A Distributed, Redundant Navigation and Fault Detection System for DARPA System F6

Jason Schmidt* and Michael Phillips*
Emergent Space Technologies, Inc, Greenbelt, MD, 20770, USA

Emergent Space Technologies has developed a distributed navigation architecture for the DARPA System F6 LEO Cluster Flight Mission that uses both GPS pseudorange measurements as well as communication system derived relative range measurements. The architecture consists of two filters on each module (a GPS filter and a relative range filter) in addition to a navigation manager that selects which filter result to publish based on a number of performance metrics such as normed innovation squared and other measurement rejection criteria. This architecture can maintain accurate relative navigation even if there are interruptions to GPS signals or failures to key GPS receiver components. This innovation is a significant improvement on the state-of-the art and particularly beneficial to clusters with more than two modules and to small satellites that cannot handle the additional weight for backup GPS receiver components.

Nomenclature

$\mathbf{a_c}$	acceleration due to thrust, $^{m}/_{s^2}$
$\mathbf{a_d}$	acceleration due to drag, $^{m}/_{s^2}$
$\mathbf{a_g}$	acceleration due to gravity, $^{m}/_{s^2}$
b	clock bias, m
d	clock bias drift rate, $^{m}/_{s}$
$E[\cdot,\cdot]$	expectation operator
F	linearized dynamics function
$\boldsymbol{f}(\mathbf{x})$	state dynamics function
H	measurement sensitivity matrix (linearized measurement function)
$h(\mathbf{x})$	measurement function
I	identity matrix
K	Kalman gain
P	state covariance
Q	discrete process noise strength
R	measurement covariance
$\mathbf{r}$	position vector, m
$\mathbf{v}$	velocity vector, $^{m}/_{s}$
$\mathbf{v}_r$	velocity relative to the atmosphere, $^{m}/_{s}$
$\mathbf{x}$	state vector
y	measurement

Greek

β	ballistic coefficient, $^{kg}/_{m^2}$

*Aerospace Engineer, Emergent Space Technologies, 355 Teller St. Suite 200, Lakewood, CO 80226

Δt_c	thrust duration length, s
$\Delta\mathbf{v}$	delta-v, $^{\mathrm{m}}/_{\mathrm{s}}$
$\Delta\mathbf{x}$	cumulative state innovation
η	measurement noise
ρ	atmospheric density, $^{\mathrm{kg}}/_{\mathrm{m}^3}$
ρ_0	reference atmospheric density, $^{\mathrm{kg}}/_{\mathrm{m}^3}$
Φ	state transition matrix
$\boldsymbol{\omega}$	process noise
ω_E	Earth rotation rate, $^{\mathrm{rad}}/_{\mathrm{s}}$

Decorations

$\tilde{o}$	truth
$\hat{o}$	estimate
$\dot{o}$	time derivative

Subscripts

i	time i
j	time j
$i \to j$	from time i to time j
r	position
v	velocity

Superscripts

$-$	pre-update
$+$	post-update
$ICRF$	International Celestial Reference Frame
$ITRF$	International Telestial Reference Frame
RIC	Radial, Intrack, Crosstrack
T	transpose

Symbols

$\mathbf{0}$	zero matrix
$[\cdot]_\times$	cross-product matrix

I. Introduction

The DARPA System F6 program was conceived to "demonstrate the feasibility and benefits of disaggregated—or fractionated—space architectures, wherein the functionality traditionally co-resident within a single, large, 'monolithic' satellite is delivered by a cluster of wirelessly-interconnected modules capable of sharing their resources and utilizing resources found elsewhere in the cluster."[1] Fault Detection Isolation and Recovery (FDIR) is of heightened importance in a satellite cluster because every additional module increases the probability of a failure. While GPS is the standard for space-based navigation, having a backup receiver and multiple antennas adds additional weight and still leaves the cluster susceptible to common GPS failures or interruptions. The System F6 mission has the additional challenge of maintaining relative navigation that is sufficiently accurate to perform safe and accurate maneuvers when there are more than two modules in a cluster. Traditional relative navigation sensors such as Lidar and cameras can only track one other vehicle at a time and require the vehicle to actively slew to track the target.

Emergent Space Technologies has developed a distributed navigation system that uses relative range information (available from the omni-directional wireless communication system) as a backup measurement source in case there are interruptions or failures in the primary GPS based navigation system. This measurement source can range to more than one other satellite at a time and does not require slewing to track the target. This navigation system is very beneficial for small spacecraft that do not have room for backup GPS hardware.

Relative range based navigation works best when there are at least four modules in the cluster and the geometry of the modules allows concurrent orthogonal measurements to be collected. However, observability

can be maintained with fewer modules if the relative trajectories provide observability along all three axes. Thus, there are some constraints on the cluster geometry, but they are not terribly onerous.[2]

Finally, the modules exchange the GPS Satellite Vehicles (GPS SV) that are being tracked by each GPS receiver. The GPS filters process GPS SV that are common to all modules in the cluster. This eliminates the largest source of relative navigation errors when processing GPS errors in independent filters.[3]

The navigation system has been tested in high fidelity, open loop simulations. The navigation system is part of a larger cluster flight application that contains, among other things, guidance and control software.

This fault-redundant navigation system is a significant improvement to the current state of the art. Clustered satellite flight is still relatively new, with only a few missions having been performed successfully. PRISMA is a recent European Space Agency mission that represents the state of the art in relative navigation and demonstrated Carrier Phase Differential GPS (CDGPS).[4,5] It has successfully demonstrated CDGPS navigation filters flying on-orbit with large attitude changes and thruster accelerations. However, only the most basic FDIR techniques were implemented. These consisted of transmitting all software input, output, and meticulously enumerated status bytes (indicating anomalies or deficiencies detected during execution) to the ground. Furthermore, while CDGPS is the current state of the art in relative navigation, it is totally dependent on access to GPS signals and receiving hardware. Disruption to the signal due to space weather effects, jamming, or failure in any of the receivers may totally inhibit CDGPS operation. It can be clearly seen that current CDGPS techniques have much to benefit from improved FDIR.

While PRISMA only involved two vehicles, the Synchronized Position Hold Engage Reorient Experimental Satellites (SPHERES), at the MIT Space Systems Laboratory has flown clusters of several autonomous fan powered modules on the International Space Station (ISS).[6] SPHERES implemented basic sensor fault detection by monitoring the residuals in the extended Kalman Filter (EKF) processing time of flight data from Ultrasonic receivers.[7,8] SPHERES also implemented thruster fault detection and isolation by comparing the accelerometer and gyro measurements with the expected values when a maneuver was performed.[9]

II. Overview of Navigation System

The navigation software is the same on each module. As seen in Figure 1, each module has a GPS filter, a relative range filter, and a navigation manager. FDIR algorithms within the filters and the navigation manager choose which navigation solution should be published. The navigation manager publishes to and collects from other modules navigation solutions and lists of tracked GPS SV. The relative range and GPS filters are identical software that are configured differently. Though not shown in the figure, maneuver information for all modules is also distributed to all modules.

An overview of the navigation software can be seen in Figure 2. While this is a simplification of the real system, key inputs, outputs, and functionality of the solution are apparent. The `NavManagerPrivate` (NavManager) contains an assortment of C++ methods, codegened MATLAB functions (MATLAB converted into C), data, and objects, including N instances of the `NavFilterPrivate` (NavFilter) class, which contain independent navigation filters. Only the first NavFilter is shown in expanded form, though each NavFilter has the same interfaces.

An initial configuration file is read at startup, but once operational, the component is totally configurable by way of the NavManagerConfig and NavFilterConfig inputs on the bottom left of Figure 2. Key inputs include GPS ephemeris data, maneuver data, NavSolution data from other modules in the cluster, and measurements. The Operating System (OS) calls the `NavScheduler` method at regular intervals (e.g. 1 second). The `NavScheduler` runs the extended Kalman filters (EKF) at scheduled intervals. It also publishes the results from the navigation filters at scheduled intervals. A variety of fault detection tests performed by `monitor_filter` on data collected by `monitor_measurements` are published along with the filter's navigation solution. The `select_nav_solution` function evaluates the results of these tests to select a filter solution to publish. Though not shown here, an unfiltered PVT solution can be selected and published if desired.

II.A. Navigation Basic IO and Functionality

II.A.1. Measurements

Measurements can be GPS pseudorange or relative range measurements to other modules. All measurements are processed as scalars and are inserted into a ring buffer unique to each NavFilter by the `collate_measurements` function. Though not shown in Figure 2, the NavManager directs each measurement to the NavFilter that

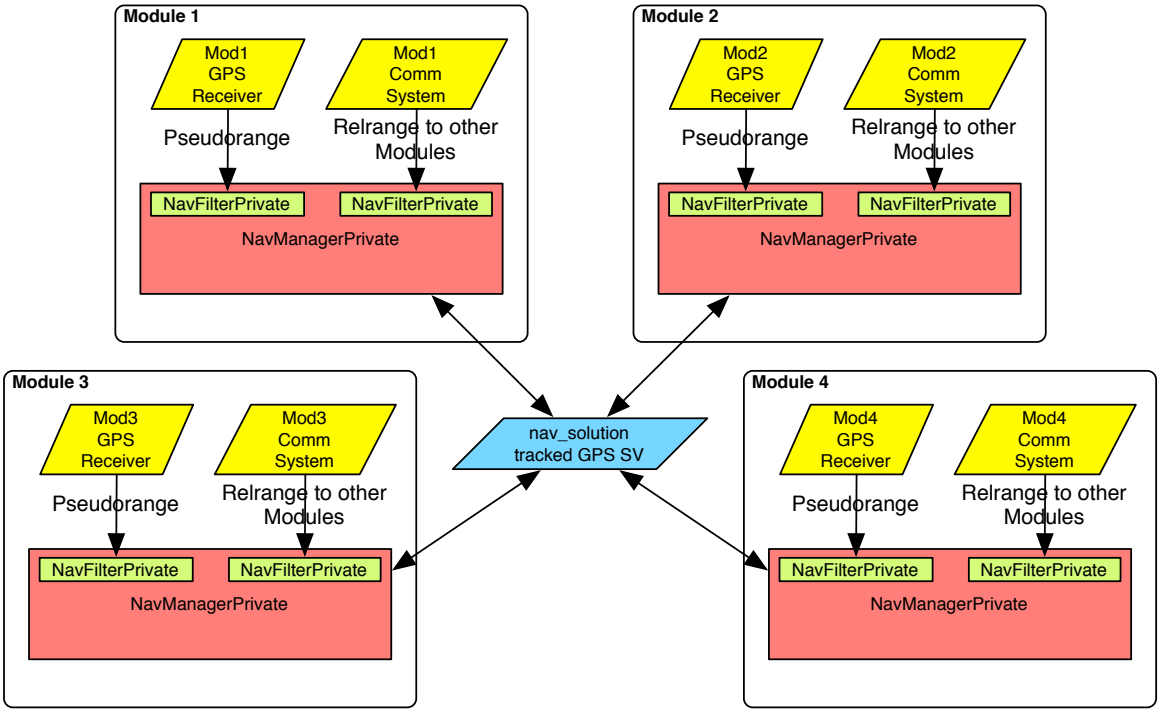

Figure 1. Overview of navigation deployment in a four module cluster.

is configured to process it and ignores measurements that none of the NavFilters have been configured to consume. The collate_measurements function performs basic checks on the measurements and if any are found to be invalid, the monitor_measurements will log the measurement and the reason it was rejected.

