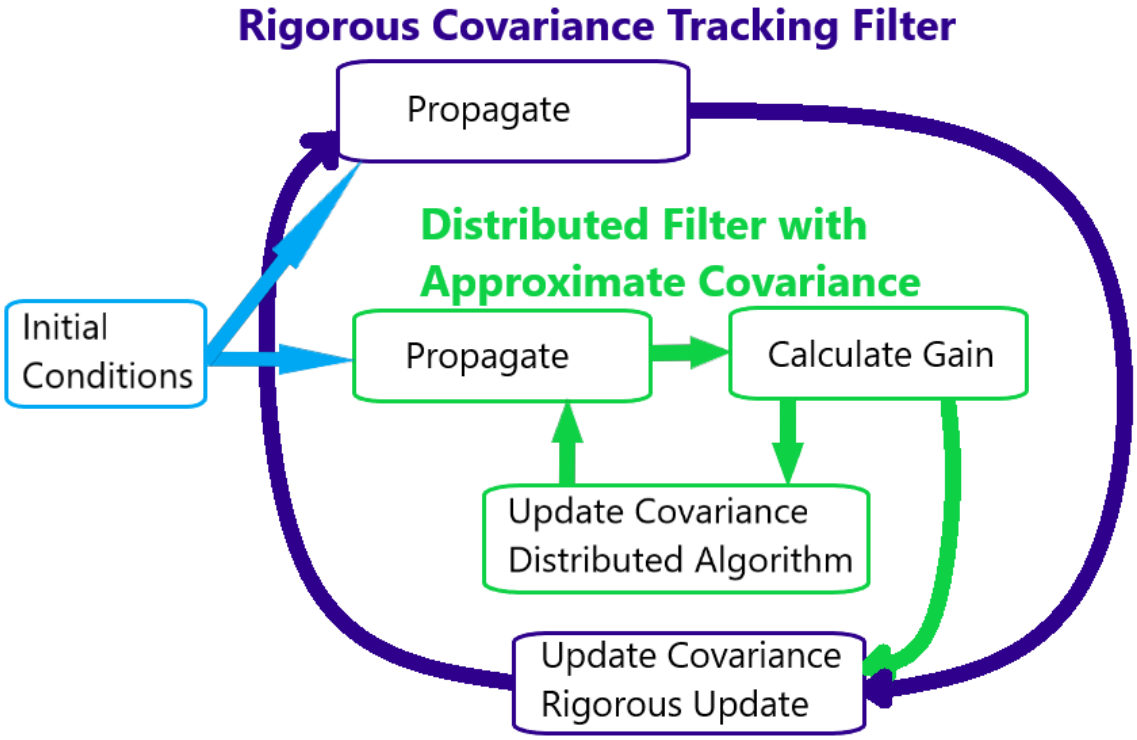
$$P_i^+ = (I - K_i H_i) P_i^- (I - K_i H_i)^T + K_i R_i K_i^T$$

- Application of the covariance update equation to the full covariance matrix P rather than the local node covariance matrix P_i, with the appropriate zero entries added to K, H, and R to match dimensions, produces the rigorously-updated full covariance matrix
  - *Note that this requires the use of the so-called Joseph's formulation of the covariance update, since the gain K is not the optimal Kalman gain for the full system*

Formal Covariance Analysis of Distributed Filters

Methodology

- The rigorous covariance update equations are equivalent to those in a centralized filter, but with a different gain matrix
- To perform the formal covariance analysis, a fully-correlated centralized filter was wrapped around the distributed filter to perform the rigorous covariance update using the approximate innovation gains calculated by the distributed filter



PDKF Formal Covariance Propagation

- The covariance estimates calculated by the PDKF algorithm are only approximations of the true covariance of the system
- In order to calculate the true covariance, the gain matrices generated by the PDKF algorithm should be applied rigorously to the full system to capture the full covariance matrix with cross-correlations between nodes included
- The gain matrix calculated by the PDKF for each node i is

$$K_{PDKF} = P_{ext}^{-} H^T (H P_{ext}^{-} H^T + R)^{-1} = \begin{bmatrix} K_i \\ K_{N_1} \\ \vdots \\ K_{N_n} \end{bmatrix}$$

- Only K_i is actually used in the PDKF update; the full K matrix is therefore formed from the K_i calculated at each node:

$$K_{full} = \begin{bmatrix} K_1 \\ \vdots \\ K_i \\ \vdots \\ K_n \end{bmatrix}$$

PDKF Formal Covariance Propagation

- The rigorous covariance is updated using the gains approximated by the PDKF as

$$P_{rig}^{+} = \left(I - K_{full}H\right)P_{rig}^{-}\left(I - K_{full}H\right)^{T} + K_{full}RK_{full}^{T}$$

DCL Formal Covariance Propagation

- The covariance estimates calculated by the distributed filter algorithm are only approximations of the true covariance of the system
- In order to calculate the true covariance, the gain matrices generated by the distributed filter should be applied rigorously to the full system to capture the full covariance matrix with all cross-correlations between nodes included
- The gain matrix calculated by the DCL for each node pair (i, j) with node-to-node measurements is

$$K_{DCL} = \begin{bmatrix} K_i \\ K_j \end{bmatrix} = \begin{bmatrix} P_{ii}^- H_i^T + P_{ij}^- H_j^T \\ P_{jj}^- H_j^T + P_{ji}^- H_i^T \end{bmatrix} S^{-1}$$

- The gain submatrices K_i, K_j and measurement submatrices H_i, H_j are filled into their corresponding slots in the full matrices K_{full}, H_{full}

$$K_{full} = \begin{bmatrix} 0 & \cdots & K_i^T & \cdots & 0 & \cdots & K_j^T & \cdots & 0 \end{bmatrix}^T$$
$$H_{full} = \begin{bmatrix} 0 & \cdots & H_i & \cdots & 0 & \cdots & H_j & \cdots & 0 \end{bmatrix}$$

