USER'S GUIDE TO THE FAULT INFERRING NONLINEAR DETECTION SYSTEM (FINDS) COMPUTER PROGRAM

A.K. Caglayan, P.M. Godiwala, and H.S. Satz

CHARLES RIVER ANALYTICS INC.
Cambridge, MA

Contract NAS1-17719
June 1988

NASA
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. FINDS Algorithm Overview</td>
<td>3</td>
</tr>
<tr>
<td>3. FINDS Algorithm Implementation</td>
<td>7</td>
</tr>
<tr>
<td>4. Subprogram Flow Charts</td>
<td>22</td>
</tr>
<tr>
<td>5. Input and Output Files</td>
<td>32</td>
</tr>
<tr>
<td>6. Program Variable Indexing Tables</td>
<td>35</td>
</tr>
<tr>
<td>7. Subprogram Description and Tables</td>
<td>43</td>
</tr>
<tr>
<td>8. Common Block Description and Tables</td>
<td>64</td>
</tr>
<tr>
<td>9. References</td>
<td>84</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1: FINDS Program Overview</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2.2: Runway Coordinate System and MLS Geometry</td>
<td>6</td>
</tr>
<tr>
<td>Figure 3.1: FINDS Main Program Execution Flow</td>
<td>8</td>
</tr>
<tr>
<td>Figure 4.1: Flow Chart for Subprogram NAV</td>
<td>23</td>
</tr>
<tr>
<td>Figure 4.2: Flow Chart for Subprogram EKFN1</td>
<td>24</td>
</tr>
<tr>
<td>Figure 4.3: Flow Chart for Subprogram BLEND</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4.4: Flow Chart for Subprogram ISOLAT</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4.5: Flow Chart for Subprogram RECONF</td>
<td>26</td>
</tr>
<tr>
<td>Figure 4.5a: Partition A of Subprogram RECONF</td>
<td>26</td>
</tr>
<tr>
<td>Figure 4.5b: Partition B of Subprogram RECONF</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4.5c: Partition C of Subprogram RECONF</td>
<td>27</td>
</tr>
<tr>
<td>Figure 4.5d: Partition D of Subprogram RECONF</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.6: Hierarchical FDI Test</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.7: Isolation Logic</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.7a: BMA/IAS Failure Isolation</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.7b: MLS/IAS Failure Isolation</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4.7c: RG Failure Isolation</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4.7d: IMU Failure Isolation</td>
<td>31</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 5.1: Design Values for No-Fail Filter Noise Parameters</td>
<td>32</td>
</tr>
<tr>
<td>Table 5.2: Detector Design Values for Measurement Sensor Noise</td>
<td>33</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Table 6.1: NFF Absolute State Indexing Convention</td>
<td>36</td>
</tr>
<tr>
<td>Table 6.2: NFF Absolute Measurement Indexing Convention</td>
<td>37</td>
</tr>
<tr>
<td>Table 6.3: NFF Absolute Input Indexing Convention</td>
<td>38</td>
</tr>
<tr>
<td>Table 6.4: NFF Process Noise Indexing Convention</td>
<td>39</td>
</tr>
<tr>
<td>Table 6.5: Absolute Sensor Indexing Convention</td>
<td>40</td>
</tr>
<tr>
<td>Table 6.6: Replicated Sensor Indexing Convention</td>
<td>41</td>
</tr>
<tr>
<td>Table 6.7: Replicated Measurement Indexing Convention</td>
<td>42</td>
</tr>
<tr>
<td>Table 7.1: Subprograms</td>
<td>43</td>
</tr>
<tr>
<td>Table 8.1: Common Blocks</td>
<td>64</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

This report describes the operation and internal structure of the computer program FINDS (Fault Inferring Nonlinear Detection System) developed by Charles River Analytics Inc. for the NASA Langley Research Center. FINDS has been developed to provide detection, isolation, and compensation for hardware failures in the flight control sensors and ground-based navigation aids [1-4].

The FINDS algorithm is designed to provide reliable estimates for aircraft position, velocity, attitude, and horizontal winds to be used for guidance and control laws in the presence of possible failures in the avionics sensors. The FINDS algorithm exploits analytic redundancy between similar as well as dissimilar sensors; it can isolate a failure in a duplicate sensor configuration and detect a failure even if there is only one sensor of a given type in the configuration. FINDS can also detect simultaneous failures in navigation aid sensors, arising for instance from ground antenna malfunctions. Hence, FINDS can be used to increase the reliability of a sensor configuration with a given redundancy. For example, the fail-operational/fail-safe capability of a triply redundant voting system can be improved to at least a fail-op/fail-op/fail-safe capability. Conversely, FINDS can be employed to reduce the hardware redundancy requirements for a given reliability figure. As an example, FINDS can be used to replace a triply redundant voting system with dual redundancy while maintaining the overall reliability of the system.

The FINDS algorithm consists of 1) a no-fail filter (NFF), which is an extended Kalman filter (EKF) based on the assumption of no sensor failures and which provides estimates for aircraft states, horizontal winds, and normal operating sensor biases; 2) a set of test-of-mean detection tests implemented over moving windows of the NFF residuals; 3) a bank of first order filters activated upon failure detection to estimate failure levels in individual
sensors; and 4) a decision function which isolates the failed sensor by selecting the most likely failure mode depending on the likelihood ratios. When a sensor failure is detected and isolated, the algorithm is restructured to eliminate the failed sensor from further processing and to remove the accumulated effects of the sensor failure on the NFF. Failure identification decisions are monitored with the use of a healer algorithm; sensors falsely identified as failed or sensors recovered from failures are restored to the system.

The FINDS algorithm was developed with the use of a digital simulation of a commercial transport aircraft (B-737) [1-4]. Flight recorded data for this aircraft were used to address the issues of sensor modeling inaccuracies, such as time-varying sensor bias and time correlated noise [5-6]. The FINDS algorithm was then modified to "fit" the size constraints of a flight computer and to meet real-time execution requirements without compromising sensor failure detection and isolation (FDI) and state estimation performance [7-10].

To meet the real-time execution requirements, the FINDS algorithm has been partitioned to execute on a dual parallel processor configuration: one based on the translational dynamics and the other on the rotational kinematics. In addition, a new hierarchical failure isolation strategy has been developed, replacing the multiple hypothesis test in the earlier versions. Finally, a multi-rate implementation of the FINDS algorithm has been implemented to further increase execution speed.

The outline of the report is as follows. An overview of the FINDS algorithm is given in the next section. The implemented equations are given in detail in Section 3. Section 4 contains the flow charts for the key subprograms. The input and output files are discussed in Section 5. Program variable indexing convention is presented as tables in Section 6. Subprogram descriptions are presented in Section 7. Finally, Section 8 contains the common block descriptions used in the program.
Given a configuration of avionics sensors on an aircraft, the FINDS algorithm generates fault tolerant estimates for the vehicle states as required by the flight control, guidance, and navigation systems in the presence of possible sensor failures. The desired qualities of FINDS are 1) use of analytical redundancy concepts to minimize hardware replication requirements; 2) timely detection of sensor failures; 3) ability to detect all types of sensor failures; 4) acceptable false alarm/detection probability performance; 5) ability to recover from false alarms; and 6) minimal computational complexity to permit real time operation on flight qualified computers.

The FINDS algorithm baseline structure is shown in Figure 2.1. The replicated sensor measurements are separated according to their function in the no-fail filter. That is, accelerometer and gyro measurements are used as input sensors to integrate the vehicle point mass equations of motion, and the remaining sensors (MLS, IAS, and IMU) are used as measurement sensors. The input sensors are processed in selection logic, and similar measurement sensors are averaged to reduce the overall complexity of the computations without a loss of generality.

The NFF shown in Figure 2.1 is an EKF which is implemented on the assumption of no sensor failures. The EKF development is based on discrete time difference equations for the vehicle equations of motion. The NFF provides estimates for the aircraft position, velocity, attitude, and horizontal winds, and estimates for the normal operating biases associated with a specified subset of the input and measurement sensors.
Replicated aircraft sensor measurements

- Accel. & gyro measurements
- MLS, IAS, IMU measurements

Select
Average

No-fail filter estimates
Automatic guidance control

Moving window detectors

Averaged measurement residuals

Expanding measurement residuals

Yes
Failure?
No

LKF long. accel. Replication 1
LKF roll rate gyro Replication 1
LKF MLS range Replication 1
LKF IAS Replication 1
LKF IMU roll Replication 1

Likelihood ratio Long. accel. 1
Likelihood ratio Roll rate gyro 1
Likelihood ratio MLS range 1
Likelihood ratio IAS 1
Likelihood ratio IMU roll 1

Expanded residuals

Failure
Compensated residuals

Multiple hypothesis test
Isolation decision
Reconfiguration/Healing

Figure 2.1: FINDS Algorithm Baseline Structure
The formulation yields a computationally efficient EKF implementation in which the input sensors are integrated into the NFF without closed loop filtering. Only one set of input sensors and the average of the measurement sensors are used. The remaining replicated sensors are held in standby and inserted as failures are detected and isolated. A decomposition procedure based on the separated EKF algorithm provides the EKF filter gains [11]-[12].

The NFF also generates a residual sequence for the averaged measurements, as seen in Figure 2.1, and a detection test is performed on these residuals over a moving window. The length of the moving window is different for input sensors and measurement sensors. A test of mean is compared to a predetermined threshold to determine a sensor failure. If a sensor failure is detected, the bank of detectors is run using the saved residuals in the corresponding moving window memory. The failure levels are estimated and the failure is isolated depending on the computed likelihood ratios.

When a failure is isolated, a reconfiguration algorithm is used to restructure the FINDS algorithm [13]. When a gyro or accelerometer (input sensor) fails, the faulty sensor is replaced. If there are no more valid sensors of that type, the NFF is restructured, provided it is able to function with the remaining set of sensors. When a measurement sensor fails, the isolated sensor is flagged to be inactive, and appropriate changes are made in the NFF noise statistics; also, the NFF is collapsed to accommodate the loss of all the sensors of a given type. The reconfiguration block also functions to reinitialize the NFF, detectors, and likelihood ratios following identification of a failure.

To recover from false alarms, each failed sensor is given a healing test. Input sensors are tested by comparison with sensors of the same type used by the NFF. A failed measurement sensor is tested with the NFF estimate of that
sensor. These are binary hypothesis tests conditioned on the decision rule that the sensor currently in use is healthy.

The NFF state estimates are initialized using the first iteration of the flight data, which includes MLS azimuth, elevation, and range, IAS, and IMU pitch, roll, and yaw measurements, to compute the aircraft position, velocity, attitude, and horizontal winds in the runway frame, shown in Figure 2.2 as required by the NFF.