II.A.2. NavSolutions

The NavSolution data published by each NavManager object is consumed by all the other NavManager objects. The NavSolution data contains position, velocity, unmodeled accelerations (if any), and associated covariances for the module that NavManager is responsible for. This data can be used to update NavFilter states by way of a state replace or the position and velocity data can be processed as a measurement.

The NavSolution data also contains the GPS satellites that are in view by that module. The `collate_gps_in_view` function collects that data and makes it available to the `select_common_gps` function. This allows the GPS filter to process GPS satellites that are common to all the modules in the cluster as discussed in Section II.C.2.

II.A.3. Maneuvers

The nature of a distributed cluster makes bandwidth a valuable commodity. As a result, Navigation approximates maneuvers as a constant acceleration with a discrete start and stop time. The maneuvers for each module are distributed throughout the cluster and collated by each Navigation component with the `collate_mnvrs` function. This data is pushed to each NavFilter object in the NavManager.

II.A.4. GPS Ephemeris

In order to process GPS pseudorange measurements, the location of the GPS satellites must be derived from the GPS ephemeris data published in the GPS signal. This data is published by each receiver and ingested by the NavManager using the `collate_gps_ephem` function. This data is pushed to each NavFilter object in the NavManager.

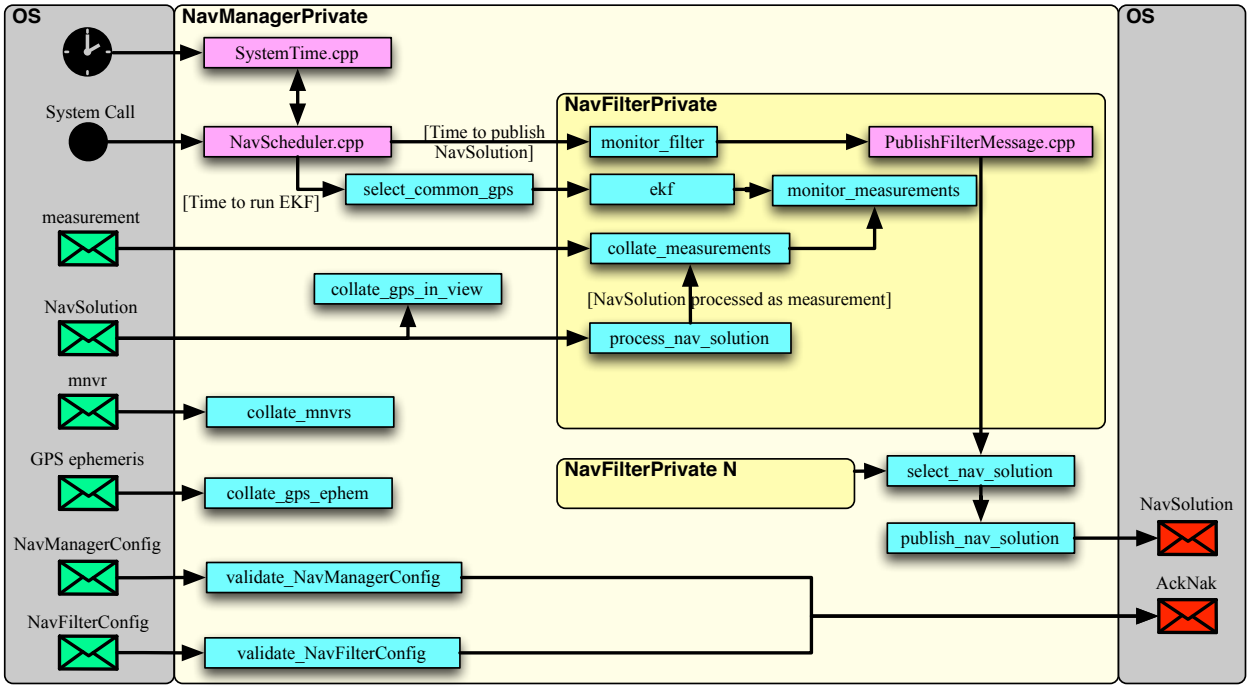

Figure 2. Overview of navigation input, output, and basic architecture.

II.A.5. Extended Kalman Filter

The EKF processes can be broken down into the basic steps shown in Figure 3. The state is completely defined by the filter_message structure with the state time defined by filter_message.t.

At the beginning of the algorithm the final propagation time (t_{end}) is calculated using the current time and a flooring function that floors the current time into fixed intervals (e.g. 10 seconds). This reduces the sensitivity of the function to run time and keeps filters running on different modules (with different clocks) in sync. Next, the measurement ring buffer is parsed to find the start and end indices for measurements that fall within the correct time interval. Performing the search here reduces computation later on.

The main body of the EKF is a while loop that will continue until the current state has been propagated to the proper time and all the measurements have been processed. The while loop also has an infinite loop check that will force an exit after 500 iterations.

The EKF has the ability to move measurement innovations in time using the state transition matrix. This feature reduces computation time because the filter can process measurements that are just a small fraction of a second apart without propagating the state over small fractions of a second. This is less accurate than propagating the state directly, but is sufficiently accurate over small time periods. Processes [C] and [D] (in Figure 3) calculate this interval and update the state with the measurements found in this interval. Processes [E] and [F] propagate the full state and covariance using a high fidelity propagator.

Update State with Measurements The EKF can process several different measurement types, but the high level logic for how they are processed in the EKF is the same. The measurement models ($h(\mathbf{x})$ in Equations (1), (4), and (5)) can be position and velocity, GPS pseudorange, or relative range.

Given the following measurement model

$$\tilde{y} = h(\tilde{\mathbf{x}}) + \tilde{\eta} \tag{1}$$

with measurement noise ($\tilde{\eta}$) with covariance

$$R = E[\tilde{\eta}_i \tilde{\eta}_j] \tag{2}$$

It is desired to update a state and state covariance ($\hat{\mathbf{x}}$, and P) at time i with measurements that take place at times j and k, which occur between times i and $i+1$ (as seen in Figure 4).

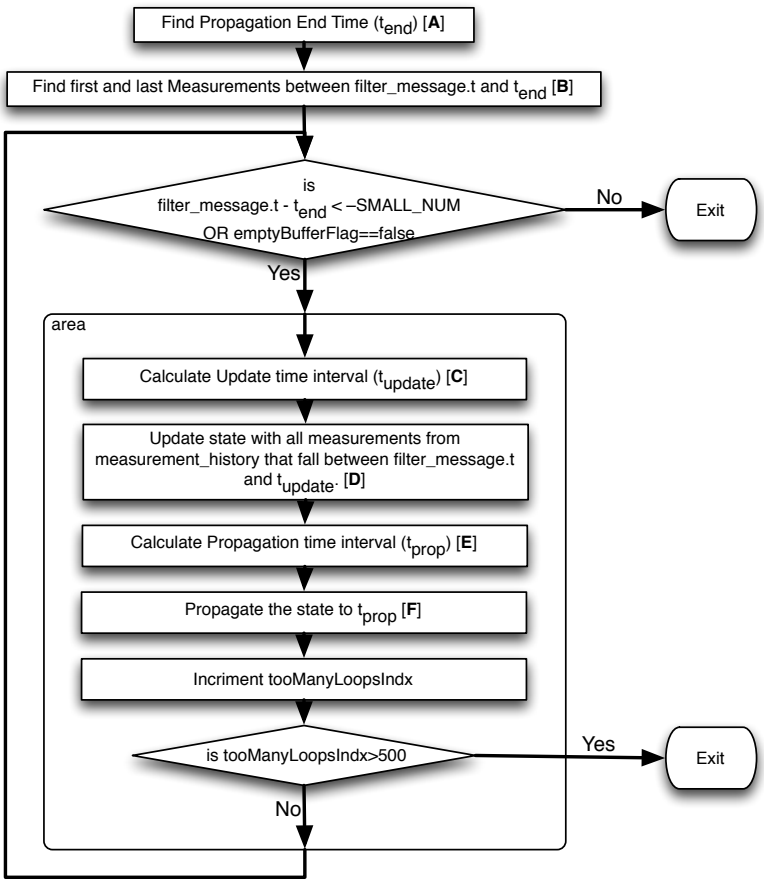

Figure 3. Flow chart of the extended Kalman filters operation.

The following algorithm (an extension of concepts presented in Schutz[10]) will be performed assuming that $\hat{\mathbf{x}}_i^-$ is the uncorrected state at time i. H_j will always be calculated with $\hat{\mathbf{x}}_i^-$ (even when processing multiple scalar measurements) while $\hat{y}_j$, the predicted measurement, will be calculated with $\hat{\mathbf{x}}_i^- + \Delta\hat{\mathbf{x}}_i$. This will eliminate the effect of measurement ordering on the filter performance. $\Delta\hat{\mathbf{x}}_i$ is the cumulative state innovation and is initially 0. The following algorithm also permits consider parameters to be part of the state vector. Consider parameters will not be updated but their value and statistics are accounted for during calculation. Thus the Kalman gain calculated in equation 8 is zero for consider parameters, and potentially non-zero for states.