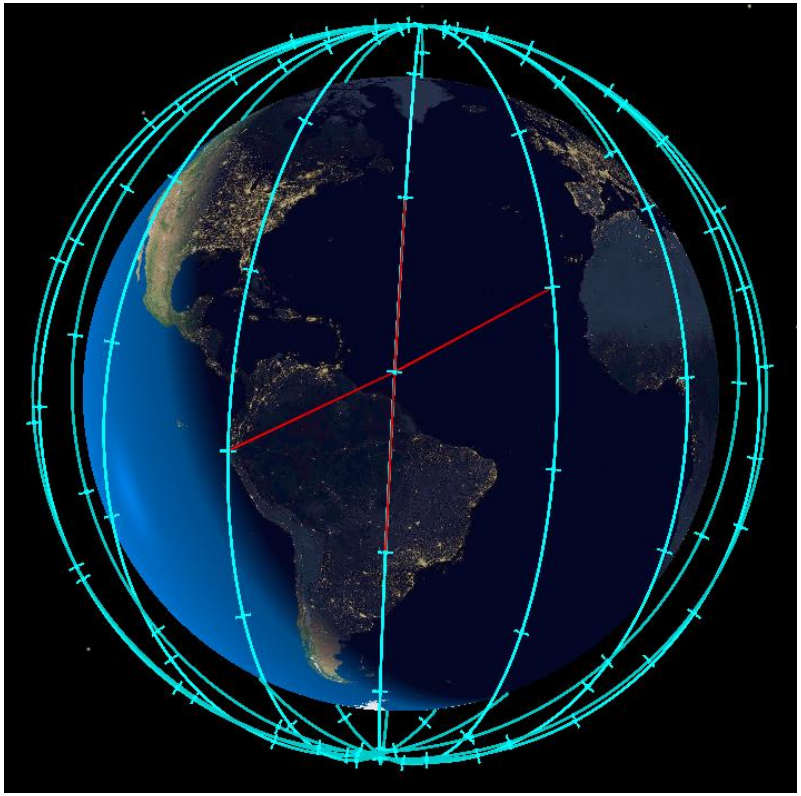
- The rigorous covariance is updated using the approximated gains as

$$P_{rig}^+ = \left(I - K_{full} H_{full}\right) P_{rig}^- \left(I - K_{full} H_{full}\right)^T + K_{full} R K_{full}^T$$

Simulation Results

Simulation Scenario

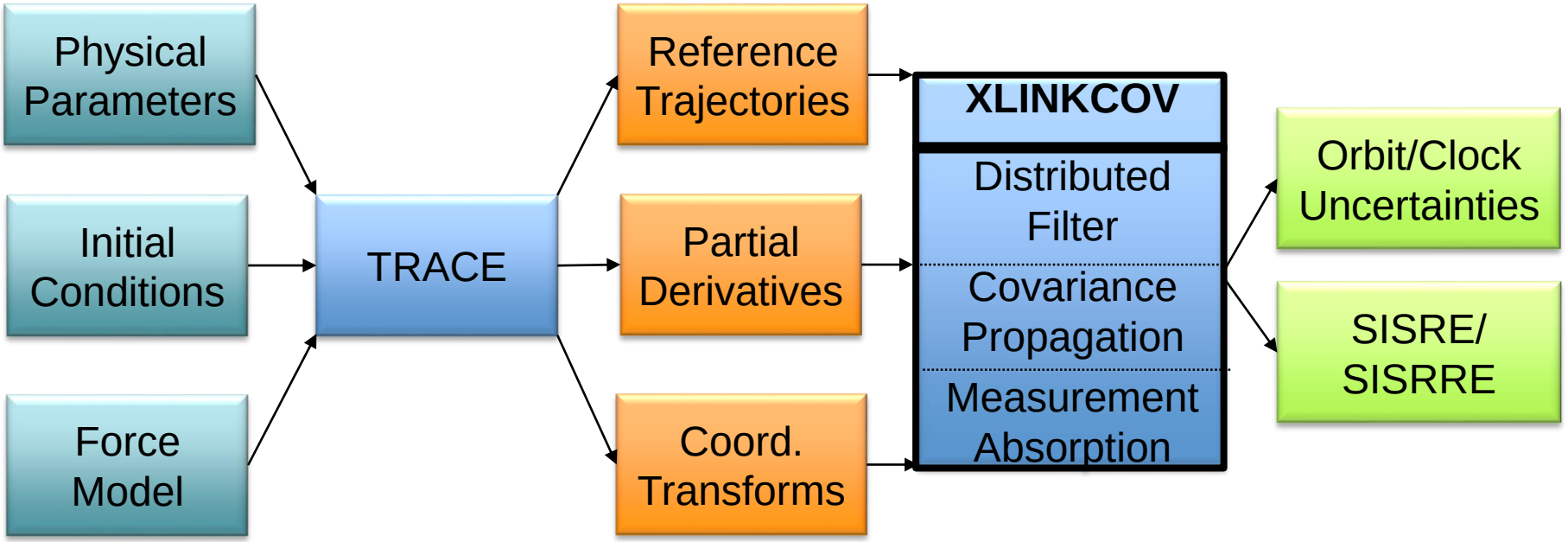
- A streets-of-coverage constellation of 112 LEO satellites in circular orbit at 1020 km altitude was simulated
- Each satellite received crosslink pseudorange measurements from its 4 nearest neighbors (links shown in red in figure)
  - *Bi-directional 1-way range measurements were assumed – each satellite in a crosslink pair takes its own 1-way range measurement, and then shares that measurement with the other*
- Each satellite estimated its own position (3 states), velocity (3 states), solar radiation pressure parameter, and clock offset (phase, frequency, frequency drift rate)