Figure 2.2: Runway Coordinate System and MLS Geometry
3. FINDS ALGORITHM IMPLEMENTATION

The interactive version of FINDS suitable for operation in a simulation environment was developed on a DEC VAX 11/780 using FORTRAN 77 under the VMS operating system. The flight data driven version of FINDS suitable for operation using either flight recorded or simulation generated sensor data was developed on Charles River Data Systems Universe 68/35 using FORTRAN 77 under the UNOS operating system, and SUN 3/160 using FORTRAN 77 under SunOS operating system. Several modifications have been made to the interactive version of FINDS to reduce the size and increase the speed of the algorithm, and to improve state estimation and sensor FDI performance. The composite version, FINDSCMP, of FINDS incorporates these changes, in particular, the hierarchical isolation strategy and multi-rate implementation. In addition, the FINDS algorithm has been partitioned into two parts for a parallel processing architecture: FINDSI processing the sensors related to rotational kinematics and FINDS2 processing the sensors related to the translational dynamics. The partitioned version of FINDS has been ported onto a dual-processor configured ROLM 1666 flight computer using ROLM FORTRAN 66 compiler under the ROLM Real Time Operating System. A DMA local data communication link has been used for communication among the processors. In this section, the implemented equations for FINDSCMP, FINDSI, and FINDS2 are described. The execution flow of the main program is illustrated in Figure 3.1.
Figure 3.1: FINDS Main Program Execution Flow
FINDSCMP (Composite Version)

Number of states, \(NX = 11\)

State vector, \(\dot{\mathbf{x}} = [\dot{x}, \dot{y}, \dot{z}, \dot{x}, \dot{y}, \dot{z}, \phi, \psi, \omega_x, \omega_y]^{T}\)

Number of biases, \(NB = 6\)

Bias vector, \(\mathbf{b} = [b_a, b_a, b_a, b_p, b_q, b_r]^{T}\)

Number of measurement types, \(NY = 7\)

Measurement vector, \(\mathbf{y} = [\text{MLS}_{az}, \text{MLS}_{e}, \text{MLS}_{rn}, \text{IAS}, \text{IMU}_{\phi}, \text{IMU}_{\theta}, \text{IMU}_{\psi}]\)

Number of input types, \(NUl = 6\)

Input vector, \(\mathbf{u} = [a_x, a_y, a_z, p, q, r]^{T}\)

--- New Time Iteration Start: time \(\text{'k'}\) ---

\(\text{READFL} : \) read the NFF input sensors \(\mathbf{u}^n(k)\) and the NFF measurement sensors

\(\mathbf{y}^n_j(k) ; i=1,6 ; j=1,7 ; n=1,2 \) (dual replication)

\(\text{INITXF:} \) Compute the NFF initial state estimates using the first iteration of

flight data. Denoting the aircraft position in the MLS frame by \(r_{xm}, r_{ym}, r_{zm}\):

\[
\dot{r}_{xm}(k) = \sqrt{f + (f^2 - h)}
\]

\[
\dot{r}_{ym}(k) = -y_{rn}(k) \cdot [\sin(y_{az}(k))]
\]

\[
\dot{r}_{zm}(k) = \sqrt{y_{rn}(k) + \dot{r}_{xm}(k) - \dot{r}_{ym}(k)}
\]

where

\[
f = x_{oe} \cdot [\sin(y_{e}(k))]^2
\]

\[
h = y_{oe} \cdot [\sin(y_{e}(k))]^2 + y_{rn}(k) \cdot [\cos(y_{e}(k))]^2
\]

\[- 2 \cdot y_{rn}(k) \cdot [\sin(y_{e}(k))]^2 - (z_{oe} - 2 \cdot z_{rn}) \cdot [\cos(y_{e}(k))]^3\]

- 9 -
where \((x_{oe}, y_{oe}, z_{oe})\) represent the coordinates of the elevation antenna in the MLS frame.

\[ \begin{align*}
\dot{x}_{o}^{(k)} &= x_M - \dot{x}_{xm}^{(k)} \\
\dot{y}_{o}^{(k)} &= y_M + \dot{y}_{ym}^{(k)} \\
\dot{z}_{o}^{(k)} &= z_M - \dot{z}_{zm}^{(k)}
\end{align*} \]

where \((x_M, y_M, z_M)\) are the azimuth/range antenna coordinates in the runway frame.

\[ \begin{align*}
\dot{x}_{o}^{(k)} &= y_{sp}^{(k)} \cdot \cos(y_{\phi}^{(k)}) \cdot \cos(y_{\psi}^{(k)}) + \dot{w}_{x}^{(k)} \\
\dot{y}_{o}^{(k)} &= y_{sp}^{(k)} \cdot \cos(y_{\phi}^{(k)}) \cdot \sin(y_{\psi}^{(k)}) + \dot{w}_{y}^{(k)} \\
\dot{z}_{o}^{(k)} &= -y_{sp}^{(k)} \cdot \sin(y_{\phi}^{(k)}) \\
\dot{w}_{x}^{(k)} &= 0 \\
\dot{w}_{y}^{(k)} &= 0
\end{align*} \]

The initial estimates for the aircraft attitude are obtained by averaging the replicated IMU measurements:

\[ \begin{align*}
\phi^{(k)} &= \frac{\gamma_{\phi}^{(k)} + \gamma_{\phi}^{(k)}}{2} \\
\theta^{(k)} &= \frac{\gamma_{\theta}^{(k)} + \gamma_{\theta}^{(k)}}{2} \\
\psi^{(k)} &= \frac{\gamma_{\psi}^{(k)} + \gamma_{\psi}^{(k)}}{2} - \psi_R
\end{align*} \]

where \(\psi_R\) is the runway yaw, fixed for the given runway configuration.

SUMIN : (i) compensate rate-gyros for earth's rotation effects
(ii) average inputs and compensate for biases:

\[ \dot{u}_{l}^{c}(k) = \frac{u_{l}^{c}(k) + u_{l}^{c}(k-1)}{2} - b_{l}^{c}(k-1) \]

where \(c\) denotes the current active replication

EKFNL(2) : (i) UPDB \(\rightarrow\) update input transition matrix \(B(x(k-1))\)
\[
\begin{bmatrix}
\Delta^2/2 & T_{GB}(\dot{x}(k-1)) & 0 \\
\Delta & T_{GB}(\dot{x}(k-1)) & 0 \\
0 & \Delta T_{ER}(\dot{x}(k-1)) & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

where the transformation from the body axes into the ground frame is computed according to:

\[
T_{GB}(\dot{x}(k-1)) =
\begin{bmatrix}
\dot{c}\dot{\theta}\dot{\phi} & \ddot{s}\dot{\theta}\dot{\phi}-c\dot{\phi}\ddot{s} & c\dot{\phi}\ddot{s}+\ddot{c}\dot{\phi}
\\
\ddot{c}\dot{\theta}\dot{\phi} & \ddot{s}\dot{\theta}\dot{\phi}+c\dot{\phi}\ddot{s} & -c\dot{\phi}\ddot{s}-\ddot{c}\dot{\phi}
\\
-s\dot{\theta} & \dot{s}\dot{\phi} & c\dot{\phi}
\end{bmatrix}
\]

where \(\dot{\phi}(k-1), \dot{\theta}(k-1), \dot{\psi}(k-1)\) are the NFF estimates for the Euler angles and \(c, s, t\) are abbreviations for the cosine, sine and tangent functions, respectively. The matrix \(T_{ER}\) relating the body rates to the Euler angles is computed according to:

\[
T_{ER}(\dot{x}(k)) =
\begin{bmatrix}
1 & t(\dot{\theta}(k))\dot{s}(\dot{\phi}(k)) & t(\dot{\theta}(k))c(\dot{\phi}(k))
\\
0 & c(\dot{\phi}(k)) & -s(\dot{\phi}(k))
\\
0 & s(\dot{\phi}(k))sc(\dot{\theta}(k)) & c(\dot{\phi}(k))sc(\dot{\theta}(k))
\end{bmatrix}
\]

where \(sc\) is the abbreviation for the secant function.

(ii) UPDQ --- update process noise covariance \(Q(\dot{x}(k-1))\)

\[
Q(\dot{x}(k)) =
\begin{bmatrix}
\frac{\Delta^3}{3} T_{GB}^T a_{GB}^T & \frac{\Delta^2}{2} T_{GB}^T V_{TGB}^T & 0 & 0 \\
\frac{\Delta^2}{2} T_{GB}^T a_{GB}^T & \Delta T_{ER} V_{r_{ER}}^T T_{ER} & 0 & 0 \\
0 & 0 & 0 & \int_0^\Delta e^T a_{GB}^T V_{w} e^T a_{GB}^T ds
\end{bmatrix}
\]

where \(V_{a}\) is the covariance for the accelerometer sensor noises given by
\[ \sigma_{ax}^2 \begin{bmatrix} 0 & 0 \\ 0 & 2 \sigma_{ay} \\ 0 & 0 & \sigma_{az}^2 \end{bmatrix} \]

where \( \sigma_{ax}, \sigma_{ay}, \sigma_{az} \) are the accelerometer sensor noise standard deviations.

\( V_{rg} \) is the covariance for the rate gyro sensor noises given by:

\[
V_{rg} = \begin{bmatrix}
V_{rg}(1) & 0 & 0 \\
0 & V_{rg}(2) & 0 \\
0 & 0 & V_{rg}(3)
\end{bmatrix}
\]

with

\[
V_{rg}(1) = \sigma_p^2 + \text{SPM} \times (A_{q}^2 + A_{r}^2) + \text{SCF} \times A_{p}^2 \\
V_{rg}(2) = \sigma_q^2 + \text{SPM} \times (A_{p}^2 + A_{r}^2) + \text{SCF} \times A_{q}^2 \\
V_{rg}(3) = \sigma_r^2 + \text{SPM} \times (A_{p}^2 + A_{q}^2) + \text{SCF} \times A_{r}^2
\]

where \( \sigma_p, \sigma_q, \sigma_r \) are the rate gyro measurement noise standard deviations; \( A_p, A_q, A_r \) are the averaged \( p, q, r \) measurements passed through symmetric limiters with thresholds 4 deg/s, 1 deg/sec, and 2.5 deg/sec:

\[
A_p = \frac{p_{ml}(k) + p_{ml}(k)}{2} ; \quad A_q = \frac{q_{ml}(k) + q_{ml}(k)}{2} ; \quad A_r = \frac{r_{ml}(k) + r_{ml}(k)}{2}
\]

where SCF is the rate gyro scale factor error variance, and SPM is the sum of SCF and rate gyro misalignment error variances.

The wind model system matrix \( A_w \) is given by

\[
A_w = \begin{bmatrix}
-\frac{1}{r_w} & 0 \\
0 & -\frac{1}{r_w}
\end{bmatrix}
\]

where \( r_w \) is the time constant associated with the wind model.
(iii) Compute prediction error covariance via:

\[ P_n(k/k-1) = A \cdot P_n(k-1/k-1) \cdot A^T + Q(x(k-1)) \]

where \( A \) is constant state transition matrix given by

\[
A = \begin{bmatrix}
  I & 0 & 0 & 0 \\
  0 & I & 0 & 0 \\
  0 & 0 & I & 0 \\
  0 & 0 & 0 & e^{\mathbf{A} \Delta}
\end{bmatrix}
\]

**BLEND(2):** (i) Compute single stage prediction:

\[
x(k/k-1) = A \cdot x(k-1) + B(x(k-1)) \cdot u(k)
\]

(ii) Update single stage prediction for measurements UPDH \( \rightarrow \)

\[
h(x(k/k-1)) = \tilde{y}_{az}(k/k-1) = \frac{1}{\sigma_{az} \tilde{e}_{az}(k/k-1)} \sin^{-1}\left(\frac{-\tilde{x}_{az}(k/k-1) + \tilde{y}_{az}(k/k-1)}{\tilde{r}_{az}(k/k-1)}\right)
\]

\[
h_2(\hat{x}(k/k-1)) = \tilde{y}_{e}(k/k-1) = \frac{1}{\sigma_{e} \tilde{e}_{e}(k/k-1)} \sin^{-1}\left(\frac{-\tilde{x}_{e}(k/k-1) + \tilde{z}_{e}(k/k-1)}{\tilde{r}_{e}(k/k-1)}\right)
\]

\[
h_3(\hat{x}(k/k-1)) = \tilde{y}_{rn}(k/k-1) = \frac{1}{\sigma_{rn} \tilde{e}_{rn}(k/k-1)} \sin^{-1}\left(\frac{-\tilde{x}_{rn}(k/k-1) + \tilde{y}_{rn}(k/k-1) + \tilde{z}_{rn}(k/k-1)}{\tilde{r}_{rn}(k/k-1)}\right)
\]

where \((x_M, y_M, z_M)\) and \((x_E, y_E, z_E)\) are the azimuth and elevation antenna positions in the runway frame, \(\sigma_{az}, \sigma_{e}, \sigma_{rn}\) are the averaged MLS sensor noise standard deviations, and \(\tilde{r}_{az}, \tilde{r}_{e}, \tilde{r}_{rn}\) are single stage predictions for the aircraft range from the azimuth and elevation antennas given by:

\[
\tilde{r}_{az}(k/k-1) = \sqrt{(\tilde{r}_x(k/k-1)-x_M)^2 + (\tilde{r}_y(k/k-1)-y_M)^2 + (\tilde{r}_z(k/k-1)-z_M)^2}
\]

\[
\tilde{r}_{e}(k/k-1) = \sqrt{(\tilde{r}_x(k/k-1)-x_E)^2 + (\tilde{r}_y(k/k-1)-y_E)^2 + (\tilde{r}_z(k/k-1)-z_E)^2}
\]

\[
h_4(\hat{x}(k/k-1)) = \tilde{y}_{sp}(k/k-1) = \frac{1}{\sigma_{sp}} \sqrt{[\tilde{r}_x(k/k-1)-\tilde{w}_x(k/k-1)]^2 + [\tilde{r}_y(k/k-1)-\tilde{w}_y(k/k-1)]^2 + [\tilde{r}_z(k/k-1)-\tilde{w}_z(k/k-1)]^2}
\]

where \(\sigma_{sp}\) is the averaged IAS sensor noise standard deviation.