1. Calculate the state transition matrix from time i to j (and vise-versa).

$$\begin{aligned} F_i &= \left.\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_i^-} \\ \Phi_{i\to j} &= e^{F_i(t_j - t_i)} \\ \Phi_{j\to i} &= e^{F_i(t_i - t_j)} \end{aligned} \tag{3}$$

2. Calculate the predicted measurement at time j with

$$\begin{aligned} \hat{y}_j &= h(\hat{\mathbf{x}}_j^+) \\ \text{where } \hat{\mathbf{x}}_j^+ &= \Phi_{i\to j}(\hat{\mathbf{x}}_i^- + \Delta\hat{\mathbf{x}}_i) \end{aligned} \tag{4}$$

3. Calculate the measurement sensitivity matrix at time j with

$$\begin{aligned} H_j &= \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_j^-} \\ \text{where } \hat{\mathbf{x}}_j^- &= \Phi_{i\to j}(\hat{\mathbf{x}}_i^-) \end{aligned} \tag{5}$$

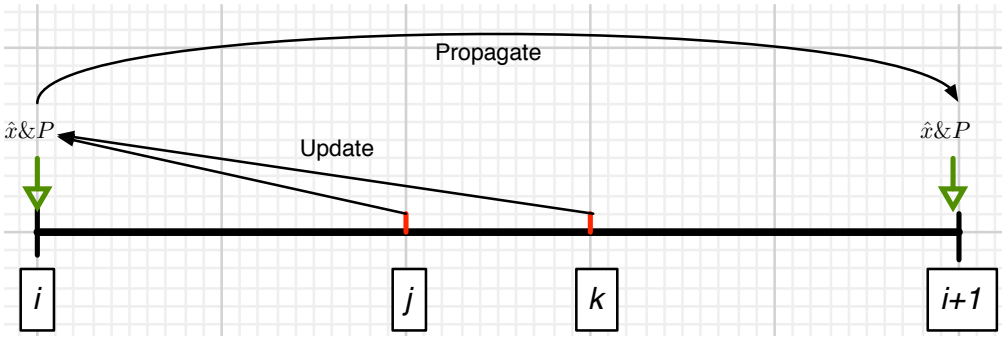

Figure 4. Timeline showing how measurements between i and i+1 are applied at time i.

4. Propagate the most current state covariance from time i to j assuming that process noise is negligible because the stepsize is very small.
$$P_j = \Phi_{i\to j} P_i \Phi_{i\to j}^{\mathrm{T}} \tag{6}$$
5. Update the cumulative state innovation at time i for states that are not consider parameters with
$$\Delta\hat{\mathbf{x}}_i = \Delta\hat{\mathbf{x}}_i + \Phi_{j\to i} K_j (\tilde{y}_j - \hat{y}_j) \tag{7}$$
$$\text{where } K_j\{m\} = \begin{cases} P_j\{m,:\} H_j^{\mathrm{T}} (H_j P_j H_j^{\mathrm{T}} + R_j)^{-1} & \text{if } \hat{\mathbf{x}}\{m\} \text{ is a state;} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$
for each element, m, of the state vector.
6. Update the state covariance estimate at time i with
$$P_i = \Phi_{j\to i} \left\{ (I - K_j H_j) P_j (I - K_j H_j)^{\mathrm{T}} + K_j R_j K_j^{\mathrm{T}} \right\} \Phi_{j\to i}^{\mathrm{T}} \tag{9}$$
7. In order to process measurement k, return to step 1 and replace j with k. Repeat for any other measurements between time i and $i+1$.
8. Update the uncorrected state ($\hat{\mathbf{x}}_{i0}$) with the cumulative update of all the measurements
$$\hat{\mathbf{x}}_{i0}^{+} = \hat{\mathbf{x}}_i^{-} + \Delta\hat{\mathbf{x}}_i \tag{10}$$

Note that if $i = j$, then $\Phi = I$ the identity matrix, and the update equations simplify to the basic Joseph form update algorithm for an extended Kalman filter.

Propagation General Overview Given the following true state ($\tilde{\mathbf{x}}$) model with process noise

$$\dot{\tilde{\mathbf{x}}} = \mathbf{f}(\tilde{\mathbf{x}}) + \tilde{\boldsymbol{\omega}} \tag{11}$$

The state estimate ($\hat{\mathbf{x}}$) is propagated by integrating the dynamics from time t_i to t_{i+1}. See Section II.B for more information about the various dynamic models. The dynamics may include maneuvers that have been represented as constant accelerations over the integration period.

$$\hat{\mathbf{x}}_{i+1} = \int_{t_i}^{t_{i+1}} \mathbf{f}(\hat{\mathbf{x}}_i) dt \tag{12}$$

The state covariance (P) is propagated by linearizing the dynamics about the state estimate at time t_i, calculating the state transition matrix, and adding in uncertainty due to process noise.

$$\begin{aligned} F_i &= \left. \frac{\partial \mathbf{f}(\mathbf{x}_i)}{\partial \mathbf{x}_i} \right|_{x=\hat{x}} \\ \Phi_{i\to i+1} &= e^{F_i (t_{i+1} - t_i)} \\ P_{i+1} &= \Phi_{i\to i+1} P_i \Phi_{i\to i+1}^{\mathrm{T}} + Q_i (t_{i+1} - t_i) \end{aligned} \tag{13}$$

The discrete process noise strength (Q) is the covariance of the zero-mean, white Gaussian process noise ($\tilde{\boldsymbol{\omega}}$)

$$Q_i \delta(t_i - t_j) = E[\tilde{\boldsymbol{\omega}}_i \tilde{\boldsymbol{\omega}}_j^{\mathrm{T}}] \tag{14}$$

where $\delta(t_i - t_j)$ is the Dirac delta function.

II.B. Dynamic Models

The state dynamics function for position and velocity is given by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{a}_g \end{bmatrix}$$

where $\mathbf{a}_g$ described in section II.B.1 is the acceleration due to gravity. The partial derivative of the dynamics function with respect to the position and velocity states F is calculated as follows.

$$F = \begin{bmatrix} \mathbf{0}_{3\times3} & I_{3\times3} \\ \frac{\partial \mathbf{a}_g}{\partial \mathbf{r}} & \mathbf{0}_{3\times3} \end{bmatrix}$$

where $\frac{\partial \mathbf{a}_g}{\partial \mathbf{r}}$ described in section II.B.1 is the partial due to the gravity acceleration.

The dynamics function for the clock states is given by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{b} \\ \dot{d} \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix}$$

The partial of the dynamics function for the clock states is given by

$$F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

II.B.1. Gravity

A gravity model (which uses the JGM-3 gravity coefficients) described by Montenbruck[11] was used to compute analytic solutions for both the acceleration due to gravity and its partial derivative $\frac{\partial \mathbf{a}_g}{\partial \mathbf{r}}$. The Earth rotation matrices that are required to convert between ICRF and ITRF were computed using an adaptation of the sofa[a] algorithms.

II.B.2. Unmodeled RIC Acceleration

A radial, in-track, cross-track acceleration can be included in the state vector to act as a catch-all for unmodeled dynamics. This can be particularly useful if a large uncertainty exists in the ballistic coefficient or the atmospheric density. For this to be effective the magnitude of the mismodeling error must be significant in comparison to the measurement noise. The new state vector augmented by three new unmodeled accelerations in the RIC frame is

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ \mathbf{a}_u^{RIC} \end{bmatrix}$$

where $\mathbf{a}_u^{RIC} = \begin{bmatrix} \ddot{x}_u^{RIC} & \ddot{y}_u^{RIC} & \ddot{z}_u^{RIC} \end{bmatrix}^{\mathrm{T}}$. The state dynamics function for the position, velocity, and unmodeled acceleration in RIC states is given by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{a}}_u^{RIC} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \ddot{\mathbf{r}}_g + T_{RIC \to ICRF}\, \mathbf{a}_u^{RIC} \\ -\mathbf{a}_u^{RIC} / \boldsymbol{\tau}_u^{RIC} \end{bmatrix}$$

[a] `http://www.iausofa.org`

where $\boldsymbol{\tau}_u^{RIC}$ is a vector of time constants associated with each element of $\mathbf{a}_u^{RIC}$. The transformation matrix from the RIC frame to the ICRF frame is calculated with

$$\begin{aligned} T_{RIC\to ICRF} &= \begin{bmatrix} \mathbf{i_r} & \mathbf{i_{r\times v}} \times \mathbf{i_r} & \mathbf{i_{r\times v}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\mathbf{r}}{|\mathbf{r}|} & \frac{\mathbf{r}\times\mathbf{v}}{|\mathbf{r}\times\mathbf{v}|} \times \frac{\mathbf{r}}{|\mathbf{r}|} & \frac{\mathbf{r}\times\mathbf{v}}{|\mathbf{r}\times\mathbf{v}|} \end{bmatrix} \end{aligned}$$

The partial of the dynamics function with respect to the position and velocity states F is calculated as follows.

$$F = \begin{bmatrix} \mathbf{0}_{3\times3} & I_{3\times3} & \mathbf{0}_{3\times3} \\ \frac{\partial \mathbf{a}_g}{\partial \mathbf{r}} + \left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{r}}\right] & \left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{v}}\right] & \left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{a}_u^{RIC}}\right] \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\frac{1}{\boldsymbol{\tau}_u^{RIC}} I_{3\times3} \end{bmatrix}$$

where $\left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{r}}\right]$, $\left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{v}}\right]$, and $\left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{a}_u^{RIC}}\right]$ are partials of unmodeled acceleration due to position, velocity and the unmodeled acceleration itself. $\left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{a}_u^{RIC}}\right]$ is defined below, and the other two partial are in the following two sections.

$$\left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{a}_u^{RIC}}\right] = T_{RIC\to ICRF}$$

Partial of Unmodeled Acceleration Due to Position

$$\left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{r}}\right] = \ddot{x}_u^{RIC}\left[\frac{\partial \mathbf{i_r}}{\partial \mathbf{r}}\right] + \ddot{y}_u^{RIC}\left[\frac{\partial\left(\mathbf{i_{r\times v}} \times \mathbf{i_r}\right)}{\partial \mathbf{r}}\right] + \ddot{z}_u^{RIC}\left[\frac{\partial \mathbf{i_{r\times v}}}{\partial \mathbf{r}}\right]$$

where

$$\begin{aligned} \left[\frac{\partial \mathbf{i_r}}{\partial \mathbf{r}}\right] &= \frac{1}{|\mathbf{r}|}\left[I_{3x3} - \mathbf{i_r}\mathbf{i_r}^{\mathrm{T}}\right] \\ \left[\frac{\partial \mathbf{i_{r\times v}}}{\partial \mathbf{r}}\right] &= \left(\frac{1}{|\mathbf{r}\times\mathbf{v}|}\left[I_{3x3} - \mathbf{i_{r\times v}}\mathbf{i_{r\times v}}^{\mathrm{T}}\right]\right)(-[\mathbf{v}]_\times) \\ \left[\frac{\partial\left(\mathbf{i_{r\times v}} \times \mathbf{i_r}\right)}{\partial \mathbf{r}}\right] &= \left[\mathbf{i_{r\times v}}\right]_\times\left[\frac{\partial \mathbf{i}_r}{\partial \mathbf{r}}\right] - \left[\mathbf{i_r}\right]_\times\left[\frac{\partial \mathbf{i_{r\times v}}}{\partial \mathbf{r}}\right] \\ &= \left[\mathbf{i_{r\times v}}\right]_\times\left(\frac{1}{|\mathbf{r}|}\left[I_{3x3} - \mathbf{i_r}\mathbf{i_r}^{\mathrm{T}}\right]\right) + \left[\mathbf{i_r}\right]_\times\left(\frac{1}{|\mathbf{r}\times\mathbf{v}|}\left[I_{3x3} - \mathbf{i_{r\times v}}\mathbf{i_{r\times v}}^{\mathrm{T}}\right]\right)[\mathbf{v}]_\times \end{aligned}$$