XLINKCOV[1]

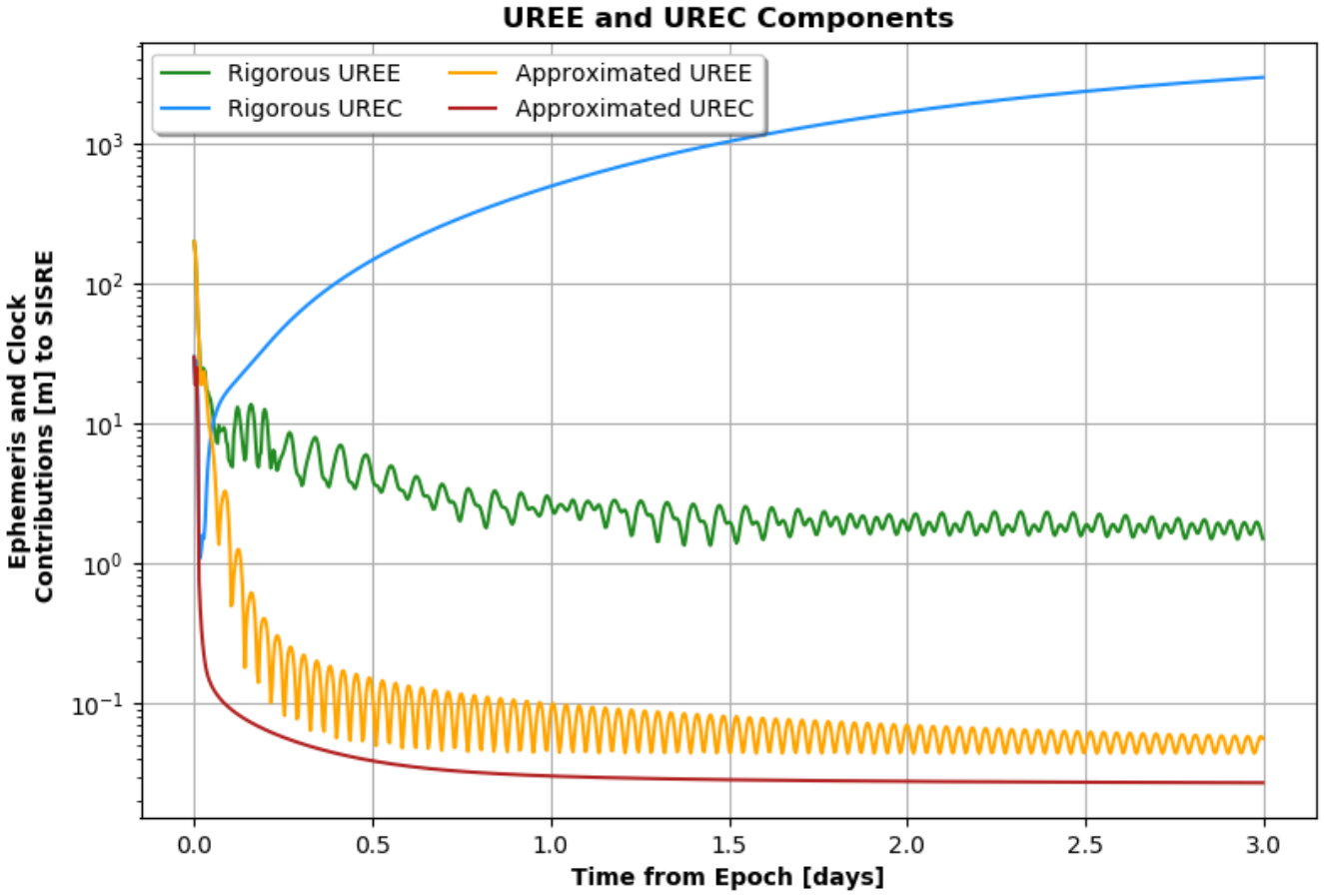
Crosslink simulation tool

- Crosslink simulation built on top of TRACE
- Runs a distributed filter to process simulated crosslink measurements and estimate orbit and clock uncertainties
  - *Different distributed algorithms were plugged in here for analysis*
- The orbit and clock uncertainties computed from the filter are used to compute the Signal-In-Space Range Error (SISRE) and Signal-in-Space Range-Rate Error (SISRRE) of a satellite averaged over its visibility footprint