\[
h_5(\hat{x}(k/k-1)) = \tilde{y}_{\phi}(k/k-1) = \frac{1}{\sigma_{\phi}} \phi(k/k-1)
\]
\[ h_6(\hat{x}(k/k-1)) = y_\theta(k/k-1) = \frac{1}{\sigma_\theta} \theta(k/k-1) \]

\[ h_7(\hat{x}(k/k-1)) = y_\psi(k/k-1) = \frac{1}{\sigma_\psi} \psi(k/k-1) \]

where \( \sigma_\theta, \sigma_\theta, \sigma_\psi \) are the averaged IMU sensor noise standard deviations.

**SUMOUT**:

\[ Y_j(k) = \frac{y_j(k) + y_{\bar{j}}(k)}{2} \]

**EKFNL(l)**: (i) **UPDPH** \( \rightarrow \) update the partials of the measurements

The nonzero elements of the measurement partial \( H(\hat{x}(k/k-1)) \) are computed according to:

\[ H_{1,1} = \frac{\hat{r}_x(k/k-1) - x_M}{\hat{r}_{xz}(k/k-1) \cdot \sigma_{xz}} \]

\[ H_{1,2} = \frac{\hat{r}_y(k/k-1) - y_M}{\hat{r}_{yz}(k/k-1) \cdot \sigma_{yz}} \]

\[ H_{1,3} = \frac{\hat{r}_z(k/k-1) - z_M}{\hat{r}_{xz}(k/k-1) \cdot \sigma_{xz}} \]

\[ H_{2,1} = \frac{\hat{r}_{x(xz)}(k/k-1) - y_M \hat{r}_{x(yz)}(k/k-1) - x_M}{\hat{r}_{xz}(k/k-1) \cdot \hat{r}_{xx}(k/k-1) \cdot \sigma_{e_x}} \]

where \( \hat{r}_{xz}(k/k-1) = \sqrt{(\hat{r}_x(k/k-1) - x_M)^2 + (\hat{r}_z(k/k-1) - z_M)^2} \)

\[ H_{2,2} = \frac{-\hat{r}_{xz}(k/k-1)}{\hat{r}_{xx}(k/k-1) \cdot \sigma_{e_x}} \]

\[ H_{2,3} = \frac{\hat{r}_{y(yz)}(k/k-1) - y_M \hat{r}_{z(yz)}(k/k-1) - z_M}{\hat{r}_{xz}(k/k-1) \cdot \hat{r}_{xz}(k/k-1) \cdot \sigma_{e_x}} \]
\[ H_{3,1} = \frac{(\hat{e}_{x} (k/k-1) - y_E) (\hat{e}_{z} (k/k-1) - z_E)}{\hat{e}_{x}^2 (k/k-1) \cdot \hat{e}_{x} (k/k-1) \cdot \sigma_{rn}} \]

where \( \hat{e}_{xy} (k/k-1) = \sqrt{(\hat{e}_{x} (k/k-1) - x_E)^2 + (\hat{e}_{y} (k/k-1) - y_E)^2} \)

\[ H_{3,2} = \frac{(\hat{e}_{y} (k/k-1) - y_E) (\hat{e}_{z} (k/k-1) - z_E)}{\hat{e}_{y}^2 (k/k-1) \cdot \hat{e}_{xy} (k/k-1) \cdot \sigma_{rn}} \]

\[ H_{3,3} = \frac{-\hat{e}_{xy} (k/k-1)}{\hat{e}_{xy}^2 (k/k-1) \cdot \sigma_{rn}} \]

\[ H_{4,4} = \frac{\hat{e}_{x} (k/k-1) - \hat{w}_{x} (k/k-1)}{\hat{s}(k/k-1) \cdot \sigma_{sp}} \]

where \( \hat{s}(k/k-1) = \sqrt{(\hat{e}_{x} (k/k-1) - \hat{w}_{x})^2 + (\hat{e}_{y} (k/k-1) - \hat{w}_{y})^2 + \hat{e}_{z} (k/k-1)} \)

\[ H_{4,5} = \frac{\hat{e}_{y} (k/k-1) - \hat{w}_{y} (k/k-1)}{\hat{s}(k/k-1) \cdot \sigma_{sp}} \]

\[ H_{4,6} = \frac{\hat{e}_{z} (k/k-1)}{\hat{s}(k/k-1) \cdot \sigma_{sp}} \]

\[ H_{4,10} = -H(4,4) \]

\[ H_{4,11} = -H(4,5) \]

(ii) Compute the bias-free NFF gain:

\[ K_x (k) = P_x (k/k-1) \cdot [H \cdot P_x (k/k-1) \cdot H^T + R(k)]^{-1} \]
(iii) Compute the bias-free NFF single stage prediction error covariance:

\[ P_e(k/k) = [I - K_x \cdot H] \cdot P_e(k/k-1) \cdot [I - K_x \cdot H]^T + K_x \cdot R(k) \cdot K_x^T \]

where \( R(k) = \text{diag} \{ l/c_i \} \)

**BIASF(1):**

(i) Update bias observation matrix:

\[ C_b(k) = H \cdot [A \cdot V_b(k-1) + B] + D \]

(ii) Update bias propagation matrix:

\[ V_b(k) = [I - K_x \cdot H] \cdot A \cdot V_b(k-1) + [-B + K_x \cdot (H \cdot B - D)] \]

(iii) Compute the NFF bias gain:

\[ K_b(k) = P_b(k-1) \cdot C_b^T(k) + [C_b(k) \cdot P_b(k-1) \cdot C_b^T(k) + R_b(k)]^{-1} \]

where \( R_b(k) = [H \cdot P_e(k/k-1) \cdot H^T + R(k)] \) from EKFNI(1)

(iv) Compute the NFF bias estimation error covariance:

\[ P_b(k) = [I - K_b(k) \cdot C_b(k)] \cdot P_b(k-l) \]

**BLEND(1):**

(i) Compute averaged measurement residuals:

\[ r(k) = \bar{y}(k) - h(\hat{x}(k/k-1)) \]

(ii) Update state estimate:

\[ \hat{x}(k) = \hat{x}(k/k-1) + [K_x(k) \cdot V_b(k) \cdot K_b(k)] \cdot r(k) \]

(iii) Update bias estimates:

\[ b(k) = b(k-1) + K_b(k) \cdot r(k) \]

**DESCMP:** Evaluate expanded measurement residual and store in moving window

\[ r(k) = \begin{bmatrix} y_1^{1/\sigma_1} - h_{1}(\hat{x}(k/k-1)) \\ y_1^{2/\sigma_1} - h_{1}(\hat{x}(k/k-1)) \end{bmatrix} \]

**DET01:** (i) Compensate measurement residual covariance inverse RTINV using sensor noise parameters for window 01
(ii) Compute the likelihood ratio for moving window 01

\[ \text{LRT01}(k) = r_{01}^T(k) \cdot \text{RTINV}_{01} \cdot r(k) \]

If \( \text{LRT01} < \text{threshold}_{01} \), then no measurement sensor failure
else ISOLATE (01)

**DET05:**
(i) Compensate RTINV for moving window 05

(ii) Compute measurement residual average for moving window 05:

\[ \bar{r}_{05}(k) = \frac{1}{5} \sum_{j=k-4}^{k} r(j) \]

(iii) Compute likelihood ratio LRT05 for under 05

\[ \text{LRT05}(k) = \bar{r}_{05}^T(k) \cdot \text{RTINV}_{05} \cdot \bar{r}_{05}(k) \]

If \( \text{LRT05} < \text{threshold}_{05} \), then no sensor failures
else ISOLATE (05)

**DET10:**
(i) Compensate RTINV for window 10

(ii) Compute measurement residual average for moving window 10

\[ \bar{r}_{10}(k) = \frac{1}{10} \sum_{j=k-9}^{k} r(j) \]

(iii) Compute likelihood ratio LRT10 for moving window 10

\[ \text{LRT10}(k) = \bar{r}_{10}^T(k) \cdot \text{RTINV}_{10} \cdot \bar{r}_{10}(k) \]

If \( \text{LRT10} < \text{threshold}_{10} \), then no sensor failures
else ISOLATE (10)

**ISOLATE (w):**
(i) Form prediction error covariance for composite state:

\[
\text{PXF}(k) =
\begin{bmatrix}
P_x(k) & P_{xb}(k) \\
\text{T}\ P_{xb}(k) & P_b(k)
\end{bmatrix}
\]
where

\[
P_x(k) = P_o(k/k) + [A \ast VB(k) \ast B(\hat{x}(k-1))] \ast P_b(k/k) \ast [A \ast VB(k) \ast B(\hat{x}(k-1))]^T
\]

\[
P_{xb}(k) = [A \ast VB(k) \ast B(\hat{x}(k-1))] \ast P_b(k/k)
\]

\[
P_b(k) = P_b(k/k)
\]

(ii) Compute inverse of innovation covariance

\[
\hat{R}^{-1}(k) = ([\bar{H} \bar{D}] \ast PXF(k) \ast [\bar{H} \bar{D}]^T + [R \ 0])^{-1}
\]

\[
R = \text{diag} \left[ (\sigma_{dw}/\sigma_i)^{**2} \right]
\]

(iii) Compute failure observation matrix:

\[
C_i(k, \hat{x}(k)) = [\bar{H} \bar{D}] \begin{bmatrix} A & -B(\hat{x}(k-1)) \\ 0 & I \end{bmatrix} \ast \begin{bmatrix} V_{ix}(k-1) \\ V_{ib}(k-1) \end{bmatrix} + [\bar{H} \bar{D}] \begin{bmatrix} -B_i(\hat{x}(k-1)) \\ 0 \end{bmatrix} + D_i(1/\sigma_i)
\]

(iv) Compute failure propagation matrix

\[
\begin{bmatrix} V_{ix}(k) \\ V_{ib}(k) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} K_o(k) \\ K_b(k) \end{bmatrix} [\bar{H} \bar{D}] \ast \begin{bmatrix} A & -B(\hat{x}(k-1)) \\ 0 & I \end{bmatrix} \ast \begin{bmatrix} V_{ix}(k-1) \\ V_{ib}(k-1) \end{bmatrix} + [\bar{H} \bar{D}] \begin{bmatrix} -B_i(\hat{x}(k-1)) \\ 0 \end{bmatrix} \]

\[
- \begin{bmatrix} K_o(k) \\ K_b(k) \end{bmatrix} [\bar{H} \bar{D}] \begin{bmatrix} -B_i(\hat{x}(k-1)) \\ 0 \end{bmatrix} + D_i(1/\sigma_i)
\]