Partial of Unmodeled Accelerations Due to Velocity

$$\left[\frac{\partial\left(T_{RIC\to ICRF}\,\mathbf{a}_u^{RIC}\right)}{\partial \mathbf{v}}\right] = \ddot{x}_u^{RIC}\left[\frac{\partial \mathbf{i_r}}{\partial \mathbf{v}}\right] + \ddot{y}_u^{RIC}\left[\frac{\partial\left(\mathbf{i_{r\times v}} \times \mathbf{i_r}\right)}{\partial \mathbf{v}}\right] + \ddot{z}_u^{RIC}\left[\frac{\partial \mathbf{i_{r\times v}}}{\partial \mathbf{v}}\right]$$

where

$$\begin{aligned} \left[\frac{\partial \mathbf{i_r}}{\partial \mathbf{v}}\right] &= \mathbf{0}_{3\times3} \\ \left[\frac{\partial \mathbf{i_{r\times v}}}{\partial \mathbf{v}}\right] &= \left(\frac{1}{|\mathbf{r}\times\mathbf{v}|}\left[I_{3x3} - \mathbf{i_{r\times v}}\mathbf{i_{r\times v}}^{\mathrm{T}}\right]\right)[\mathbf{r}]_\times \\ \left[\frac{\partial\left(\mathbf{i_{r\times v}} \times \mathbf{i}_r\right)}{\partial \mathbf{v}}\right] &= -\left[\mathbf{i_r}\right]_\times\left[\frac{\partial \mathbf{i_{r\times v}}}{\partial \mathbf{v}}\right] \\ &= -\left[\mathbf{i_r}\right]_\times\left(\frac{1}{|\mathbf{r}\times\mathbf{v}|}\left[I_{3x3} - \mathbf{i_{r\times v}}\mathbf{i_{r\times v}}^{\mathrm{T}}\right]\right)[\mathbf{r}]_\times \end{aligned}$$

II.C. GPS Filter Operation

When the NavFilter is configured to process pseudorange measurements, the pseudorange measurement model in the EKF will be exercised. The `collate_gps_ephem`, `collate_gps_in_view`, `select_common_gps` functions are used as well . Typically, the filter state will include position, velocity, and clock bias and drift, for a total of eight states. Unmodeled accelerations in the radial, intrack, crosstrack (RIC) frame can also be included if desired.

II.C.1. GPS Pseudorange Measurement Model

As seen in Figure 5, the GPS Pseudorange measurement model has several steps. First, the position of the GPS satellite in the International Celestial Reference Frame (ICRF) at reception time is calculated by processing the GPS ephemeris data. Then the flight time of the signal and the position of the GPS satellite at broadcast time is iteratively determined. The predicted measurement will be the magnitude of the difference between the estimated position of the GPS receiver at reception time and the position of the GPS satellite at broadcast time plus clock errors. This pseudorange measurement model is standard practice and follows algorithms detailed in Montenbruck.[11] In order to reduce the impact of Earth orientation errors on the measurement prediction ($\hat{y}$) for a module at LEO, $\hat{y}$ is calculated in the International Terrestrial Reference Frame (ITRF).

$$\hat{y} = \left| r_{gps}^{ITRF} - T_{ICRF \to ITRF}\, r_{receiver}^{ICRF} \right| + b_{receiver} - b_{gps} \tag{15}$$

The measurement sensitivity matrix (H) or the partial of the measurement with respect to the states must be calculated in the ICRF frame. The pertinent filter states are the position,velocity, bias and drift of the receiver. Thus, H is defined as

$$H = \left[\begin{array}{cccc} \frac{\partial \hat{y}}{\left(\partial r_{receiver}^{ICRF}\right)^{\mathrm{T}}} & \frac{\partial \hat{y}}{\left(\partial v_{receiver}^{ICRF}\right)^{\mathrm{T}}} & \frac{\partial \hat{y}}{\partial b_{receiver}} & \frac{\partial \hat{y}}{\partial d_{receiver}} \end{array} \right] \tag{16}$$

$$H = \left[\begin{array}{cccccc} -\frac{\left(r_{gps}^{ICRF} - r_{receiver}^{ICRF}\right)^{\mathrm{T}}}{\left\| r_{gps}^{ICRF} - r_{receiver}^{ICRF} \right\|} & 0 & 0 & 0 & 1 & 0 \end{array} \right] \tag{17}$$

II.C.2. Tracking Common GPS Satellites

In cluster flight the primary objective of the navigation algorithm is to achieve accurate relative position and velocity solutions. To achieve this when processing pseudorange measurements it is best to process measurements received from a common set of GPS satellites for the filters on each module. This will drastically reduce the relative nav errors because each filter will see nearly the same errors. It should be noted that processing a common set of GPS measurements will result in decreased absolute nav accuracy because fewer measurements will be processed. If absolute nav performance is the primary concern the following steps should not be taken.

There are several reasons that the modules in the cluster may not be tracking a common set of GPS satellites. Among them are different antenna configurations, and different orientations of the modules. Tracking of different GPS satellites can also be caused be atmospheric disruptions. As such, applying a mask angle to measurements can help reduce the differences between the modules. However, listing the satellites in view for each of the modules and manually eliminating those that differ is the only way to effectively eliminate the problem.

To solve this, an algorithm that keeps track of all the GPS satellites that are in view by each of the modules was implemented. It works by obtaining from each module a list of the GPS satellites that it has received pseudorange measurements from over a certain time frame. A separate list is stored by each module. Due to the distributed nature of the system small discrepancies can appear momentarily between the lists, however the effect of these variations is negligible.

An algorithm was implemented that matches up all of the GPS satellites that are commonly tracked by all modules. If the number in common is less than some limit, then the method attempts to match up as many as possible on all but one of the modules. The module that is left out simply processes all of the measurements available to it. If the number of GPS satellites that are in common amongst the remaining group falls bellow some limit then this process repeats again. The module in the group with the fewest number of matching GPS satellites is excluded from the group, and the matching satellites amongst the

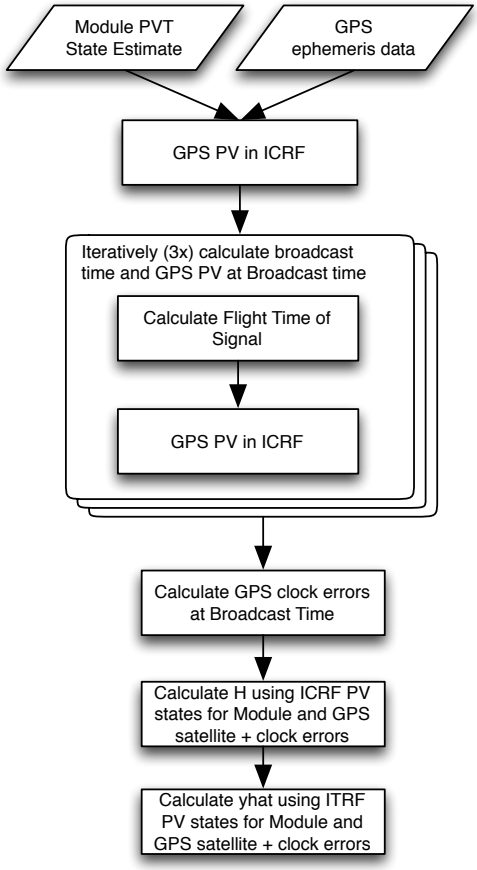

Figure 5. Flow chart of the GPS pseudorange measurement model.

remaining modules are listed. This process continues to repeat until an acceptable set of GPS satellites is obtained or a single module remains in the group. Due to processor constraints, an optimal solution is not always obtained for an arbitrary cluster size. For cluster sizes up to four modules the optimal solution is always obtained from the flight software algorithm, but for more than four modules the optimal solution is not guaranteed. This was tested by comparing the optimal solution against the flight software solution for a large set of random cases. For cluster sizes larger than four module the two solutions agreed for all but a small percentage of the cases.

II.D. Relative Range Filter Operation

When the NavFilter is configured to process relative range measurements, the `process_nav_solution` function and possibly the position and velocity measurement models will be exercised. Typically, the filter state will include the position and velocity of a "primary" module as well as a relative position and velocity of the local module with respect to the primary. The filter can also contain position and velocity states from up to two other modules ("secondaries"). Thus, the local module will have the ability to process relative range measurements for up to three other modules. Unmodeled accelerations in the RIC frame can also be included for each module if desired.

II.D.1. Processing NavSolutions from other Modules

The primary and secondary states are updated with the NavSolution published by the navigation component on those modules. If these states are consider parameters, then the NavSolution position, velocity and covariance directly replace the values previously in the NavFilter. Off diagonal correlations with other module states are all set to zero. If the NavSolution data must be propagated to match the current NavFilter state time, the process_nav_solution function propagates the NavSolution data.

If the primary and secondary states are not consider parameters, the NavSolution data is converted into position and velocity measurements with measurement covariances obtained from the diagonal of the NavSolution covariance.

II.D.2. Position and Velocity Measurement Model

Position and Velocity Measurements models are simply the current inertial state where H is unity for the observed states. These position and velocity measurements are obtained from the output of other filters.

II.D.3. Relative Range Measurement Model

The relative range measurement model is simply the magnitude of the difference in the estimated positions of the origin and target satellite modules.

$$\hat{y} = \rho = |r_{target} - r_{origin}| \tag{18}$$

The partial of the measurement with respect to the target inertial or relative position state is

$$\frac{\partial \hat{y}}{\partial r_{target}} = \frac{\rho^{\mathrm{T}}}{|\rho|} \tag{19}$$

The partial of the measurement with respect to the origin inertial or relative position state is

$$\frac{\partial \hat{y}}{\partial r_{origin}} = -\frac{\rho^{\mathrm{T}}}{|\rho|} \tag{20}$$

Though not detailed here, these basic equations are modified as needed if relative position states are involved in the calculation.

II.E. Fault Detection, Isolation and Recovery

Fault detection, isolation and recovery (FDIR) is tightly woven into the entire navigation architecture. The primary data source for fault detection is measurements. Missing measurements or unexpectedly large measurement residuals are highly indicative of faults. To facilitate the processing of this data, all measurements are tracked from the point they enter the filter to the end of their digital life. Sanity checks are performed at different stages. If a measurement is processed in the filter, rejected or ignored for any reason, its fate is noted. Statistics are gathered on measurement populations and compared to acceptable bounds. If these configurable bounds are violated, the health status of the filter will be downgraded. The NavManager evaluates the health and priority of each filter and publishes the "best" result along with a health status. Relative range filters will ignore relative range measurements to modules that have degraded health status. The details of this FDIR architecture are in the sections below.