Formal Covariance Analysis - DCL

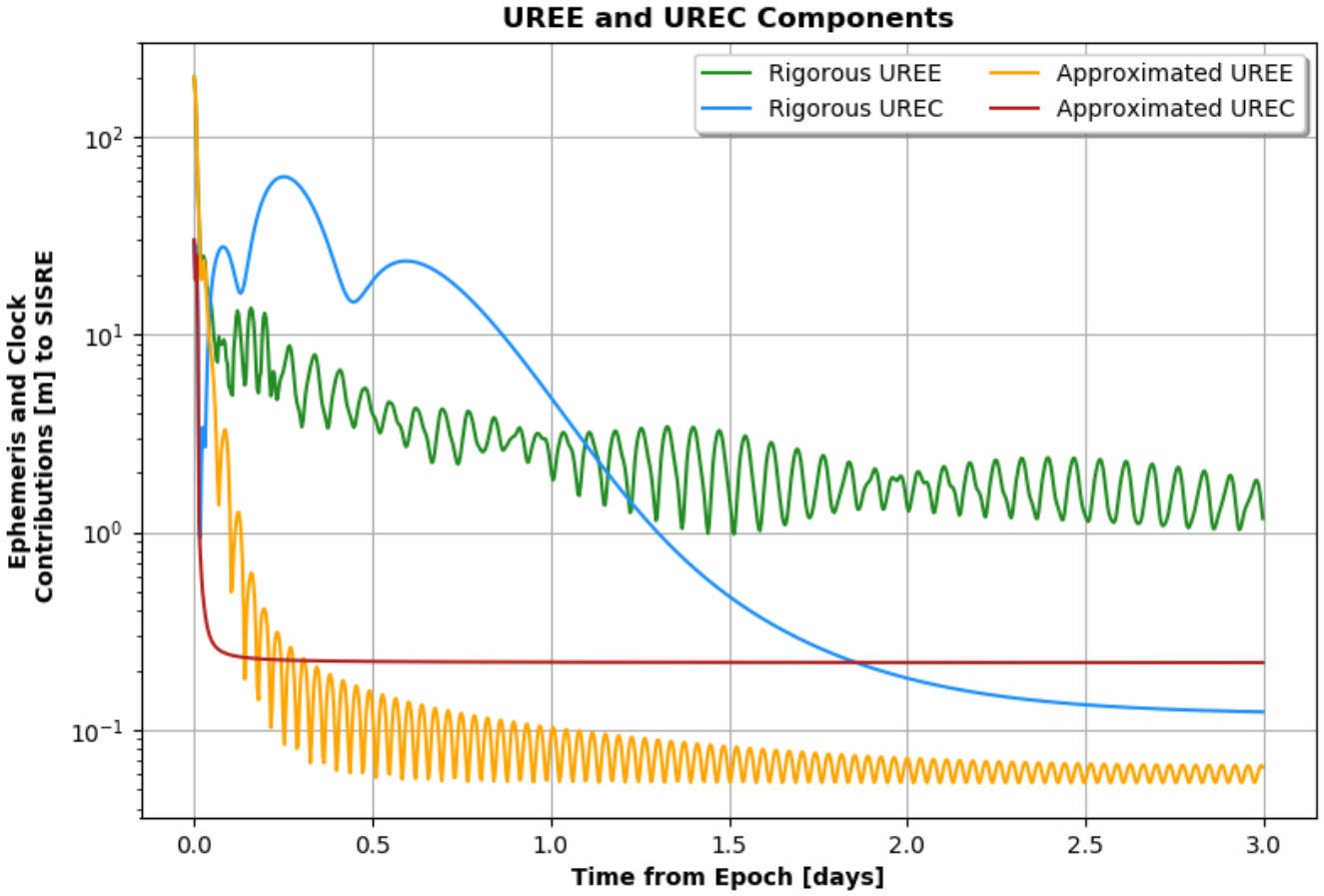
Nominal Results



The DCL algorithm is not able to properly track the clock covariance, leading to clock divergence

Formal Covariance Analysis - DCL

After Filter Tuning



Clock divergence can be prevented by tuning the process noise in the distributed filter