(v) Compute failure level estimates

\[
\hat{m}_i(k) = \hat{m}_i(k-1) + G_i(k) \ast \text{RES}(k)
\]

where

\[
\text{RES}(k) = [r_o(k) \ast C_i(k) \ast \hat{m}_i(k-1)]
\]

\[
G_i(k) = [C_i^T(k) \ast \hat{R}^{-1}(k)]/P_i(k/k)
\]

\[
P_i(k/k) = P_i(k-1/k-1) + C_i^T(k) \ast \hat{R}(k)^{-1} \ast C_i(k)
\]
(vi) Compute likelihood ratios
\[ a_i(k) = RES^T(k) \times R^{-1}(k) \times RES(k) + a_i(k-1) \]

**NOTE:** Steps (iii) --> (vi) are performed in loop 'w' number of times depending on which window has detected failure.
\[ C_i(0) = V_i(0) = P_i(0/0) = 0 , \quad a_i(0) = -12 \times \ln (\text{Prior}_i) \]

**DECIDE:** Find the minimum \( a \) and check failure level constraint \( \hat{m}_i > 1.0 \),
=> isolate failed sensor

**RECONF (-1):** Reconfigure system for any new 'failed' sensor
Check if system can operate with remaining set --> else ABORT

**GTOI**
1) compute a/c latitude & longitude
2) compute rate-gyro compensation terms.
3) compute gravity vector

---End of Time 'k'---

**FINDS1 (Rotational Kinematics)**

**Number of states, \( NX = 3 \)**
State vector, \( \hat{x} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T \)
**Number of biases, \( NB = 3 \)**
Bias vector, \( \hat{b} = [\dot{b}_p, \dot{b}_q, \dot{b}_r]^T \)
**Number of measurement types, \( NY = 3 \)**
Measurement vector, \( y = [\text{IMU}_\phi, \text{IMU}_\theta, \text{IMU}_\psi]^T \)
**Number of inputs, \( NU_1 = 3 \)**
Input vector, \( u = [p, q, r]^T \)

---New Time Iteration Start: time `k`---

**READFL**: Read the NFF input sensors \( u_i^n(k) \), and the NFF measurement sensors, \( y_j^n(k) \), \( i=1,2,3, j=1,2,3; n=1,2 \)

**EKFNL(2)**: UPDG \( \rightarrow \) Update input transition matrix \( B(\dot{x}(k-l)) \):

The differences from FINDSCOMP:

\[
B(\dot{x}(k-l)) = \Delta \cdot T_{ER}(\dot{x}(k-l))
\]

and the only other difference from FINDSCOMP:

\( A = I \)

**EKFNL(1)**:

The differences from FINDSCOMP are the following measurement partials:

\[
H(\dot{x}(k-l)) = \begin{bmatrix}
1/\sigma_\phi & 0 & 0 \\
0 & 1/\sigma_\theta & 0 \\
0 & 0 & 1/\sigma_\psi \\
\end{bmatrix}
\]

**FINDS2**: (Translational Dynamics)

Number of states, \( NX = 8 \)

State vector, \( \dot{x} = [\dot{x}_x, \dot{y}_y, \dot{z}_z, \dot{f}_x, \dot{f}_y, \dot{f}_z, \dot{\delta}_x, \dot{\delta}_y]^T \)

Number of biases, \( NB = 3 \)

Bias vector, \( b = [\dot{b}_{ax}, \dot{b}_{ay}, \dot{b}_{az}]^T \)

Number of measurement types, \( NY = 4 \)

Measurement vector, \( y = [\text{MLS}_{az}, \text{MLS}_{et}, \text{MLSS}_{rn}, \text{IAS}]^T \)
Number of inputs, \( N_{Ul} = 3 \)

Input vector, \( u = [ax, ay, az]^T \)

---Start of New Time Tick: (time \( \cdot k \))---

\textbf{READFL:} Read the NFF input sensors \( u^n_\mathbf{i}(k) \), and the NFF measurement sensors, \( y^n_j(k) \); \( i=1,3 \); \( j=1-4 \); \( n=1-2 \)

\textbf{EKFN1(2):} UPDB \rightarrow Update input transition matrix \( B(\hat{x}(k-1)) \):

The differences from FINDSCMP:

\[
B(\hat{x}(k-1)) = \begin{bmatrix}
\frac{\Delta^2}{2} & \frac{\Delta^2}{2} & \Delta^2 \\
\Delta & T_{GB}(\hat{x}(k-1)) & \Delta I \\
0 & 0 & I
\end{bmatrix}
\]

where \( \phi(k-1), \theta(k-1) \) and \( \psi(k-1) \) in the evaluation of \( T_{GB}(\hat{x}(k-1)) \) are supplied by FINDS1.

\[
A = \begin{bmatrix}
I & \Delta I & 0 \\
0 & I & 0 \\
0 & 0 & A_w
\end{bmatrix}
\]

\textbf{EKFN1(1):} The difference from FINDSCMP: The rows corresponding to IMU measurements are deleted.
4. SUBPROGRAM FLOW CHARTS

This section of the User's Guide contains signal flow and processing diagrams of the key subprograms of FINDS. The figures have been arranged in a nested sequence of increasing level of detail. Wherever possible, a figure is supported by those next in sequence.
Figure 4.1: Flow Chart for Subprogram NAV

- enter
- compute jump parameters
- fd? on?
- no
- call: healr reconf(1)
- call: sumout
  ekfn1(2)
  blend(2)
  sumout
  ekfn1(1)
  biasf
  blend(1)
- yes
- fd? on?
- call: rescmp det01 det10 det05 reconf(-1)
- abort yes
- update heater window
- gto1
- exit

translational(FINS2) algorithm reads gravity vector from FINS1
rotational(FINS2) algorithm reads position and velocity estimates from FINS2
Figure 4.3: Flow Chart for Subprogram BLEND

Figure 4.4: Flow Chart for Subprogram ISOLAT
Figure 4.5: Flow Chart for Subprogram RECONF

Figure 4.5a: Partition A of Subprogram RECONF
Figure 4.5b: Partition B of RECONF

Figure 4.5c: Partition C of RECONF
Figure 4.5d: Partition D of RECONF

Figure 4.6: Hierarchical FDI Test
Figure 4.7: Isolation Logic

Figure 4.7a: BMA/IAS Failure Isolation
Figure 4.7b: MLS/IAS Failure Isolation

Figure 4.7c: RG Failure Isolation
Figure 4.7d: IMU Failure Isolation
5.  INPUT AND OUTPUT FILES

This section contains the descriptions of the input files required by and the output files generated by the FINDS program. In addition, the typical input design parameters are given in tables.

FINDS reads in the following files:

**ALGIN.DAT**
- detector thresholds 01, 05, 10 windows
- process noise SD
- measurement noise SD 01, 05, 10 windows
- wind model time contents

**RUNWAY.DAT**
- initial aircraft latitude, longitude position
- runway orientation relative to north
- elevation and azimuth/range MLS locations
- MLS and VOR antenna height above sea level

**FLDAT.NOF**
Flight data time history of the NFF input (rate gyro, accelerometer) and measurement (MLS, IAS, IMU) sensors, two replications each for a total of 26 channels of data per record.

Tables 5.1 and 5.2 depict typical values used as design parameters.

**Table 5.1: Design Values for No-Fail Filter Noise Parameters**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Noise S.D.</th>
<th>Replications Used</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Noises</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acc. Long.</td>
<td>0.05</td>
<td>1</td>
<td>m/s/s/s</td>
</tr>
<tr>
<td>Acc. Lat.</td>
<td>0.05</td>
<td>1</td>
<td>m/s/s</td>
</tr>
<tr>
<td>Acc. Vert.</td>
<td>0.05</td>
<td>1</td>
<td>m/s/s</td>
</tr>
<tr>
<td>Gyro Roll</td>
<td>0.05</td>
<td>1</td>
<td>deg/s</td>
</tr>
<tr>
<td>Gyro Pitch</td>
<td>0.05</td>
<td>1</td>
<td>deg/s</td>
</tr>
<tr>
<td>Gyro Yaw</td>
<td>0.05</td>
<td>1</td>
<td>deg/s</td>
</tr>
<tr>
<td>x-Wind-rw</td>
<td>0.10</td>
<td>N/A</td>
<td>m/s</td>
</tr>
<tr>
<td>y-Wind-rw</td>
<td>0.10</td>
<td>N/A</td>
<td>m/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measurement Noises</th>
<th>Noise S.D.</th>
<th>Replications Used</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLS Azim.</td>
<td>0.06</td>
<td>1</td>
<td>deg</td>
</tr>
<tr>
<td>MLS Elev.</td>
<td>0.06</td>
<td>1</td>
<td>deg</td>
</tr>
<tr>
<td>MLS Range</td>
<td>6.00</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>IAS</td>
<td>3.00</td>
<td>2</td>
<td>m/s</td>
</tr>
<tr>
<td>INS Roll</td>
<td>0.25</td>
<td>2</td>
<td>deg</td>
</tr>
<tr>
<td>INS Pitch</td>
<td>0.50</td>
<td>2</td>
<td>deg</td>
</tr>
<tr>
<td>INS Yaw</td>
<td>0.30</td>
<td>2</td>
<td>deg</td>
</tr>
</tbody>
</table>
Table 5.2 Detector Design Values for Measurement Sensor Noise Parameters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Noise S.D.</th>
<th>Replications</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLS Azim</td>
<td>3.00E-02</td>
<td>1</td>
<td>deg</td>
</tr>
<tr>
<td>Elev</td>
<td>3.50E-02</td>
<td>1</td>
<td>deg</td>
</tr>
<tr>
<td>Range</td>
<td>5.50E-00</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>IAS</td>
<td>2.00E-00</td>
<td>2</td>
<td>m/s</td>
</tr>
<tr>
<td>INS-Roll</td>
<td>1.30E-01</td>
<td>2</td>
<td>deg</td>
</tr>
<tr>
<td>Pitch</td>
<td>1.50E-01</td>
<td>2</td>
<td>deg</td>
</tr>
<tr>
<td>Yaw</td>
<td>5.00E-01</td>
<td>2</td>
<td>deg</td>
</tr>
</tbody>
</table>

The following files are written by the program during execution:

- **CHNGREP.DAT**: sensor failure data (index, replication, time) for post processing
- **RUNNEW.PLT**: time history of NFF states: position, velocity, attitude, and horizontal steady winds
- **RUNNEW.TLN**: summary of events during the course of execution
- **LRT01.PLT, LRT05.PLT, LRT10.PLT**: time history of likelihood ratio and measurement sensor residuals for detection windows 1, 5, and 10, respectively
- **EXPRES.PLT**: expanded residual time history for those sensors with replications (IAS, IMU)
- **GTOI.XFl**: time history of position and velocity states
- **SUMIN.UFl**: time history of gravity vector
- **IMU.XFl**: time history of attitude states

**Note:**

a) The partitioned algorithms FINDS1 and FINDS2 will read the same input as described above for FINDSCMP. In addition, GTOI.XFl is input for FINDS1 and IMU.XFl and SUMIN.UFl are both input for FINDS2
b) Both FINDS1 and FINDS2 write a subset of the output shown above for FINDSCMP according to the table shown:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>States</th>
<th>Residuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINDSCMP</td>
<td>position, velocity, attitude, wind,</td>
<td>MLS, IAS,</td>
</tr>
<tr>
<td></td>
<td>accelerometer bias, gyro bias</td>
<td>IMU</td>
</tr>
<tr>
<td>FINDS1</td>
<td>attitude, gyro bias</td>
<td>IMU</td>
</tr>
<tr>
<td>FINDS2</td>
<td>position, velocity, wind, accelerometer</td>
<td>MLS, IAS</td>
</tr>
<tr>
<td></td>
<td>bias</td>
<td></td>
</tr>
</tbody>
</table>
6. PROGRAM VARIABLE INDEXING TABLES