II.E.1. Sanity tests and residual masking performed in `collate_measurements` *and* `ekf`

Measurements are sent to the `collate_measurements` function which performs preliminary sanity checks on the data and populates the measurement buffer with valid measurements. Invalid measurements are sent directly to the `monitor_measurements` function. The EKF ensures that all the data required for processing is available (e.g. GPS ephemeris) and rejects the measurement if anything is missing. Finally, before updating the state, the magnitude of the residual is compared to the expected statistics. If the normalized residual, q, is larger than a configurable bound, the measurement is rejected. The residual, ν, is normalized by the residual covariance, S, as such

$$q = \nu S^{-1} \nu \tag{21}$$

where ν is given by

$$\nu = y - H\hat{\mathrm{x}} \tag{22}$$

and

$$S = HPH^{\mathrm{T}} + R \tag{23}$$

II.E.2. Data gathered and evaluated in `monitor_measurements` *and* `monitor_filter`

All measurements eventually find their way into the `monitor_measurements` function. This function synthesizes all "end-of-life" measurements into bins distinguished by type, origin, target, and history. The history is a record of the measurement's final fate: if and where it was rejected, ignored, or processed. If a pre-update residual was calculated for the measurements in a bin, then the normed innovation square (NIS),[12] mean, and variance of the measurements in that bin are calculated. The bins only contain statistical information, not the actual measurements, and are updated as new measurements arrive. The total number of measurement summed across each bin is $n_p = \sum_{k=1}^{N} n_k$, where N is the number of bins. The mean of the normed innovation for a bin can be updated by combining the mean of the new measurements with the previous mean to obtain a "pooled" mean, μ_p, given by

$$\mu_p = \frac{1}{n_p} \sum_{k=1}^{N} n_k \mu_k \tag{24}$$

where n_k is the number of samples in each bin (or the current number and new number) and μ_k is the mean of the normalized innovations for each bin. The pooled variance, σ_p^2, can be found by summing the variance of the means and the mean of the variances

$$\sigma_p^2 = \frac{1}{n_p - 1} \left(\sum_{k=1}^{N} n_k \left(u_k - \mu_p \right)^2 + \sum_{k=1}^{N} n_k \sigma_k^2 \right) \tag{25}$$

If the statistics are pooled multiple times (as is the case when bins collect measurements over multiple calls to EKF), care must be taken to ensure that only final pooled variance is normalized by $n_p - 1$, else the variance will be artificially inflated.

The norm innovation squared (NIS) is obtained by simply summing the NIS values from each bin. The NIS value is evaluated using a χ^2-test. For example, if 10 measurements are taken and the confidence level of the test is 0.99, the NIS limit is 23.21. For any NIS value greater than 23.21 the test would fail.

Tests are also performed that check if a sufficient number of measurements are received and if the percentage of measurements that are rejected is below a set bound. In addition to the tests that are performed on the measurements, additional sanity checks are performed on the state and covariance. For instance, the covariance is checked that it is positive definite and that the diagonal elements are below a set bound.

The filter estimate is outputted at regular interval by each NavFilter within NavManager. Right before the state is outputted, the `monitor_filter` function processes the binned measurement statistics and the filter estimate and performs a suite of tests that are useful in determining the health of the filter. The results of these tests are outputted along with the filter estimate. The measurement bins are reset after being evaluated.

II.E.3. Filter selection and publication logic in `select_nav_solution` *and* `publish_nav_solution`

At the NavManager level, the `select_nav_solution` function weighs the test results from each filter to determine its health status. The available health quality statuses are GOOD, DEFICIENT, POOR, and BAD. As configured for this paper, GOOD means the filter is healthy, DEFICIENT means the covariances in the filter are larger than a configurable bound, POOR means there is reason to believe the covariance may be inaccurate (e.g. too many rejected measurements), and BAD means the state or covariance is not reasonable (position inside earth, non-symmetric covariance, etc). The determination of these status are very flexible. Each non-good, health quality status (BAD, POOR, DEFICIENT) has an associated vector of configurable weights and an acceptable limit that is totally configurable by the operator.

For example, given four tests where the first and second tests failed and BAD weights of [10 5 5 0], POOR weights of [0 10 10 0], and DEFICIENT weights of [0 0 0 10], the results would be BAD=15, POOR=10, and DEFICIENT = 0. If the configurable limits were all 10, then the filter would be rated BAD, because the worse status always takes precedence.

The health quality status along with a configurable priority vector, and some hysteresis logic all come into play to determine which filter solution (and associated health quality) will be published. For example, if the highest priority filter is POOR and the secondary filter is GOOD and the highest priority filter was

previously outputted, then the state and covariances of the two filters are compared. If they are significantly different, then the secondary filter will be published. If they are not significantly different, then the highest priority filter solution will be published with a GOOD health quality status. This logic prevents FDIR from superfluously switching filter solutions due to failed tests when the solutions are not significantly different.

Six tests are performed to determine if filters are significantly different. Position and velocity Mahalanobis distances with respect to position and velocity covariances from both filters and the maximum and minimum ratios between the position and velocity covariance diagonals are compared to configurable bounds. If Mahalanobis distance limits are set to 0 and the ratios are set to 1, then there will effectively be no hysteresis and any difference in health status will cause the outputted filter solution to be switched to the best one.

Mahalanobis distance[13] is defined as the number of standard deviations that exist between the mean and a vector endpoint and is calculated with:

$$d_{mahalanobis} = \sqrt{\mathbf{r}^{\mathrm{T}} P^{-1} \mathbf{r}}$$

The only exception to this flow is when the primary filter status is BAD and the secondary has a superior health status. In this case the secondary filter solution will be published. If all the filters are BAD, and a PVT solution is available, then the NavManager will publish the PVT solution along with a health quality status set by configuration data.

This architecture allows the operator to tune how the navigation system autonomously responds to failed tests and to modify that response on the fly. If all weights are set to zero, they the ground can completely control which filter solution is published with the priority vector while still seeing the results of the tests on each filter through telemetry.

II.E.4. Responding to Navigation Quality in `process_nav_solution` and `collate_measurements`

The `process_nav_solution` function within each filter responds to degraded health statuses by flipping an "ignore" flag. The `collate_measurements` function will ignore measurements associated with modules that have this flag set to true. This helps navigation problems on one module from spreading to other modules in the cluster.

III. Testing and Results

In order to demonstrate the unique strengths of the navigation architecture, various sensor faults were introduced into a high fidelity simulation. The results show how the system detects, isolates and responds to the faults

III.A. Overview of Simulation Environment

To test the navigation software, Trick Simulation Environment 13[b] is used in conjunction with the JEOD[c] environment models. This provides a high-fidelity, independently developed simulation for testing the navigation algorithms. The degree and order of the spherical harmonic gravity model were set to 20. Drag is negligible at the simulated altitude of 600 km and solar radiation pressure was not simulated. The simulation also provides pseudorange and relative range measurements. Pseudorange errors are summarized in Table 1. The relative range errors are summarized in Table 2. The pseudorange measurements and relative range measurements are generated and sent the the filters at 0.1 Hz. The simulation environment is also capable of simulating various failure scenarios including GPS outages and artificially noisy pseudorange measurements.

[b] `http://ntrs.nasa.gov/search.jsp?R=20120010428`

[c] `http://www.nasa.gov/centers/johnson/techtransfer/technology/MSC-24556-1-jeod2.html`

Table 1. Summary of errors in true pseudorange measurement model.

Pseudorange Error Type	Description
Ionospheric Delay	40-50 meters
Tropospheric Delay	Not Applicable on orbit
GPS Receiver Tracking Loop Errors	Gaussian distribution noise, 1.0 meter standard deviation, zero mean
GPS Receiver Oscillator	Thermally Compensated Crystal Oscillator, ~ 1 ppm
GPS Satellite Ephemeris Errors	Not Simulated
GPS Satellite Clock Errors	Not Simulated
Multipath	Not Simulated

Table 2. Erros in true relative range measurement model.

Relative Range Error Type	Description
Quantization Error	$\pm 15\,m$ Triangular Distribution with 0 mean
Random Noise	Gaussian distribution noise, 0.551 meter standard deviation, zero mean

An open loop simulation was performed in order to evaluate the ability of the navigation system to detect GPS faults, switch to relative navigation, and then switch back to the GPS solution once the fault is gone. The cluster geometry is shown in Figure 7, which is a local vertical, curvilinear (LVC) representation of the modules relative motion with respect to Module 1 (the dot at [0,0,0]). Module 4 is denoted by the cyan trajectory on the left.

The initial conditions for these modules in the ICRF frame are shown in the table below. These states approximately correspond to a 600 km altitude, circular orbit at 63.43° inclination.

Module	1	2	3	4
r(1), m	6978137.98	6978137.98	6978136.84	6978136.84
r(2), m	-344.67	344.67	344.76	-344.76
r(3), m	-432.36	432.36	1459.84	-1459.84
v(1), m/s	0.71	-0.71	-1.21	1.21
v(2), m/s	3380.56	3380.56	3380.56	3380.56
v(3), m/s	6759.67	6759.67	6759.67	6759.67

III.B. Simulation Results

III.B.1. GPS Outage for One Orbit

The first GPS fault is a sudden absence of GPS pseudorange measurements on module 4 that begins one orbit into the simulation and ends two orbits into the simulation. As expected, the navigation architecture detected the fault and isolated it to the GPS filter because the number of pseudorange measurements dropped below the expected minimum value of 4, as seen in Figure 6, and responded by switching over to the relative range filter solution during the outage. The health of the outputted navigation solution and the filter source for each module are shown in Figure 8. As expected, the FDIR system switches to the relative range filter at approximately one orbit and then returns to the GPS filter at approximately two orbits. However, it does not switch instantly back to the GPS solution once pseudorange measurements return. There is a slight delay while the GPS filter converges and the health is changed from "deficient" to "good". Once the GPS filter health is good, the GPS filter solution is outputted.

The absolute navigation errors and filter state covariance (1σ) are shown in Figure 9. The large errors in position are a result of ionospheric delays on the pseudorange measurements. The magenta lines that start at one orbit represent the empirical mean and standard deviation of the errors after one orbit to the end of the simulation. Even though the absolute state errors are fairly large, the relative state errors are quite small, as seen in Figures 7 and 10. This is because each module is seeing the same ionospheric delays. It is also clear that the relative range filter is producing the navigation state for module 4 from approximately one to two orbits.