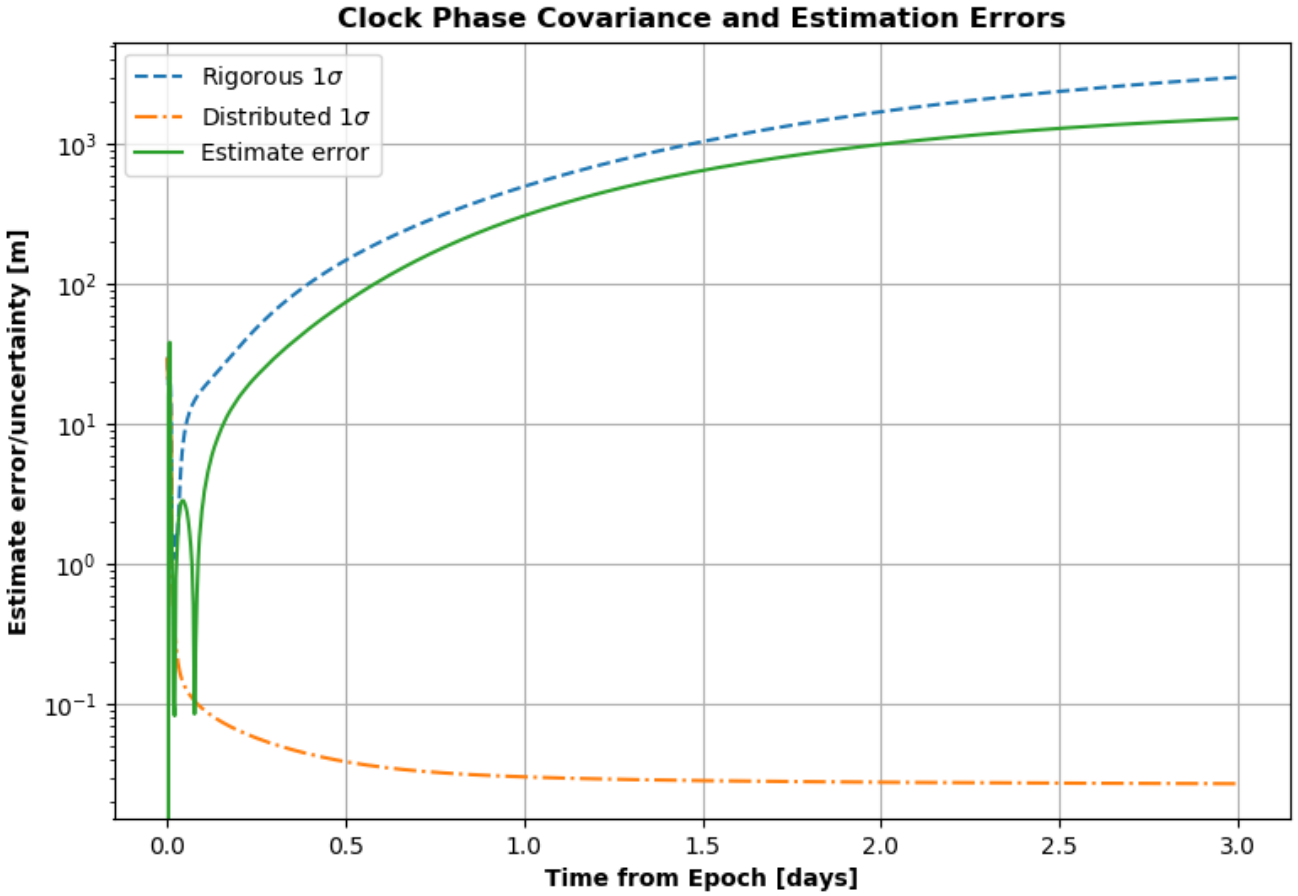
Formal Covariance Analysis Verification

Using Simulated Measurements

- Satellite trajectory perturbations and noisy measurements were simulated, and corrections to the nominal trajectory were estimated according to the distributed filter algorithm
  - *The trajectory was not re-linearized around the perturbed states to save compute time, so the filter only maintained stability for ~3 days before the total perturbations grew so large that linearization errors caused filter divergence*
    - Use of an extended Kalman filter formulation (which re-linearizes of the trajectory around the current predicted state) would likely eliminate this issue
- Estimated corrections were compared to the truth perturbations to calculate estimation errors
- The simulation was run with several different random seeds to verify that the results were not a fluke, but a full Monte Carlo analysis was not performed
  - *Results on the following slides are shown for a single run*
  - *Absolute value of the estimate errors are shown to allow plotting on a log scale*

Formal Covariance Analysis Verification

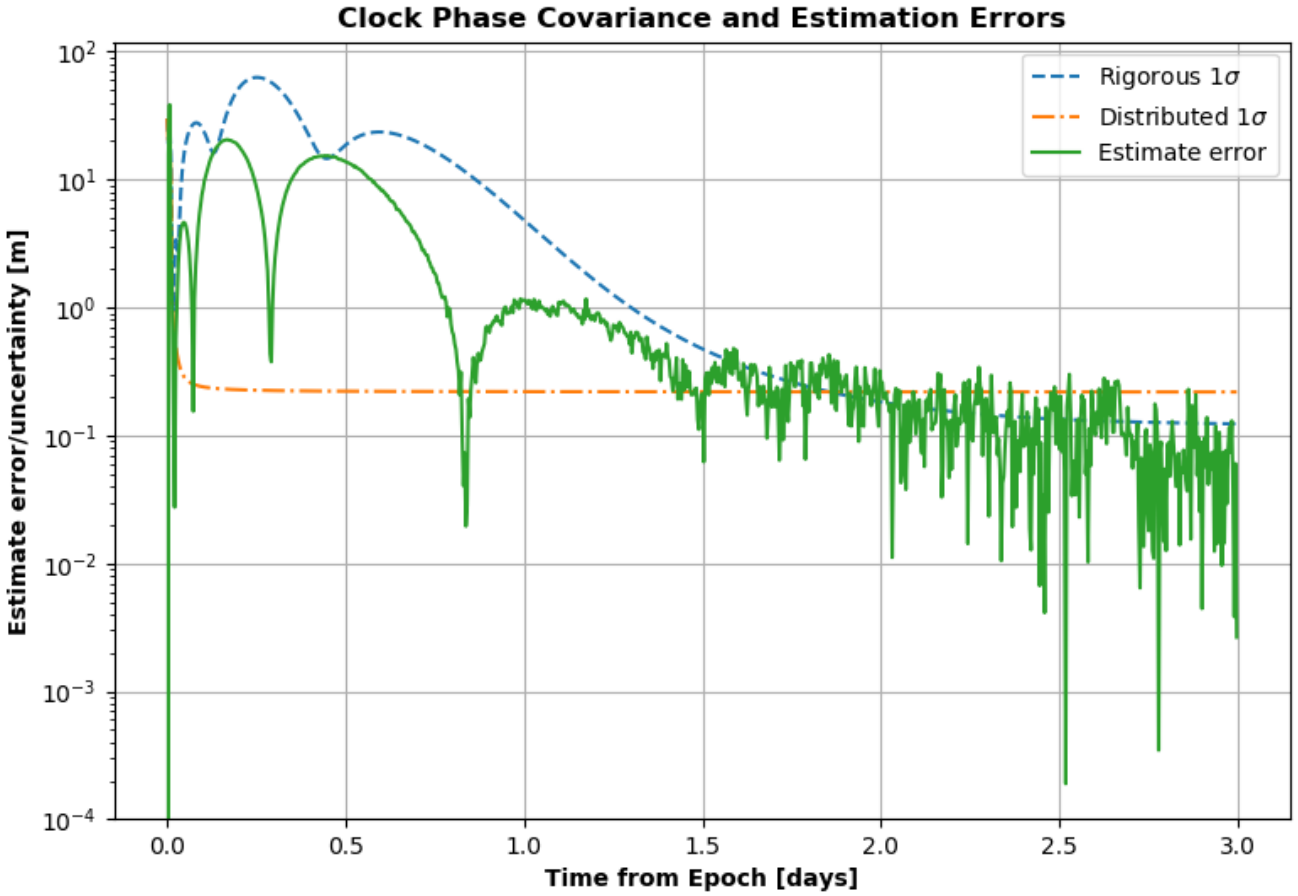
Comparison to Estimation Errors using Simulated Measurements – Un-tuned clock phase estimate error