This section describes the array indexing convention used in the FINDS software. These tables include the following array variables: NFF state and measurement vectors, process noise input vector, and the measurement vector.
Table 6.1: NPP Absolute State Indexing Convention

Program Arrays: XFl

<table>
<thead>
<tr>
<th>Array Index</th>
<th>State Variable</th>
<th>Program Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINDSCMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$x_{rw}$</td>
<td>m</td>
</tr>
<tr>
<td>2</td>
<td>$y_{rw}$</td>
<td>m</td>
</tr>
<tr>
<td>3</td>
<td>$z_{rw}$</td>
<td>m</td>
</tr>
<tr>
<td>4</td>
<td>$\dot{x}_{rw}$</td>
<td>m/s</td>
</tr>
<tr>
<td>5</td>
<td>$\dot{y}_{rw}$</td>
<td>m/s</td>
</tr>
<tr>
<td>6</td>
<td>$\dot{z}_{rw}$</td>
<td>m/s</td>
</tr>
<tr>
<td>7</td>
<td>$\phi$</td>
<td>radians</td>
</tr>
<tr>
<td>8</td>
<td>$\theta$</td>
<td>radians</td>
</tr>
<tr>
<td>9</td>
<td>$\psi$</td>
<td>radians</td>
</tr>
<tr>
<td>10</td>
<td>$x_{w}$</td>
<td>m/s</td>
</tr>
<tr>
<td>11</td>
<td>$y_{w}$</td>
<td>m/s</td>
</tr>
<tr>
<td>FINDS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\phi$</td>
<td>radians</td>
</tr>
<tr>
<td>2</td>
<td>$\theta$</td>
<td>radians</td>
</tr>
<tr>
<td>3</td>
<td>$\psi$</td>
<td>radians</td>
</tr>
<tr>
<td>FINDS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$x_{rw}$</td>
<td>m</td>
</tr>
<tr>
<td>2</td>
<td>$y_{rw}$</td>
<td>m</td>
</tr>
<tr>
<td>3</td>
<td>$z_{rw}$</td>
<td>m</td>
</tr>
<tr>
<td>4</td>
<td>$\dot{x}_{rw}$</td>
<td>m/s</td>
</tr>
<tr>
<td>5</td>
<td>$\dot{y}_{rw}$</td>
<td>m/s</td>
</tr>
<tr>
<td>6</td>
<td>$\dot{z}_{rw}$</td>
<td>m/s</td>
</tr>
<tr>
<td>7</td>
<td>$x_{w}$</td>
<td>m/s</td>
</tr>
<tr>
<td>8</td>
<td>$y_{w}$</td>
<td>m/s</td>
</tr>
</tbody>
</table>
Table 6.2: NFF Absolute Measurement Indexing Convention

Program Arrays: RES80, RP1D01, RP1D05, RP1D10, YP1, YSCALE, INOYP, INOYPI, SIG (latter part), SIGD01 (latter), SIGD05 (latter), SIGD10 (latter), HXXP1

<table>
<thead>
<tr>
<th>Array Index</th>
<th>Measurement Name</th>
<th>Program Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINDSCMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MLS Azimuth</td>
<td>radians</td>
</tr>
<tr>
<td>2</td>
<td>MLS Elevation</td>
<td>radians</td>
</tr>
<tr>
<td>3</td>
<td>MLS Range</td>
<td>m</td>
</tr>
<tr>
<td>4</td>
<td>IAS</td>
<td>m/s</td>
</tr>
<tr>
<td>5</td>
<td>IMU Roll</td>
<td>radians</td>
</tr>
<tr>
<td>6</td>
<td>IMU Pitch</td>
<td>radians</td>
</tr>
<tr>
<td>7</td>
<td>IMU Yaw</td>
<td>radians</td>
</tr>
<tr>
<td>FINDS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>IMU Roll</td>
<td>radians</td>
</tr>
<tr>
<td>2</td>
<td>IMU Pitch</td>
<td>radians</td>
</tr>
<tr>
<td>3</td>
<td>IMU Yaw</td>
<td>radians</td>
</tr>
<tr>
<td>FINDS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MLS Azimuth</td>
<td>radians</td>
</tr>
<tr>
<td>2</td>
<td>MLS Elevation</td>
<td>radians</td>
</tr>
<tr>
<td>3</td>
<td>MLS Range</td>
<td>m</td>
</tr>
<tr>
<td>4</td>
<td>IAS</td>
<td>m/s</td>
</tr>
</tbody>
</table>
Table 6.3: NPP Absolute Input Indexing Convention

Program Arrays: UP1, INDUP, XBFO

<table>
<thead>
<tr>
<th>Array Index</th>
<th>Input Name</th>
<th>Program Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINDSCMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$a_x$</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>2</td>
<td>$a_y$</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>3</td>
<td>$a_z$</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>4</td>
<td>p</td>
<td>radians/s</td>
</tr>
<tr>
<td>5</td>
<td>q</td>
<td>radians/s</td>
</tr>
<tr>
<td>6</td>
<td>r</td>
<td>radians/s</td>
</tr>
<tr>
<td>FINDS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>p</td>
<td>radians/s</td>
</tr>
<tr>
<td>2</td>
<td>q</td>
<td>radians/s</td>
</tr>
<tr>
<td>3</td>
<td>r</td>
<td>radians/s</td>
</tr>
<tr>
<td>FINDS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$a_x$</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>2</td>
<td>$a_y$</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>3</td>
<td>$a_z$</td>
<td>m/s$^2$</td>
</tr>
</tbody>
</table>
Table 6.4: NFF Process Noise Indexing Convention

Program Arrays: QFl, SIG (former part), SIGD01 (former), SIGD05, (former), SPGD10 (former)

<table>
<thead>
<tr>
<th>Array Index</th>
<th>Name</th>
<th>Program Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINDSCMP</td>
<td>a_x</td>
<td>m/s^2</td>
</tr>
<tr>
<td></td>
<td>a_y</td>
<td>m/s^2</td>
</tr>
<tr>
<td></td>
<td>a_z</td>
<td>m/s^2</td>
</tr>
<tr>
<td></td>
<td>p</td>
<td>radians/s</td>
</tr>
<tr>
<td></td>
<td>q</td>
<td>radians/s</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>radians/s</td>
</tr>
<tr>
<td></td>
<td>x_w</td>
<td>m/s</td>
</tr>
<tr>
<td></td>
<td>y_w</td>
<td>m/s</td>
</tr>
<tr>
<td>FINDS1</td>
<td>p</td>
<td>radians/s</td>
</tr>
<tr>
<td></td>
<td>q</td>
<td>radians/s</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>radians/s</td>
</tr>
<tr>
<td>FINDS2</td>
<td>a_x</td>
<td>m/s^2</td>
</tr>
<tr>
<td></td>
<td>a_y</td>
<td>m/s^2</td>
</tr>
<tr>
<td></td>
<td>a_z</td>
<td>m/s^2</td>
</tr>
<tr>
<td></td>
<td>x_w</td>
<td>m/s</td>
</tr>
<tr>
<td></td>
<td>y_w</td>
<td>m/s</td>
</tr>
<tr>
<td>Array Index</td>
<td>Sensor Type</td>
<td>Program Units</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>FINDSCMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$a_x$</td>
<td>m/s²</td>
</tr>
<tr>
<td>2</td>
<td>$a_y$</td>
<td>m/s²</td>
</tr>
<tr>
<td>3</td>
<td>$a_z$</td>
<td>m/s²</td>
</tr>
<tr>
<td>4</td>
<td>$p$</td>
<td>radians/s</td>
</tr>
<tr>
<td>5</td>
<td>$q$</td>
<td>radians/s</td>
</tr>
<tr>
<td>6</td>
<td>$r$</td>
<td>radians/s</td>
</tr>
<tr>
<td>7</td>
<td>MLS Azimuth</td>
<td>radians</td>
</tr>
<tr>
<td>8</td>
<td>MLS Elevation</td>
<td>radians</td>
</tr>
<tr>
<td>9</td>
<td>MLS Range</td>
<td>m</td>
</tr>
<tr>
<td>10</td>
<td>IAS</td>
<td>m/s</td>
</tr>
<tr>
<td>11</td>
<td>IMU $\phi$</td>
<td>radians</td>
</tr>
<tr>
<td>12</td>
<td>IMU $\theta$</td>
<td>radians</td>
</tr>
<tr>
<td>13</td>
<td>IMU $\psi$</td>
<td>radians</td>
</tr>
<tr>
<td>FINDS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$p$</td>
<td>radians/s</td>
</tr>
<tr>
<td>2</td>
<td>$q$</td>
<td>radians/s</td>
</tr>
<tr>
<td>3</td>
<td>$r$</td>
<td>radians/s</td>
</tr>
<tr>
<td>4</td>
<td>IMU $\phi$</td>
<td>radians</td>
</tr>
<tr>
<td>5</td>
<td>IMU $\theta$</td>
<td>radians</td>
</tr>
<tr>
<td>6</td>
<td>IMU $\psi$</td>
<td>radians</td>
</tr>
<tr>
<td>FINDS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$a_x$</td>
<td>m/s²</td>
</tr>
<tr>
<td>2</td>
<td>$a_y$</td>
<td>m/s²</td>
</tr>
<tr>
<td>3</td>
<td>$a_z$</td>
<td>m/s²</td>
</tr>
<tr>
<td>4</td>
<td>MLS Azimuth</td>
<td>radians</td>
</tr>
<tr>
<td>5</td>
<td>MLS Elevation</td>
<td>radians</td>
</tr>
<tr>
<td>6</td>
<td>MLS Range</td>
<td>radians</td>
</tr>
<tr>
<td>7</td>
<td>IAS</td>
<td>m/s</td>
</tr>
</tbody>
</table>
Table 6.6: Replicated Sensor Indexing Convention