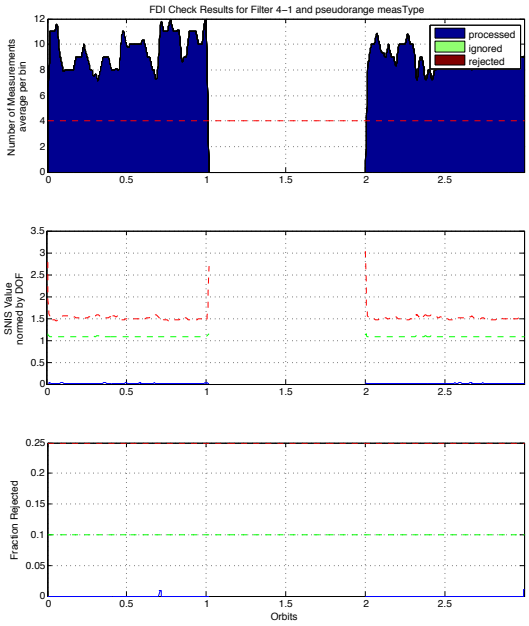

Figure 6. FDIR results for GPS failure simulation.

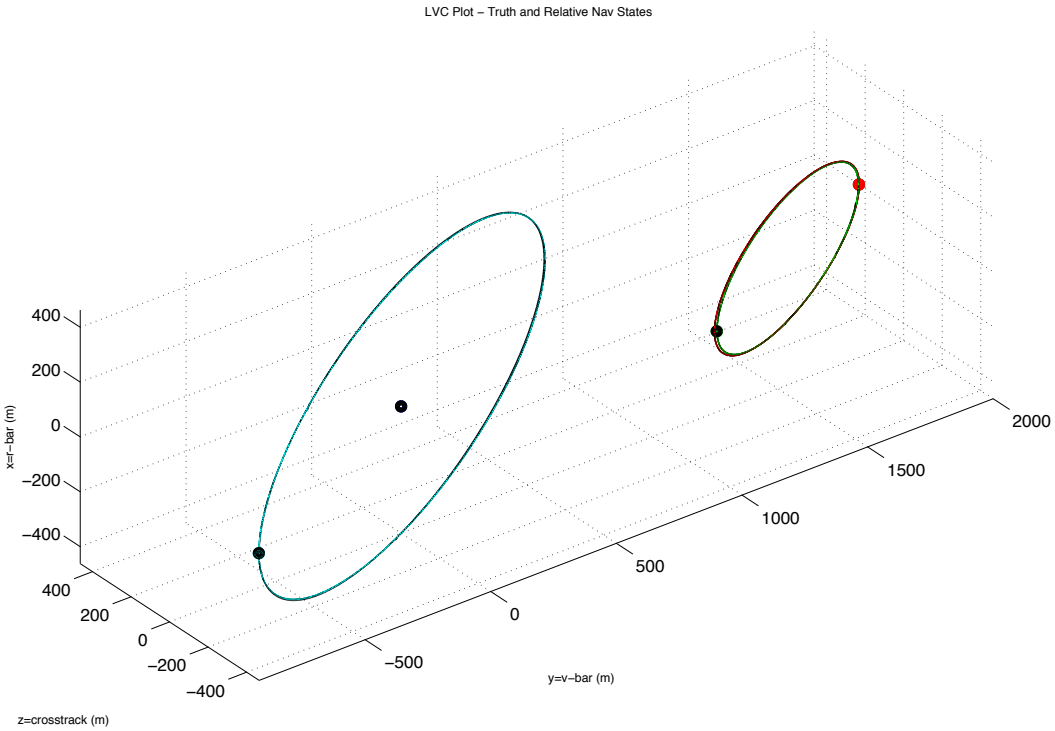

Figure 7. Cluster geometry for GPS failure simulation.

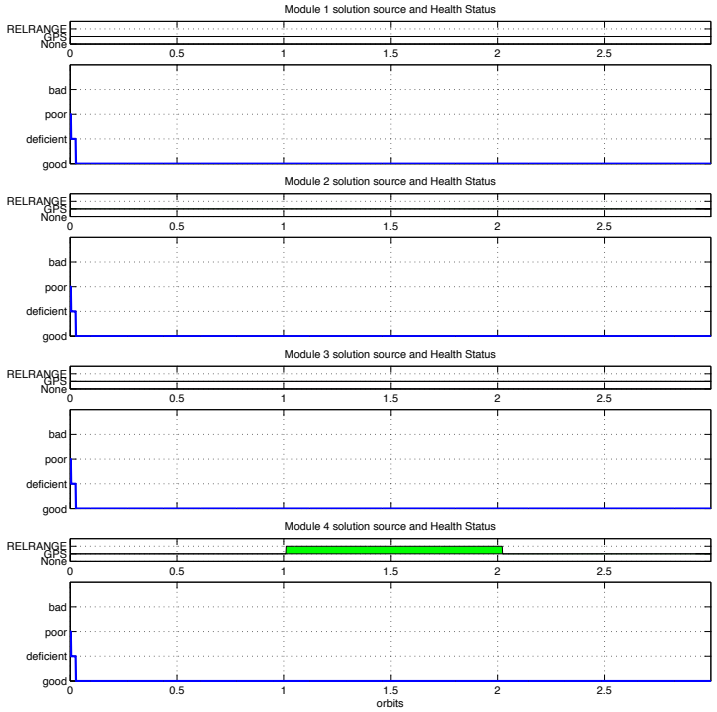

Figure 8. Navigation health and filter source for each module in the cluster for the GPS failure simulation.

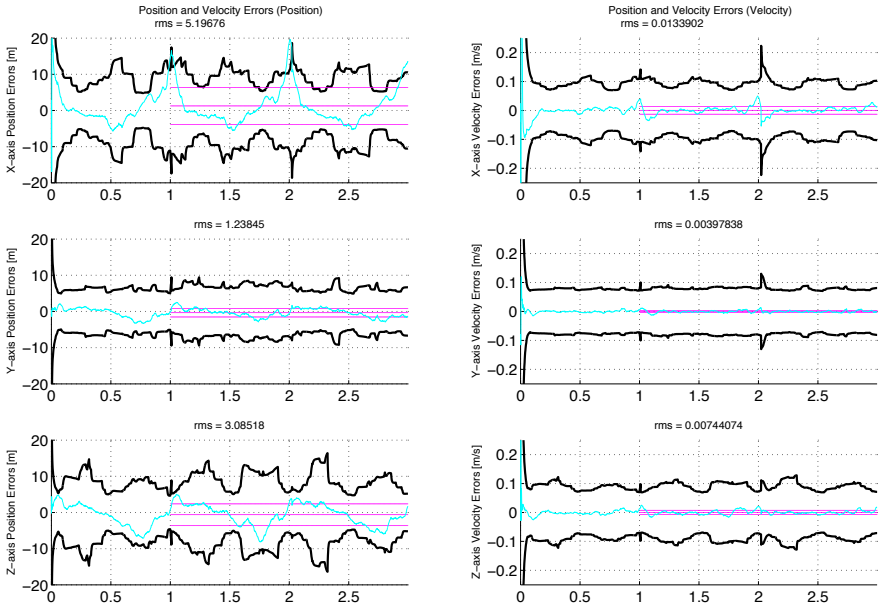

Figure 9. Absolute navigation errors and covariance for module 4 for the GPS failure simulation

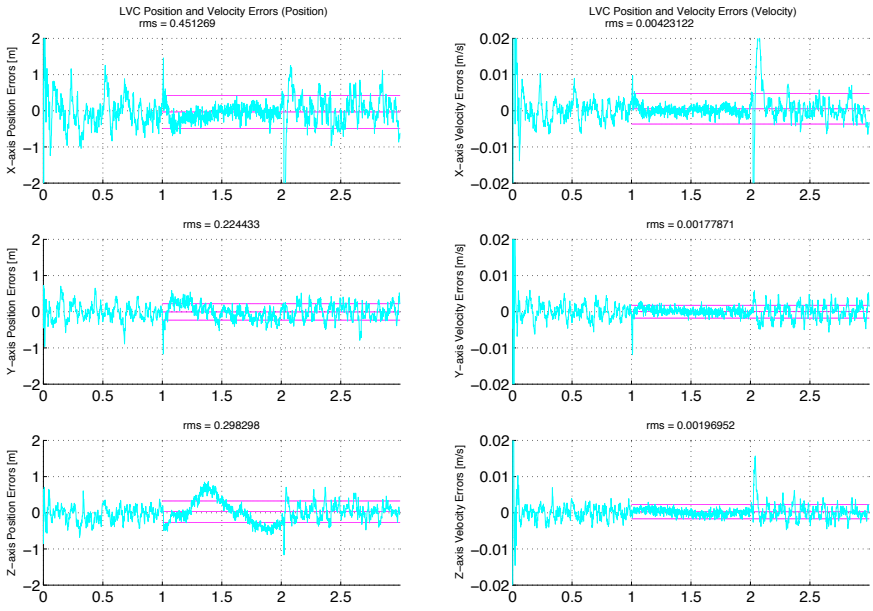

Figure 10. Relative navigation errors for module 4 with respect to module 1 for the GPS failure simulation.

III.B.2. Unexpectedly Noisy GPS Pseudorange Measurements for One Orbit

The second fault is unexpected noisy pseudorange measurements received by module 4, which occur during the second orbit. During this time the pseudorange measurements have an extra 50 m of uniform noise added to them. This noise level isn't enough for the FDIR algorithm, under the current setup, to start rejecting measurements. As can be seen in Figure 11, none of the measurements are rejected. However, it does cause the NIS value to increase to the point that the pseudorange filter's health degrades. As can be seen from Figure 12, the FDIR algorithm switches to the relrange filter during the period of noisy measurements, and then appropriately switches back once the measurements return to normal. Figure 13 shows small jumps in the relative navigation error as the solution transitions between the pseudorange and relrange filter solutions. This small jump can likely be reduced with further tuning of the FDIR parameters.

III.B.3. Unusable GPS Pseudorange Measurements for One Orbit

The third fault is also unexpected noisy pseudorange measurements received by module 4, which occur during the second orbit. Unlike the fault case in Section III.B.2, the pseudorange measurements have an extra 2000 m of uniform noise added to them. This noise level is enough for the FDIR algorithm, to start rejecting measurements. The majority of the measurements are rejected and, as shown in Figure 14, the limit for the fraction of rejected measurements is exceeded. Depending how the various tests are weighted, this could result in a further degradation of the health status. As can be seen from Figure 15, the FDIR algorithm switches to the relrange filter during the period of noisy measurements. Once the additional noise on the measurements goes away, it takes nearly an orbit for the solution to reconverge and for the navigation solution to switch back to the pseudorange filter (Figure 16). Again there is a jump in the relative navigation error as the solution source switches between the pseudorange and relative range filters.