Simulated clock error diverges as predicted in the nominal case

Formal Covariance Analysis Verification

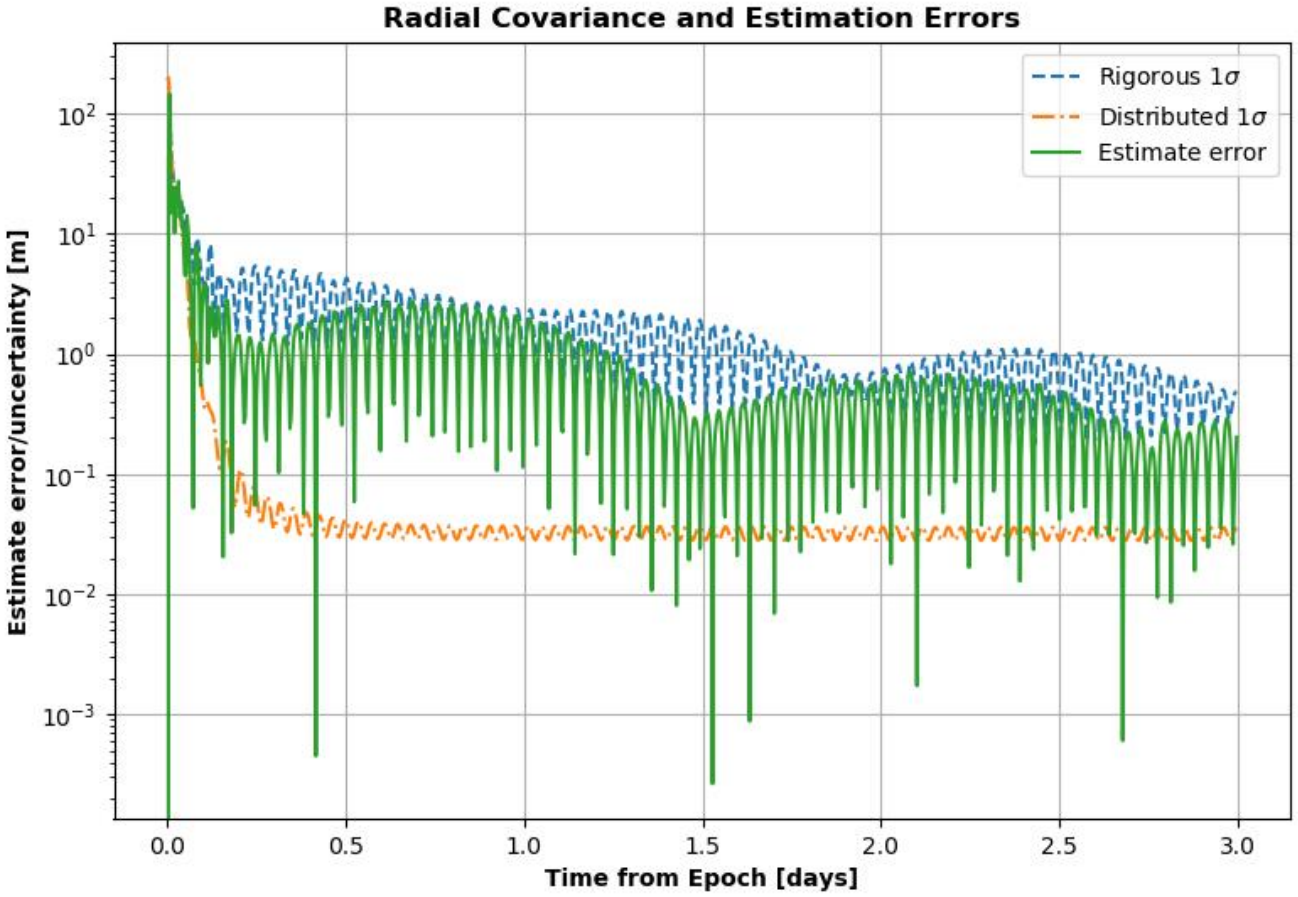
Comparison to Estimation Errors using Simulated Measurements – Clock phase estimate error after tuning