Program Arrays: XBFI, PBFI, RESBI, CBFI, ICNTSN, PRIORI, ALAMDA

<table>
<thead>
<tr>
<th>Array Index</th>
<th>Sensor Type/Repl.</th>
<th>Program Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINDSCMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$a_x$-n*</td>
<td>m/s</td>
</tr>
<tr>
<td>2</td>
<td>$a_y$-n*</td>
<td>m/s</td>
</tr>
<tr>
<td>3</td>
<td>$a_z$-n*</td>
<td>m/s</td>
</tr>
<tr>
<td>4</td>
<td>$p$-n*</td>
<td>radians/s</td>
</tr>
<tr>
<td>5</td>
<td>$q$-n*</td>
<td>radians/s</td>
</tr>
<tr>
<td>6</td>
<td>$r$-n*</td>
<td>radians/s</td>
</tr>
<tr>
<td>7</td>
<td>MLS Azim-n*</td>
<td>radians</td>
</tr>
<tr>
<td>8</td>
<td>MLS Elev-q</td>
<td>radians</td>
</tr>
<tr>
<td>9</td>
<td>MLS Rng-n</td>
<td>m</td>
</tr>
<tr>
<td>10</td>
<td>IAS-1</td>
<td>m/s</td>
</tr>
<tr>
<td>11</td>
<td>IMU $\phi$-1</td>
<td>radians</td>
</tr>
<tr>
<td>12</td>
<td>IMU $\theta$-1</td>
<td>radians</td>
</tr>
<tr>
<td>13</td>
<td>IMU $\psi$-1</td>
<td>radians</td>
</tr>
<tr>
<td>14</td>
<td>IAS-2</td>
<td>m/s</td>
</tr>
<tr>
<td>15</td>
<td>IMU $\phi$-2</td>
<td>radians</td>
</tr>
<tr>
<td>16</td>
<td>IMU $\theta$-2</td>
<td>radians</td>
</tr>
<tr>
<td>17</td>
<td>IMU $\psi$-2</td>
<td>radians</td>
</tr>
<tr>
<td>FINDS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$p$-n*</td>
<td>radians/s</td>
</tr>
<tr>
<td>2</td>
<td>$q$-n*</td>
<td>radians/s</td>
</tr>
<tr>
<td>3</td>
<td>$r$-n</td>
<td>radians</td>
</tr>
<tr>
<td>4</td>
<td>IMU $\phi$-1</td>
<td>radians</td>
</tr>
<tr>
<td>5</td>
<td>IMU $\theta$-1</td>
<td>radians</td>
</tr>
<tr>
<td>6</td>
<td>IMU $\psi$-1</td>
<td>radians</td>
</tr>
<tr>
<td>7</td>
<td>IMU $\phi$-2</td>
<td>radians</td>
</tr>
<tr>
<td>8</td>
<td>IMU $\theta$-2</td>
<td>radians</td>
</tr>
<tr>
<td>9</td>
<td>IMU $\psi$-2</td>
<td>radians</td>
</tr>
<tr>
<td>FINDS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$a_x$-n</td>
<td>m/s</td>
</tr>
<tr>
<td>2</td>
<td>$a_y$-n</td>
<td>m/s</td>
</tr>
<tr>
<td>3</td>
<td>$a_z$-n</td>
<td>m/s</td>
</tr>
<tr>
<td>4</td>
<td>MLS Azim-n</td>
<td>radians</td>
</tr>
<tr>
<td>5</td>
<td>MLS Elev-n</td>
<td>radians</td>
</tr>
<tr>
<td>6</td>
<td>MLS Rng-n</td>
<td>m</td>
</tr>
<tr>
<td>7</td>
<td>IAS-1</td>
<td>m/s</td>
</tr>
<tr>
<td>8</td>
<td>IAS-2</td>
<td>m/s</td>
</tr>
</tbody>
</table>


*n* refers to the replication currently in use by the NFF (i.e., 1 or 2)
<table>
<thead>
<tr>
<th>Array Index</th>
<th>Meas. Sensor Type/Rep1.</th>
<th>Program Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINDSCMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MLS Azim-n$_a$</td>
<td>radians</td>
</tr>
</tbody>
</table>
| 2           | MLS Elev-$
$            | radians       |
| 3           | MLS Rng-n               | m             |
| 4           | IAS-1                   | m/s           |
| 5           | IMU $\phi$-1           | radians       |
| 6           | IMU $\theta$-1         | radians       |
| 7           | IMU $\psi$-1           | radians       |
| 8           | IAS-2                   | m/s           |
| 9           | IMU $\phi$-2           | radians       |
| 10          | IMU $\theta$-2         | radians       |
| 11          | IMU $\psi$-2           | radians       |
| FINDS1      |                         |               |
| 1           | IMU $\phi$-1           | radians       |
| 2           | IMU $\theta$-1         | radians       |
| 3           | IMU $\psi$-1           | radians       |
| 4           | IMU $\phi$-2           | radians       |
| 5           | IMU $\theta$-2         | radians       |
| 6           | IMU $\psi$-2           | radians       |
| FINDS2      |                         |               |
| 1           | MLS Azim-n$_a$          | radians       |
| 2           | MLS Elev-$
$            | radians       |
| 3           | MLS Rng-n               | m             |
| 4           | IAS-1                   | m/s           |
| 5           | IAS-2                   | m/s           |

n refers to the replication currently in use by the NFF (i.e., 1 or 2)
This section contains a description of all subprograms in FINDS. Table 7.1 is a "quick" reference list of each subprogram and its associated "calls to" and "called by" programs. Subsequent paragraphs explain the specific function of each subprogram and list its associated common blocks.

### Table 7.1

#### Subprograms

<table>
<thead>
<tr>
<th>Called by:</th>
<th>Name</th>
<th>Calls to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main program (FINDS/FINDS1/FINDS2)</td>
<td>READFL</td>
<td>HEALR, RECONF, SUMIN, EKFNI, BLEND, SUMOUT, BIA SF, RESCMP, DETO1, DETO5, DET10, GTO1</td>
</tr>
<tr>
<td>Main program</td>
<td>NAV</td>
<td>BUBBL2, INITXF, UPDB, VEQUAL, GTO1</td>
</tr>
<tr>
<td>Main program</td>
<td>INITG</td>
<td></td>
</tr>
<tr>
<td>INITG</td>
<td>INITXF</td>
<td></td>
</tr>
<tr>
<td>NAV</td>
<td>SUMIN</td>
<td></td>
</tr>
<tr>
<td>NAV</td>
<td>SUMOUT</td>
<td></td>
</tr>
<tr>
<td>NAV, INITG</td>
<td>GTO1</td>
<td></td>
</tr>
<tr>
<td>NAV</td>
<td>EKFNI</td>
<td>UPDPH, PDMINV, MATIA, MAT3 VSCALE, MATS, MADD, UPDB, UPDQ, PD3NV1, PMA XB, PMABT, PMABT2, PMAPB, PD4NV1</td>
</tr>
<tr>
<td>NAV</td>
<td>BIA SF</td>
<td>VSUB, MEQUAL, MAT1A, MATVAC, VSCALE, MSUB, MATXYT, MADD, PDMINV, PMBEA, PMA XB, PMAXV, YSCALE, PMAMB, PMABT, PMAPB, PD3NV1, PD4NV1</td>
</tr>
<tr>
<td>NAV</td>
<td>BLEND</td>
<td>MAT1A, MADD, MATVC2, UPDH, PMA XB, PMAPB, PMAXV2</td>
</tr>
<tr>
<td>NAV</td>
<td>DETO1</td>
<td>MEQUAL, MAT3B, ISOLAT, PMBEA, PMVTAV</td>
</tr>
<tr>
<td>NAV</td>
<td>DETO5</td>
<td>MEQUAL, MAT3B, ISOLAT, PMBEA, PMVTAV</td>
</tr>
<tr>
<td>NAV</td>
<td>DET10</td>
<td>MEQUAL, MAT3B, ISOLAT, PMBEA, PMVTAV</td>
</tr>
<tr>
<td>RECONF</td>
<td>SETISN</td>
<td></td>
</tr>
<tr>
<td>INITG, EKFNI</td>
<td>UPDB</td>
<td></td>
</tr>
<tr>
<td>EKFNI</td>
<td>UPDQ</td>
<td></td>
</tr>
<tr>
<td>BLEND</td>
<td>UPDH</td>
<td></td>
</tr>
<tr>
<td>CLIPSIO, EKFNI</td>
<td>RESCMP</td>
<td></td>
</tr>
<tr>
<td>NAV</td>
<td>UPDPH</td>
<td></td>
</tr>
<tr>
<td>DETO1, DETO5, DET10</td>
<td>ISOLAT</td>
<td>MAT1A, PDMINV, VEQUAL, VSUB, VADD, LKF, DECIDE, MEQUAL, TRANS2, MATXYT, MADD, MADZ, MSUB, MATLN2, MATVEC, LRT</td>
</tr>
<tr>
<td>Called by:</td>
<td>Name</td>
<td>Calls to:</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>ISOLAT</td>
<td>LKF</td>
<td>MAT3B</td>
</tr>
<tr>
<td>ISOLAT</td>
<td>LRT</td>
<td>VEQUAL, TLOUT, VMPRT, VMPRT2, MINIM2, MINIM3</td>
</tr>
<tr>
<td>ISOLAT</td>
<td>DECIDE</td>
<td>PNTINV, RCOV, SETLSN, CLPS10, TLOUT, IMEG2, NOISR, MATLN2, RCOV, PNTINV, IMTCG2, CLPSBE, NOISR, UPDPH</td>
</tr>
<tr>
<td>NAV</td>
<td>RECONF</td>
<td>CLPSIO</td>
</tr>
<tr>
<td>RECONF</td>
<td>CLPSIO</td>
<td>NOISR</td>
</tr>
<tr>
<td>RECONF, CLPS10</td>
<td>CLPSBE</td>
<td>ADJTPB, MATCG2, PNT1NV, IMTCG2, MATCG3</td>
</tr>
<tr>
<td>CLPS10</td>
<td>ADJTBP</td>
<td>PNT1NV, IMTCG2</td>
</tr>
<tr>
<td>CLPSBE</td>
<td>RCOV</td>
<td>VMPRT, MATLN2, VMPRT2, MATLN3</td>
</tr>
<tr>
<td>CLPS10, RECONF</td>
<td>HEALR</td>
<td>BUBBL2, TLOUT, LRTHLR</td>
</tr>
<tr>
<td>NAV</td>
<td>LRTHLR</td>
<td></td>
</tr>
<tr>
<td>Main Program, DECIDE, RECONF, HEALR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Includes files 'FINDSCMP.FOR', 'FINDSI.FOR', 'FINDS2.FOR'.

**NOTE:**
(a) The exact-dimensioned versions of FINDSI & FINDS2 are summarized here. The documentation for the 'NDIM' dimensioned versions is along the same lines as for FINDSCMP.

(b) Everything is common to all 3 files except where specified by filename.

(c) Notation in this document is as follows:
- **func** --- function (of routine)
- **refs** --- refers (other routines it refers to)
- **refby** --- referred by (other routines it gets called by)
- **comm** --- common blocks (used in the routine)
- **args** --- variables in the argument list

### I. DESCRIPTION OF SUBROUTINES

**name:** FINDS/FINDSI/FINDS - (Main Program)
**func:** Coordinates the run-time operation of the FINDS algorithm. FINDS1 is the rotational kinematics portion and FINDS2 is the translational dynamics portion of the composite algorithm. Initializes program variables, reads-in first iteration of flight data and initializes the filter. The basic run-time loop consists of reading in one iteration of flight data (READFL) and passing control to NAV which coordinates the FTN/FDI algorithm.
**refs:** INITG, READFL, TLOUT, NAV
**comm:** FINDSCMP --- EARTH, MCONCO, SYNC, IMLS, MLSALL, PSIR, CNTROL, ABRTCM
FINDS1 --- EARTH, MCONCO, SYNC, IMLS, PSIR, CNTROL, ABRTCM
FINDS2 --- SYNC, MLSALL, CNTROL, ABRTCM

**name:** READFL
**func:** Flight data interface routine -- reads in the flight data from binary data file, assigns data to the various sensor variables and converts data to program working units (i.e., radians, m, m/s, m/s²). Also checks for data dropouts and "fixes" them by substituting data from previous iteration.
**call:** Call READFL
**args:** None
**refs:** None
**refby:** FINDS/FINDSI/FINDS2
**comm:** FINDSCMP --- SYNC, MCONCO, RGOUT, LAOUT, AGOUT, ASOUT, MLOUT, NAMES, RDLOCL
FINDS1 --- SYNC, MCONCO, RGOUT, AGOUT, ASOUT, MLOUT, NAMES, RDLOCL
FINDS2 --- SYNC, MCONCO, LAOUT, ASOUT, MLOUT, FLTIN, NAMES, RDLOCL