III.B.4. GPS Outage without Relative Range Backup

The final failure to be considered is identical to the failure in Section III.B.1, except that no backup relative range filter exists on module 4. As in the previous scenario Module 4 stops receiving GPS measurements during the second orbit (Figure 17), however with no backup filter the navigation solution degrades to a POOR health state as seen in Figure 20. The relative range filters on the other modules stop ingesting the

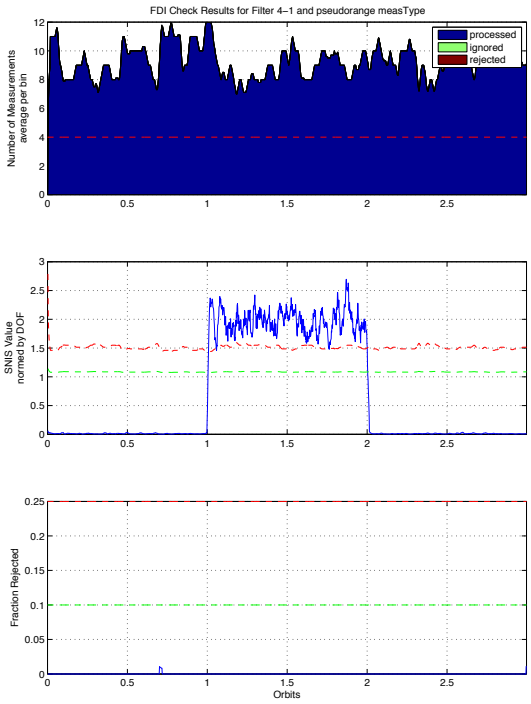

Figure 11. FDIR results for noisy pseudorange measurements simulation.

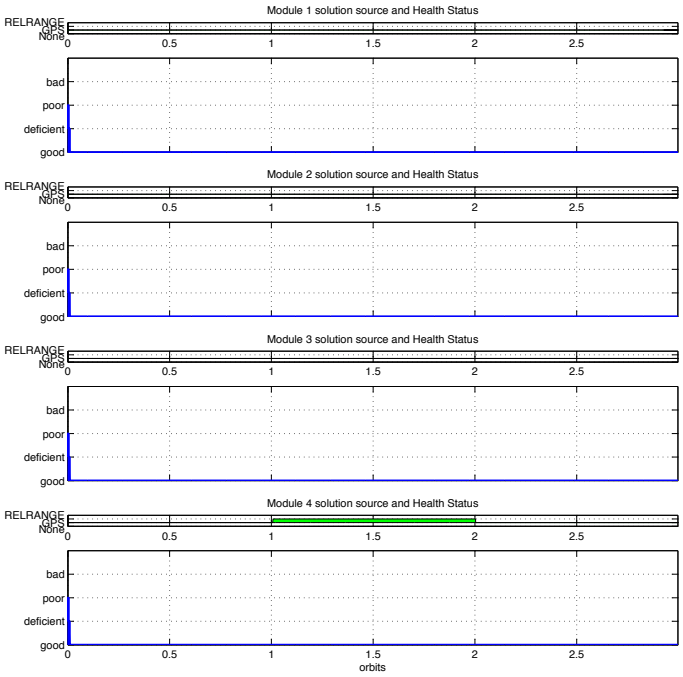

Figure 12. Navigation health and filter source for each module in the cluster for the noisy pseudorange measurements simulation.

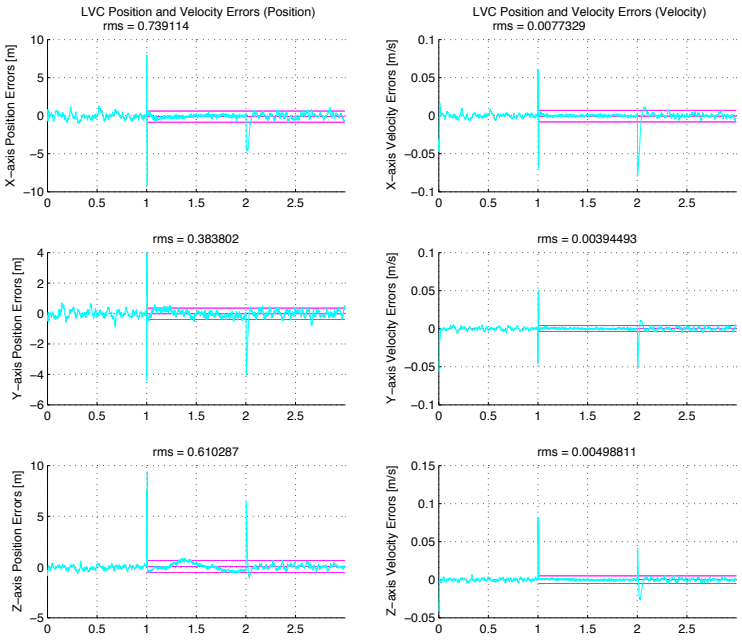

Figure 13. Relative navigation errors for module 4 with respect to module 1 for the noisy pseudorange measurements simulation.

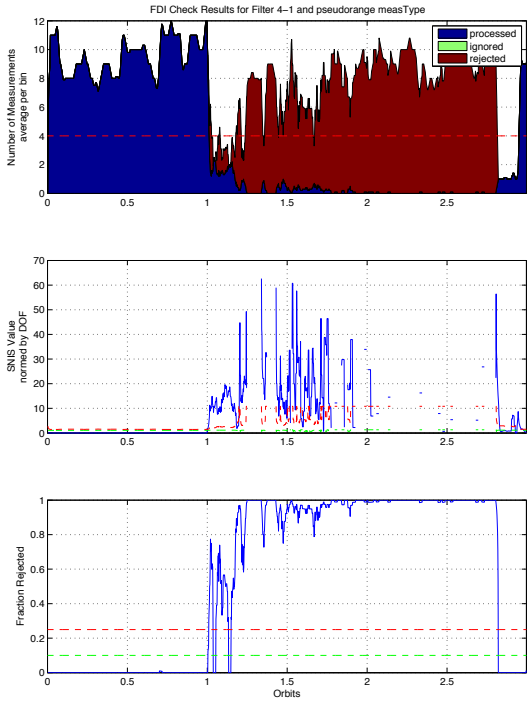

Figure 14. FDIR results for unusable pseudorange measurements simulation.

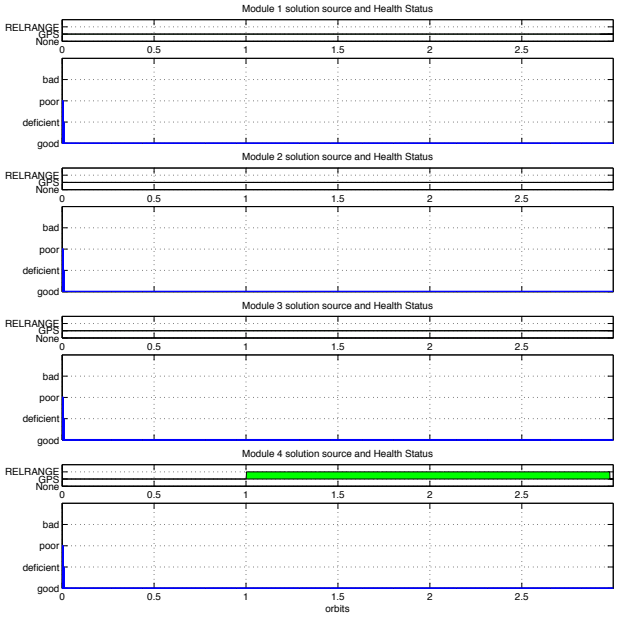

Figure 15. Navigation Health and filter source for each module in the cluster. Unusable Pseudorange Measurements Simulation

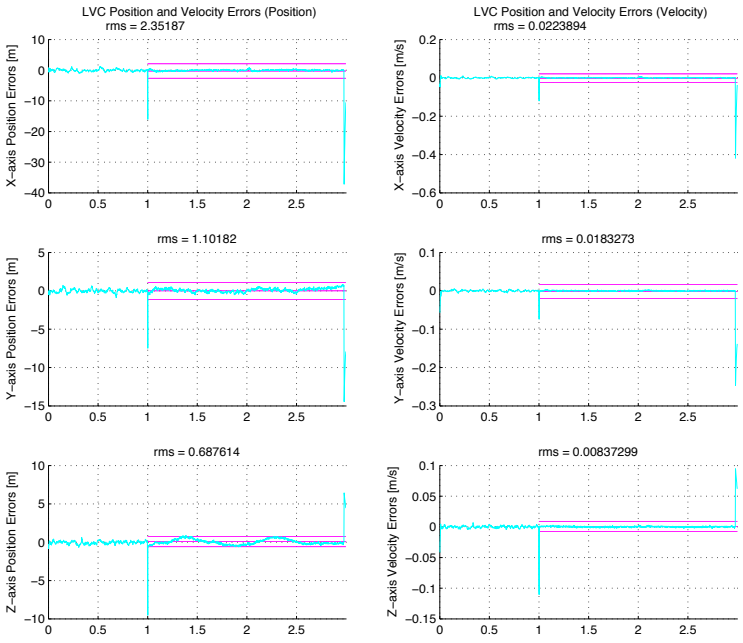

Figure 16. Relative Navigation Errors for Module 4 with respect to Module 1. Unusable Pseudorange Measurements Simulation.

navigation solution from module 4 once the health degrades to a POOR state. The solution on module 4, shown in Figures 19 and 22, during the GPS outage is obtained solely from propagation of the estimate. As can be seen in Figure 18 the relrange filters also begin ignoring measurements from the module with a POOR health. This automatic response reduces the odds that a bad filter will corrupt the other filters in the cluster. Figure 21 shows that the uncertainty continues to grow until GPS measurements are once again received, after which the errors return to an acceptable level.

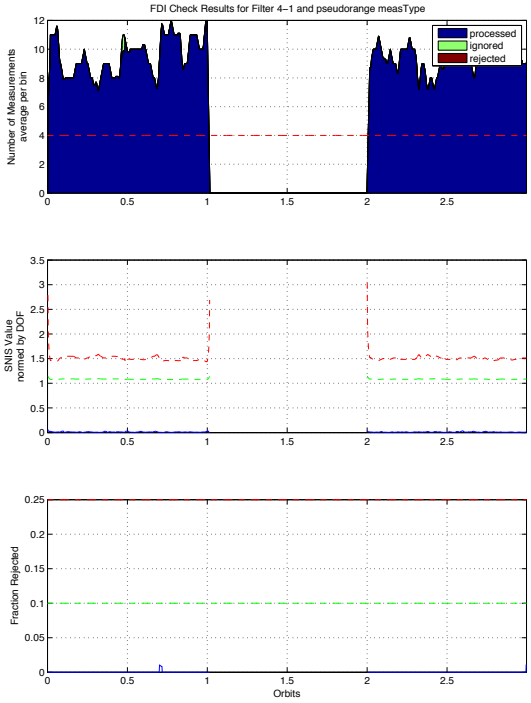

Figure 17. FDIR results for the GPS pseudorange filter on module 4 for the GPS failure without backup simulation.