Simulated clock error converges after tuning – again as predicted

Formal Covariance Analysis Verification

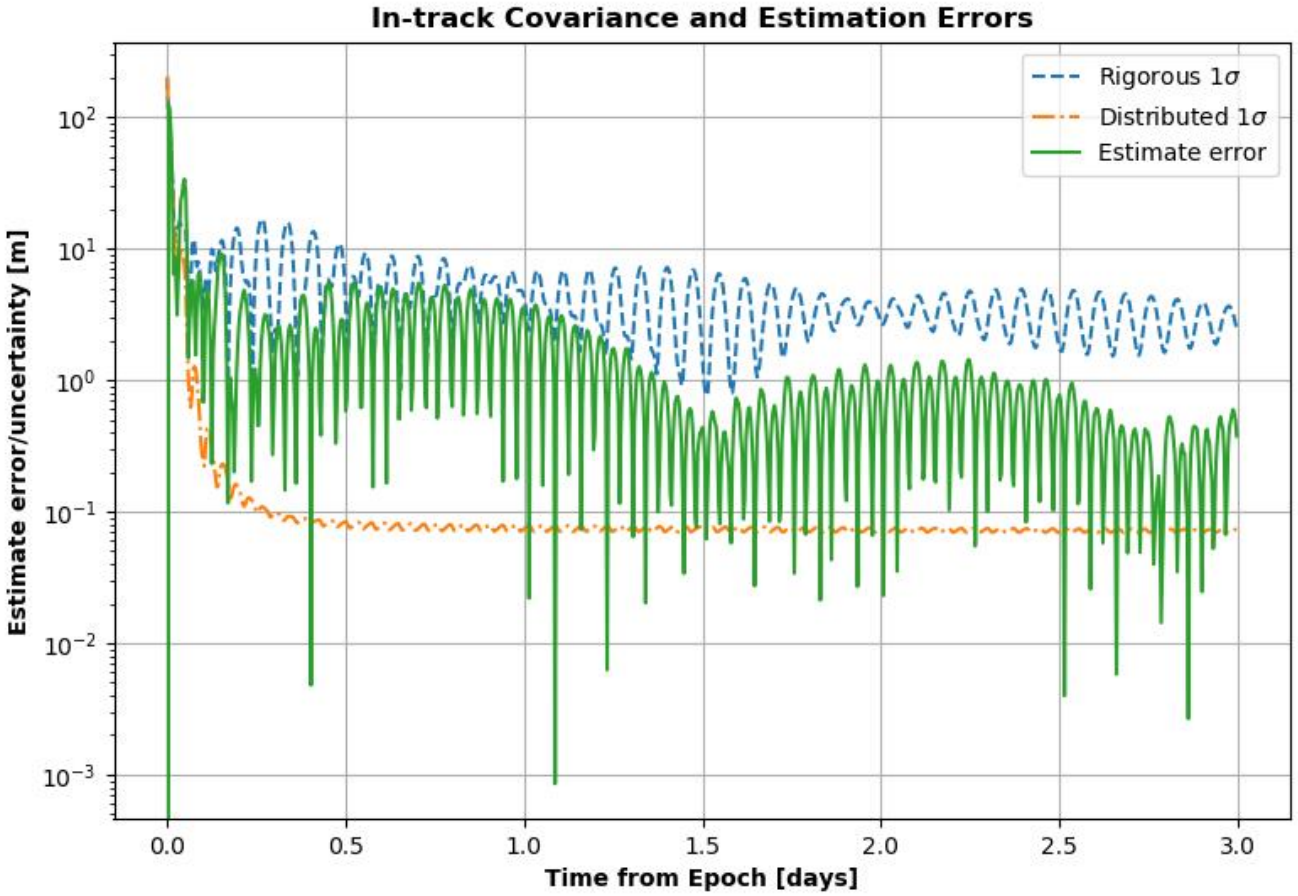
Comparison to Estimation Errors using Simulated Measurements – Radial position estimate error



Errors in the position components also fall within the rigorously-calculated covariance envelope (similar trends seen for other position components)

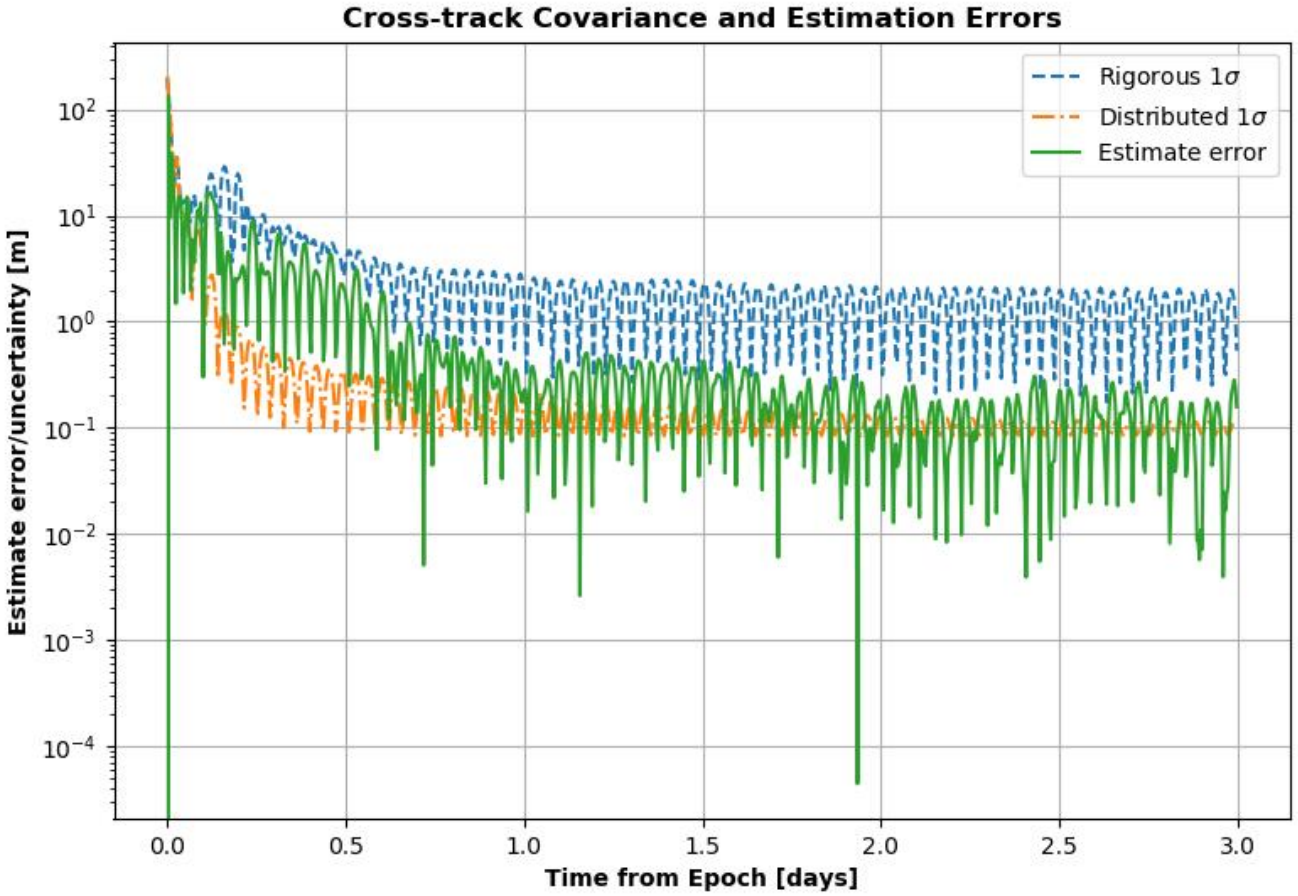
Formal Covariance Analysis Verification

Comparison to Estimation Errors using Simulated Measurements – In-track position estimate error