**name:** NAV
**func:** Executive program which coordinates the no-fail filter (NFF) (or fault tolerant navigator FTN) and failure detection (isolation (FDI) modules, (see attached flow chart)
**call:** Call NAV
**args:** None
refs: HEALR, RECONF, SUMIN, EKFN1, BLEND, SUMOUT, BIASF, RESCMP, DET01, DET10, DET05, GTOI (note: FINDS2 does not contain routine GTOI)
refby: FINDS/FINDS1/FINDS2
comm: SYNC, CNTROL, ABRTCM, EKF1, HEALCM, SYSXBO, JUMPCM, DTSYNC, HPFCM

name: INITG
func: Sets program flags and initializes parameters used in the NFF, FDI and reconfiguration modules. The initialization process is in two passes; the first pass configures the system dimensions based on sensor replications used and also sets the healer parameters. The second pass sets the initial conditions for the NFF states and initializes the NFF measurement and covariances.
call: Call INITG
args: None
refs: FINDSCMP --> BUBBL2, VEQUAL, INITXF, UPDB, GTOI
FINDS1 --> BUBBL2, INITXF, UPDB, GTOI
FINDS2 --> BUBBL2, INITXF, UPDB
refby: FINDS/FINDS1/FINDS2
comm: SYSX1, SYSYW, SYSU1, EKF1, EKBF0, SYSXBO, CMPSTF, DETXBD, SYNC, MCONCO, FILTRT, INITVL, DETINF, CNTROL, FILTIC, YOBSRV, MAINT, HEALCM
In addition, FINDSCMP/FINDS2 contain blocks SIGTAU, ASOUT
FINDSI contains block SIG

name: INITXF
func: Uses the first iteration of the flight data to compute the NFF state initial conditions. A/C position is calculated using a reconstruction algorithm from the MLS measurements. Velocity is estimated by resolving the averaged IAS measurement in the appropriate axis. A/C attitude initial estimates are obtained by averaging the replicated IMU measurements. Initial horizontal winds are estimated to be zero.
call: Call INITXF
args: None
refs: None
refby: INITG
comm: FINDSCMP --> FILTRT, MCONCO, EKF1, ASOUT, MLSALL, AGOUT, MLOUT, PSIR
FINDS1 --> FILTRT, AGOUT PSIR, EKF1
FINDS2 --> FLTIN, MLOUT, ASOUT, MLSALL, MCONCO, FILTRT, EKF1

name: SUMIN
func: Provides a proper set of inputs to the NFF. The input vector is formed as follows:
1) Only on replication of all input sensors is in active mode; the second replication is kept either in standby or in failed status.
2) the input vector, UFl, is formed such that trapezoidal integration is performed, i.e., U(k) = 0.5 \* [u(k) + u(k-1)]
3) current estimates of input sensor biases (XBFO) are subtracted from UFl.
4) FINDSCMP, FINDS1 --> rate gyro measurements are compensated for earth and platform rates
5) FINDSCMP, FINDS2 --> the gravity vector (Gx, Gy, Gz) expressed in the G-frame is added to the end of UFl.
6) FINDSCMP --> UFl = [Ax, Ay, Az, P, Q, R, Gx, Gy, Gz]

- 46 -
FINDS1 \rightarrow UFI = [P, Q, R]^T
FINDS2 \rightarrow UFI = [Ax, Ay, Az, Gx, Gy, Gz]^T

7) In the split versions, FINDS1 generates the gravity vector in GTOI which is then transferred over to FINDS2 and used there.

call: Call SUMIN
args: None
refs: None
refby: NAV
comm: FINDSCMP \rightarrow MAIN1, RGOUT, LAOUT, EKBF0, SYSUI, SYSXBO, FILTRT, SYNC, EARTH, PSIR, TRBER, LATLON, SUMLOC

name: SUMOUT
func: Forms a set of measurements (YFl) to be used by the NFF
1) each sensor replication has an active or failed or standby status, and the number of available active replicated measurements are averaged
2) each measurement is normalized by the expected variance of that signal (scale factor is set in INITG)
3) psi measurements are compensated for runway yaw in FINDSCMP and FINDS1
4) FINDSCMP \rightarrow YFI = [Azim, Elev, Rng, IAS, Phi, Theta, Psi]^T
   FINDS1 \rightarrow YFI = [Phi, Theta, Psi]^T
   FINDS2 \rightarrow YF2 = [Azim, Elev, Rng, IAS]^T

call: Call SUMOUT
args: None
refs: None
refby: NAV
comm: FINDSCMP \rightarrow PSIR, ASOUT, AGOUT, MLOUT, SYSW1, FILTRT, YOBSRV, DETXBI
   FINDS1 \rightarrow PSIR, AGOUT, SYSW1, FILTRT, YOBSRV, DETXBI
   FINDS2 \rightarrow ASOUT, MLOUT, SYSW1, FILTRT, YOBSRV, DETXBI

name: GTOI (not in FINDS2)
func: Forms estimates for inertial position, velocity and acceleration, and runway acceleration. Also computes the a/c's current longitude and latitude along with their rates of change. In addition, coriolis and centripetal correction terms for compensating the platform gravity force are also computed. [NOTE: in FINDS1, this routine needs the a/c position and velocity estimates generated by FINDS2]

call: Call GTOI
args: None
refs: None
refby: NAV, INITG
comm: MAIN1, FILTRT, RGOUT, SYSUI, EKF1, TRBER, MCONCO, EARTH, IMLS, PSIR, LATLON, PQRDEG, GRVYTC, GTOILC

name: EKFN1
func: Represents the bias-free filter portion of the NFF and is implemented as an extended Kalman filter (EKF). Covariance propagation of the stabilized normal equations is performed. The state estimates, XFl, are not computed in this routine. (see attached flow chart)

call: Call EKFN1 (Iup)

- 47 -
args: Iup -- integer in ; update/proagate flag (1 ==> update, 2 ==> propagate)

refs: FINDS_CMP --> UPDPH, PMINV, MAT1A, MAT3, VSCALE, MAT2, MADD, UPDB, UPDQ
      FINDS_1 --> PD3NV1, PMAKB, PMABAT, VSCALE, PMABT2, PMAKB, UPDB, UPDQ
      FINDS_2 --> UPDPH, PD4NV1, PMAKB, PMABAT, VSCALE, PMABT2, PMAKB, UPDB, UPDQ

refby: NAV

comm: FINDS_CMP --> MAIN2, SYSX1, SYSY1, SYSU1, EKFL, SYXBO, SYSYB0, SYSXBO, FILTER, TSTORE, CTRL, EKFBIA, JUMPCM
      FINDS_1 --> MAIN2, SYSX1, SYSY1, SYSU1, EKFL, SYXBO, SYSYB0, FILTER, CTRL, EKFBIA, JUMPCM
      FINDS_2 --> SYSX1, SYSY1, SYSU1, EKFL, SYXBO, FILTER, CTRL, EKFBIA, JUMPCM
            JUMPCM, EKFLN, EKFWRK

name: BIASF

func: Implements the bias filter portion of the NFF. There are no bias filter dynamics; hence no propagation step is required and this routine is called only during the update mode of the NFF.

call: Call BIASF

args: None

refs: FINDS_CMP --> VSUB, MEQUAL, MAT1A, MATVEC, VSCALE, MSUB, MATXYT, ADD, PMINV
      FINDS_1 --> VSUB, PMBEA, PMAKB, PMAKV, VSCALE, PMAMB, PMABT, PMAKB, PD3NV1
      FINDS_2 --> VSUB, PMBEA, PMAKB, PMAKV, VSCALE, PMAMB, PMABT, PMAKB, PD4NV1

refby: NAV

comm: MAIN1, SYSX1, SYSY1, SYSU1, EKFL, SYXBO, GBLEND, YOBSRV, FILTER, EKFL, EKFBIA, LRTINV, DETCOV, JUMPCM, CTRL
      In addition to the above, FINDS_CMP --> MAIN2, SYSYBO, TSTORE
      FINDS_1 --> MAIN2, SYSYBO, BSFWRK
      FINDS_2 --> BSFWRK

name: BLEND

func: Computes the bias and bias-free state estimates and "blends" them together to form the total state and bias estimates. Also forms the Kalman gain matrix. (see flow chart)

call: Call BLEND (Iup)

args: Iup -- integer in ; update/proagate flag (1 ==> update, 2 ==> propagate)

refs: FINDS_CMP --> MAT1A, MADD, MATVC2, UPDPH
      FINDS_1, FINDS_2 --> PMAKB, PMAKB, PMAKB, UPDPH

refby: NAV

comm: SYSX1, SYSY1, SYSU1, EKFL, EKBFO, SYXBO, GBLEND, CMPSTF, DETINF, FILTER, JUMPCM
      In addition to the above, FINDS_CMP --> MAIN2, TSTORE
      FINDS_1 --> MAIN2
      FINDS_2 --> EKFBIA, BLNDWK
name: DET01
func: Implements the failure detector of moving residual window length 1 sample, i.e., the current filter residual. Performs a Chi-square test on the NPP averaged measurement residual RESBO and checks against set thresholds to detect failures. Calls isolation routine ISOLAT if failure is detected.
call: Call DET01
args: None
refs: FINDSCMP --> MEQUAL, MAT3B, ISOLAT
       FINDS1, FINDS2 --> PMBEA, PMVTAV, ISOLAT
refby: NAV
comm: SYNC, SYSW1, SYSU1, FILTRT, EKBFO, LRTINV, JUMPCM, LRTMAX, DTCT01, CNTROL, DETPRI

name: DET05
func: Implements the failure detector of moving residual window length 5 samples. Performs a Chi-square test on the moving average of RESBO over the last 5 samples (incl. current residual).
call: Call DET05
args: None
refs: FINDSCMP --> MEQUAL, MAT3B, ISOLAT
       FINDS1, FINDS2 --> PMBEA, PMVTAV, ISOLAT
refby: NAV
comm: SYNC, SYSW1, SYSU1, FILTRT, EKBFO, LRTINV, JUMPCM, LRTMAX, DTCT05, CNTROL, DETPRI

name: DET10
func: Implements the failure detector of moving residual window length 10 samples. Performs a Chi-square test on the moving average of RESBO over the last 10 samples (incl. current residual).
call: Call DET10
args: None
refs: FINDSCMP --> MEQUAL, MAT3B, ISOLAT
       FINDS1, FINDS2 --> PMBEA, PMVTAV, ISOLAT
refby: NAV
comm: SYNC, SYSW1, SYSU1, FILTRT, EKBFO, LRTINV, JUMPCM, LRTMAX, DTCT10, CNTROL, DETPRI

name: SETISN
func: Maintains the value of vector ICNTSN in which the ordering of elements corresponds to the absolute replicated sensor ordering (Table 6.6). The value of each element is the location in UPl for the input elements (six for FINDSCMP, 3 for FINDS1/FINDS2), and the location in the expanded innovations for the rest of ICNTSN. ICNTSN provides a mapping between an absolute indexing scheme and a collapsed indexing scheme in the event of failures.
call: Call SETISN
args: None
refs: None
refby: RECONF
comm: DETINF, FILTRT, SYSU1, DETXBI
name: UPDB
func: Updates the discrete input weighting matrix BF1 and also evaluates and saves:
1) sines and cosines of the estimated Euler angles (in FINDS2, these are the estimates transferred over from FINDS1 at each iteration).
2) the transformation from the B to the R frame
3) the transformation from the R to the E frame (not in FINDS2).
call: Call UPDB
args: None
refs: None
refby: INITG, EKFN1
comm: MAIN1, TRBER, EULER, SYNC, SYSU1, EKF1, SYSX1

name: UPDQ
func: Updates the discrete process noise covariance matrix EF1. Assumes that UPDB has been called before this routine, hence transformation matrices Trb and Ter are current. In addition, for FINDSCMP and FINDS1, terms to represent the rate gyro errors due to scale factor and misalignment are added to the measurement noise variance.
call: Call UPDQ
args: None
refs: None
refby: EKFN1
comm: MAIN1, TRBER, SYNC, MCONCO, SYSX1, SYSYW1, UPDQLC
In addition to the above, FINDSCMP --> SIGTAU, PQRDEG
FINDS1 --> SIG, PQRDEG
FINDS2 --> SIGTAU

name: UPDH
func: Updates the nonlinear observations function H, called HXKPl
call: Call UPDH
args: None
refs: None
refby: BLEND
comm: YOBRSV, SYSX1, SYSYW1, EKFL, SYSUL, EKBFO, SYSXBO
In addition, FINDSCMP, FINDS2 --> MLSALL

name: UPDPH (not in FINDS1)
func: Updates the partial of H (i.e., HXKPl) w.r.t. XFl, called HPl. Not used in FINDS1 as HPl is an identity matrix in that algorithm.
call: Call UPDPH
args: None
refs: None
refby: EKFN1, CLPS1O
comm: MAIN1, MLSALL, YOBRSV, SYSXBO, SYSUL, SYSYW1, CMPSTF, SYSX1, EKFL

name: RESCMP
func: Computes the expanded residuals sequence (RESB0C) from the residual sequence (RESB0) generated by the NFF. This sequence is the same as
the one which would have been generated had the filter been driven by
all replications of the measurement sensors rather than their average
value. This expanded residuals sequence is used in the failure
isolation strategy.

call: Call RESCMP
args: None
refs: None
refby: NAV
comm: EKF1, YOBSRV, SYSYW1, DETINF, FILTRT, SYSU1, DTSYNC

In addition,
FINDSCMP --- ASOUT, AGOUT, MLOUT, PSIR
FINDS1 --- ASOUT, PSIR
FINDS2 --- ASOUT, MLOUT

name: ISOLAT
func: Implements a bank of first order filters and likelihood ratio
computers in the isolation strategy. Each filter hypothesizes the
occurrence of a failure at the beginning of the residual window
(based on the length of the detector sequence which flagged the
failure), and estimates the level of a bias jump failure by observing
the expanded (and saved) residuals sequence over that window. The
hypothesized failure is assumed to affect the NFF input measurements
or output measurements only. Thus, a single failure cannot directly
enter into BOTH an input and an output measurement.