IV. Conclusion

Emergent Space Technologies has developed a distributed, redundant navigation architecture for cluster flight. The architecture provides sufficient accuracy for the needs of clustered flight and is robust to sensor failures. Simulation results show that the multi-filter design allows uninterrupted navigation through a GPS failure with a relative navigation solution that, though less accurate and robust in comparison to GPS, is sufficiently accurate to maintain cluster flight.

The distributed nature of the architecture reduces the computational requirements for a given module, and supports the addition or removal of modules from the cluster without interruption. The navigation architecture is flexible in that a filter can easily be configured with any combination of states, and can be set up to process any of the supported measurement types for any of the chosen states. Additionally, the filter configurations and FDIR tuning can be reconfigured on the fly. Finally, by implementing an algorithm to track a common set of GPS satellites the architecture is capable of achieving acceptable relative navigation performance when processing GPS pseudorange measurements despite its distributed nature.

The design of the navigation architecture is a significant improvement on the state of the art and has room for future improvements, including the ability to process additional measurement types such as angles or range-rate.

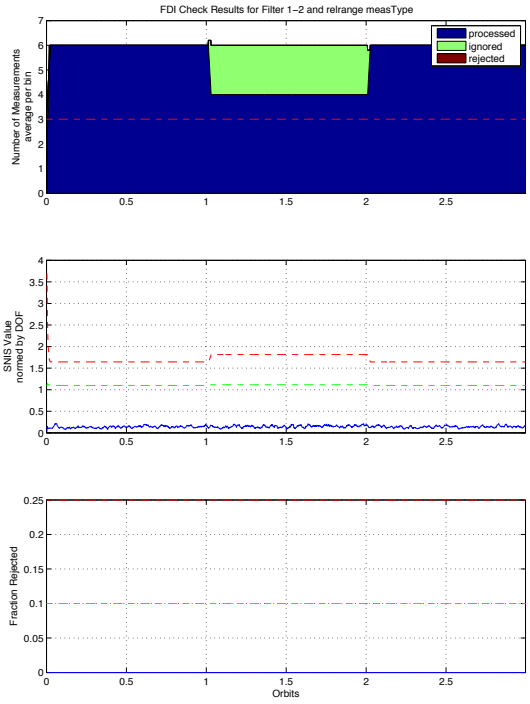

Figure 18. FDIR results for the GPS relrange filter on module 1 for the GPS failure without backup simulation.

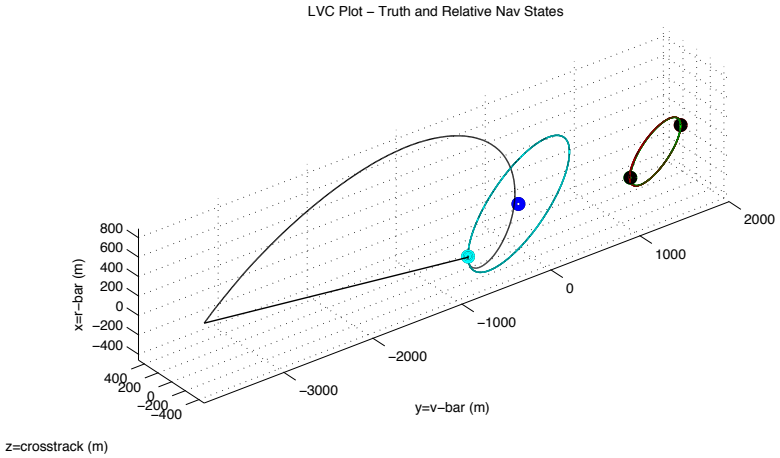

Figure 19. Cluster geometry and navigation solution for GPS failure without backup simulation.

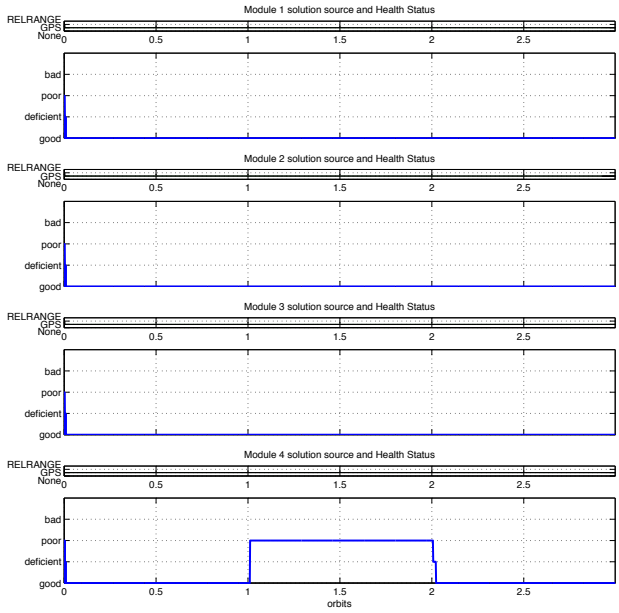

Figure 20. Navigation health and filter source for each module in the cluster for the GPS failure without backup simulation.

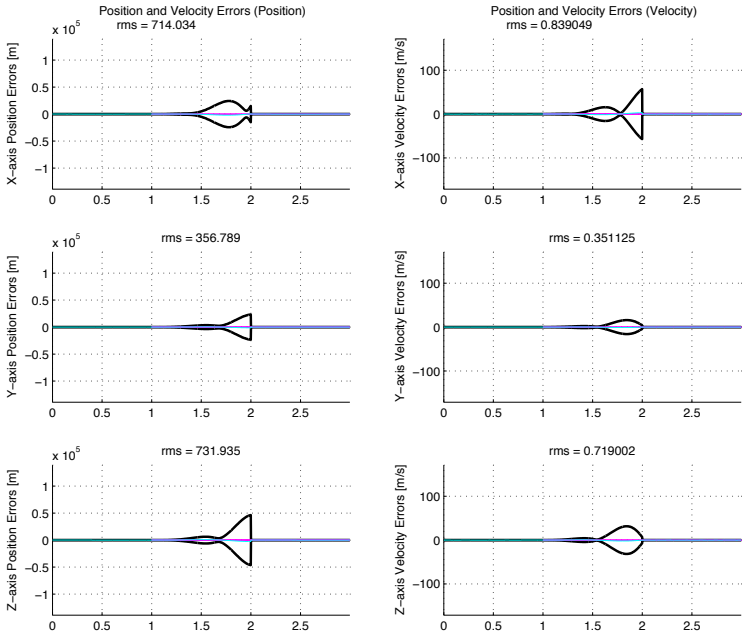

Figure 21. Absolute navigation errors and covariance for module 4 for the GPS failure without backup simulation.

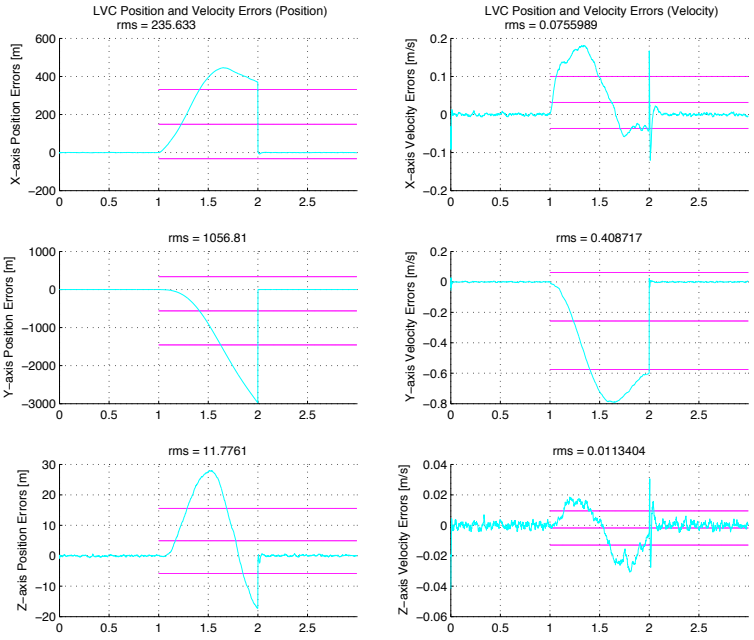

Figure 22. Relative navigation errors for module 4 with respect to module 1 for the GPS failure without backup simulation.

Acknowledgments

We thank Jason Hartmann, Lucas Ward, and Shaun Stewart for providing key support in software and simulation design as well as generating results. We thank Brendan O'Connor's for his excellent ideas about software architecture and Matthew Ruschmann for his knowledge of FDIR design. We are also grateful for the contributions of the other members of the Emergent Space Technologies System F6 Team.

This work was sponsored by DARPA under the System F6 program. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

[1]Tactical Technology Office, "DARPA-BAA-11-01," October 2010.

[2]Schmidt, J. and Phillips, M., "Cluster Configuration Design Considerations," *Astrodynamics Specialist Conference*, AAS/AIAA, Hilton Head, South Carolina, August 2013.

[3]Kelbel, D., Lee, T., Long, A., Carpenter, J. R., and Gramling, C., "Evaluation of Relative Navigation Algorithms for Formation-Flying Satellites," *Flight Mechanics Symposium.*, Vol. 1, June 2001.

[4]D'Amico, S., *Autonomous Formation Flying in Low Earth Orbit*, Ph.D. thesis, TU Delft, 2010.

[5]D'Amica, S., Ardaens, J.-S., and Larsson, R., "Spaceborne Autonomous Formation Flying Experiment on the PRISMA Mission," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 3, 2012.

[6]Saenz-Otero, A., Katz, J., and Millder, D. W., "SPHERES Demonstrations of Satellite Formations aboard the ISS," *AAS Guidance and Control Conference*, Breckenridge, Colorado, Feb 2009.

[7]Nolet, S., Seanz-Otero, A., Millder, D. W., and Fejzic, A., "SPHERES Operations aboard the ISS: Matruation of GN&C Algorithms in Microgravity," AAS, 2007.

[8]Nolet, S., "The SPHERES Navigation System: from Early Development to On-Orbit Testing," AIAA, 2007.

[9]Wilson, E., Sutter, D. W., and Mah, R. W., "Motion-Based Thruster Fault Detection and Isolation," *AIAA Conference*, Arlington, Virginia, September 2005.

[10]Schutz, B. E., Tabley, B. D., and Born, G. H., *Statistical Orbit Determination*, Elsevier Academic Press, 2004.

[11]Montenbruck, O. and Gill, E., *Satellite Orbits*, Springer, The Netherlands, 1st ed., 2005.

[12]Simpson, R. and Revell, J., "Towards a Taxonomy of Performance Metrics, Bounds and Tests for Tracking and SLAM Algorithms," *Systems Engineering for Autonomous Systems Defence Technology Centre*, Vol. 4, No. AA022, July 2009.

[13]Mahalanobis, P. C., "On the generalised distance in statistics," *Proceedings National Institute of Science, India*, Vol. 2, No. 1, April 1936, pp. 49–55.