Formal Covariance Analysis Verification

Comparison to Estimation Errors using Simulated Measurements – Cross-track position estimate error



Comments on Results

- Tuning of the distributed filter was accomplished by multiplying the filter's position and velocity process noise by a factor of 2.7 and multiplying the clock process noise by a factor of 5000
  - *The increase in clock process noise is so large that it causes the filter's steady-state clock covariance estimate to actually be larger than the rigorous covariance*
  - *The increase in position/velocity process noise is less significant, so the filter's steady-state position covariance is significantly under-estimated compared to the rigorous covariance*
    - The position/velocity process noise cannot be increased arbitrarily to a point where the filter's steady-state matches the rigorous steady-state, as the filter loses stability before that point
- The convergence rates and magnitudes of the estimation errors match much closer to the envelope of the rigorous covariance than to the filter's internal covariance estimate

Filter Comparison

Algorithm	Run Time	Internal SISRE	Formal SISRE
CKF	5m 21s	0.07379 m	0.07379 m
PDKF	14s	0.05479 m	Did not converge
DCL	15s	0.2688 m	2.9520 m
DCIF	20s	Did not converge	Did not converge

- The centralized filter offers the most accurate navigation solution, which was to be expected since it tracks all cross-correlations
  - *Computational burden and run-time are the primary reasons that the CKF is not preferred for proliferated constellations*
- Distributed filters tend to greatly under-estimate their solution covariance, so care needs to be taken if using their internal covariance estimates for other purposes (e.g. conjunction analysis, integrity checks, etc.)
- DCL was the only distributed algorithm found to be convergent after light manual tuning

  - *Note: Signal-in-Space Range Error (SISRE), also called User Range Error (URE), is the average projection of the satellite position and clock errors onto the line-of-sight of terrestrial users – in other words, the error they would see in a range measurement from that satellite. This is a common performance indicator for the orbit determination of GPS satellites and provides a convenient single metric with which to gauge the orbit determination and clock synchronization results*

Conclusion

Summary

- Reviewed existing literature on distributed filter algorithms
- Developed methodology and simulation framework to analyze and compare the performance of different distributed filter algorithms
  - *Methodology verified via simulation*
- Tested several promising algorithms and found only one of the algorithms tested could be easily tuned to converge

Future Work

- The algorithms tested, even after tuning, perform worse than a centralized Kalman filter by an order of magnitude or more
  - *Further research into developing a distributed algorithm tailor-made for proliferated constellations might produce better performance*
- Since the covariance reported by distributed filters tends to be overly optimistic, it is of interest to research ways to obtain a more accurate estimate of the covariance for use in conjunction analysis and integrity monitoring
- Prototype demonstration of distributed filter operation using Raspberry Pis (coordinated with Chain Mail: Plate Ventures Blitz)
  - *Future work might involve real (rather than simulated) measurements and dynamic movement of some sort of bot*
  - *Eventual goal would be an on-orbit demonstration using a cluster of cubesats*

References

- [1] Binder, Andrew J.; Won, Jeffrey; and Cruz, Jaime Y. "Space Transport Layer: Hosted Navigation Payload Performance." Aerospace Report No. TOR-2019-01554. April 22, 2019.
- [2] Chong, Chee-Yee; Chang, Kuo-Chu; Mori, Shozo. "Comparison of Optimal Distributed Estimation and Consensus Filtering." *Proceedings of the 19th International Conference on Information Fusion,* Heidelberg, Germany, July 5-8, 2016.
- [3] Luft, L., Schubert, T., Roueliotis, S. I., and Burgard, W., "Recursive Decentralized Collaborative Localization for Sparsely Communicating Robots," *Robotics: Science and Systems*, 2016.
- [4] Khan, Usman A. and Moura, Jose M. F. "Distributing the Kalman Filter for Large-Scale Systems." *IEEE Transactions on Signal Processing*, Vol. 56, No. 10, October 2008. pp 4919-4935.
- [5] Olfati-Saber, R. "Distributed Kalman Filtering for Sensor Networks." *Proceedings of the 46th IEEE Conference on Decision and Control,* New Orleans, LA, USA, Dec. 12-14, 2007. pp 5492-5498.

References

- [6]: T. Jin, Multi-Sensor Data Fusion with Covariance Intersection in Robotic Space with Network Sensor Devices, *J. Control and Automation*, 9(7), pp. 71-78, 2016.
- [7]: S. J. Julier, J. K. Uhlmann, A Non-divergent Estimation Algorithm in the Presence of Unknown Correlations, *Proceedings of the American Control Conference*, Albuquerque, NM, June 1997, pp. 2369-2373.
- [8]: W. Niehsen, Information Fusion based on Fast Covariance Intersection Filtering, *Proceedings of the 5th International Conference on Information Fusion*, Annapolis, MD, July 8-11, 2002.
- [9]: L. Chen, P. O. Arambel, R. K. Mehra, Estimation under Unknown Correlation: Covariance Intersection Revisited.