A select subset of all first order filters is activated depending on
which detector caught the failure. If the detector of window length
1 sample (DET01) signals the failure, then only the output sensor
filters are activated. Similarly, if DET10 flags the failure, then
only the input sensors (and the IAS sensor in the case of
FINDSCMP/FINDS2) filters are activated. For DET05, no such
assumptions are made and all of the sensors are equally "suspect."

The first order filters generate a sequence of failure compensated
residuals which are used by the bank of likelihood ratio computers to
compute the log likelihood of a singleton sensor failure (or a dual
simultaneous failure in MLS sensors).

Subroutine ISOLAT functions as an executive of this bank of filter/LR
computers. In the current version of FINDSCMP/FINDS2, only one
replication of the MLS sensors is kept active and the other is in
standby status (like the input sensors); hence, dual simultaneous MLS
sensor failures are not considered in this routine or in DECIDE.
(see attached flowchart)

call: Call ISOLAT (Ifilwin)
args: Ifilwin -- integer in ; length of detector window which flagged the
      failure (has value of either 1 or 5 or 10)
refs: MAT1A, PDMINV, VEQUAL, VSUB, VADD, LKF, DECIDE
      In addition,
      FINDSCMP --- MEQUAL, TRANS2, MATXYT, MADD, MATZ, MSUB, MATNL2,
               MATVEC, LRT
refby: DET01, DET05, DET10
comm: MAIN1, SYSX1, SYSYW1, SYSU1, SYSXBO, YOBSRV, EKF1, EDBFO, CMPSTF,
      DETXBI, DETINF, DCIDEI, DETYBI, INITVL, FILTRT, DTSYNC, DETCOV,
      DETLC3
In addition,
FINDSCMP → MAIN2, TSTORE, MULTDT, DETLC2
FINDS1 → MAIN2, DETWRK
FINDS2 → MULTDT, DETWRK, DETLC2

name: LKF
func: Provides the failure estimator structure in the isolation strategy. Implements a linear Kalman filter using the information form, and assumes a scalar state equation. The plant, measurements and filter equations are commented in the actual code in each algorithm. Generates a set of failure compensated residuals and also a "best" estimate of failure level for each suspect sensor.
call: Call LKF (Index, Ci, Istart)
args: Index -- Integer in ; points to particular sensor in question (has value based on Table 6.6 indexing)
Ci -- real in ; effective observations matrix (computed in ISOLAT)
Istart -- integer ; location in saved, expanded residual sequence RESBOC (has value between 1 and 10 depending on current location and Iflwin)
refs: None
refby: ISOLAT
comm: MAIN1, DETINF, DETXBI, DETYBI, DETLC3

name: LRT (only in FINDSCMP)
func: Computes the log likelihood ratios in the isolation strategy. The computations are as follows:
1) if loop = 1, A = -PHj. This initializes the log likelihood ratio A to -ln(PHj) at the start of the detection/decision residual window.
2) SUMI = RES * RTinv * RES
3) A = 0.5 * SUMI + A)
args: Loop -- integer in ; detection/decision window step (has values from 1 to Iflwin)
PHj --> real in ; log of a-priori probability that the j'th sensor will fail
RES --> real in ; failure corrected innovations sequence from the j'th LKF
A --> real in out ; computed value of log likelihood ratio for j'th failure hypothesis.
refs: MAT3B
refby: ISOLAT
comm: MAIN1, DETINF, DETLC3

name: DECIDE
func: Chooses the most likely failure hypothesis by finding the smallest log likelihood ratio of those computed in LRT. For a chosen hypothesis, it checks for a minimum acceptable failure level, else chooses the next likely hypothesis. Also, prints out various user messages.
call: Call DECIDE (Iflwin)
args: Iflwin -- integer in ; length of detector window which flagged the failure (either 1 or 5 or 10)
refs: FINDSCMP → VEQUAL, MINIM2, TLOUT, VMPRT
FINDS1 ——→ VEQUAL, MINIM3, TLOUT, VMPT2
FINDS2 ——→ VEQUAL, MINIM2, TLOUT, VMPT2

refby: ISOLAT
comm: DETINF, FILTRT, SYSUL, DETXBI, DCIDEI, SYNC, HFCOM, MCONCO, JUMPCM, NAMES.
(In addition, FINDS CMP/FINDS2 ——→ MULTDT, SIGTAU & FINDS1 ——→ SIG)

name: RECONF
func: Reconfigures the FTS for proper operation (if possible) after failures have been detected and isolated, and after sensors heal.
call: Call RECONF (Ihfail)
args: Ihfail —— integer in ; Heal/fail reconfiguration flag where Ihfail = 1 for failures and -1 for healings
refs: PNTINV, RCOV, SETISN, CLPSIO, TLOUT, IMTCG2
In addition,
FINDS CMP ——→ NOISR, MATNL2
FINDS2/FINDS2 ——→ MATNL3
refby: NAV
comm: DETINF, FILTRT, SYSUL, DETXBI, DCIDEI, SYNC, SYSXBO, INITVL, EKBP0, HEALCM, HFCOM, SYSX1, GBLEND, EKF1, ABRTCM.
In addition,
FINDSCMP ——→ MULTDT
FINDS1 ——→ SYSYWL, SIG
FINDS2 ——→ MULTDT, SYSYWL, SIGTAU

name: CLPSIO
func: Used to collapse (or expand) the NFF and its associated data structures due to a single failure (or healing) of a measurement sensor. This routine is not called when an input sensor is involved.
  1) If Iclps < 0 (i.e., collapse NFF)
      * set RPl (icmd) = 0
      * reset PFI and PBFO by calling subroutine RCOV
      * decrement NY, NYF
      * update INOYP, INORYP, INOYPI
      * if meas. sensor bias is estimated, collapse bias portion of filter by calling subroutine CLPSBE
  2) If Iclps > 0 (i.e., expand NFF)
      * call NOISR to set RFI
      * increment NY, NYF
      * update INDYP, INORYP, INOYPI
      * correct partial derivative of h w.r.t. XF1, i.e., HP1 by calling UPDPH
call: Call CLPSIO (Iclps, Isns, Ireplc)
args: Iclps —— integer in ; flag used to control collapse/expansion of NFF where Iclps = 1 ——> collapse & Iclps = 1 ——> expand
      Isns —— integer in ; absolute index of sensor (from Table 6.5)
      Ireplc —— integer in ; replication of the sensor (1 or 2)
refs: RCOV, PNTINV, IMTCG2, CLPSBE
In addition, FINDS CMP ——→ NOISR, UPDPH & FINDS 2 ——→ UPDPH
refby: RECONF
comm: SYSXBO, SYSUL, SYSYWL, DETXBI, DETINF, INITVL, SYSX1
In addition, FINDS1 ——→ FILTRT, SIG & FINDS2 ——→ FILTRT, SIGTAU
name: NOISR (only in FINDSCMP)
func: Resets the measurement noise covariance terms in the NFF for a given sensor type and replication

call: Call NOISR (Isns, Ireplc, Imul)
args: Isns -- integer in ; absolute index of sensor (from Table 6.5)
Ireplc -- integer in ; not used
Imul -- integer in ; flag to use higher noise covariance when collapsing the IMU portion of filter. (default value = 1, value = 2 when IMU is involved)

refs: None
refby: RECONF, CLPSIO
comm: FILTRT, SYSYW1, SIGTAU, SYSU1

name: CLPSBE
func: Responsible for resetting the bias estimator portion of the NFF such that a single bias can be added or deleted
1) calls ADJTBP to determine IBkey and IYkey and to adjust the bias pointer vector INOBP, as well as NXB, NUB, NYB, NUBL, and NB
2) if kflag = -1 (i.e., collapse the bias estimator)
   (a) the IBkey row and column of the bias filter error covariance PBFO, is deleted.
   (b) the IBkey column of the bias filter blender gain, VBO is deleted
   (c) the IBkey row of the bias estimation vector, XBFO, is deleted
3) if kflag ≠ -1 (i.e., expand the bias estimation)
   (a) PBFO is expanded about the IBkey row and column, and they are zeroed out
   (b) the initial bias filter error covariance is loaded into the appropriate diagonal element s.t. PBFO (IBkey) = PBFOI (Ibias) **2
   (c) VBO is expanded about the IBkey column, and it is zeroed out.
   (d) XBFO is expanded about the IBkey column, and zeroed out.

call: Call CLPSBE (kflag, Ibias)
args: kflag -- integer in ; flag to collapse/expand the bias filter
Ibias -- integer in ; absolute index of bias type to be added or deleted. (from Table 6.5)

refs: FINDSCMP --> ADJTBP, MATCG2
       FINDS1/FINDS2 --> PHTINV, IMTCG2, MATCG3
refby: CLPSIO
comm: SYSXB0, EKBF0, INITVL, GBLEND
       In addition, FINDS1/FINDS2 --> SYSX1, DEXBI, CMPSTF, SYSU1, SYSYW1

name: ADJTBP (only in FINDSCMP)
func: Increments or decrements various vectors/scalars used by CLPSBE and the bias filter, when adding or deleting biases in the estimator

call: Call ADJTBP (Iflag, Index, Irkey, Iykey)
args: Iflag -- integer in ; flag indicating addition/deletion of bias (1 => add, -1 => delete)
Index -- integer in ; absolute index to sensor type of bias to be added or deleted (from Table 6.5)
Irkey -- integer out ; pointer to index in reduced bias set
Iykey -- integer out ; pointer to output type which corresponds to bias referred to by index. (If bias is an input bias, iykey = 0) Table 6.2

refs:  PNTINV, IMTCG2
refby:  CLPSBE
comm:  SYSX1, DETXBI, CMPSTF, SYSXBO, SYSU1, SYSYW1

name:  RCOV
func:  Resets the NFF estimation error covariances once a failure has been detected and isolated. In particular, it sets,

\[ PFI = PFI + VBI \cdot VBI' + (XMI \cdot XMI + 1.0/PHI) \]

if PBFO > PBFOI --- PBFO = PBFOI, XBFO = 0

call:  Call RCOV (Vi, Xmi, Pmi, Icmd)
args:  Vi -- real in ; blender gain for i'th detector (i \equiv Table 6.6)
       Xmi -- real in ; estimate of i'th failure level (i \equiv Table 6.6)
       Pmi -- real in ; information matrix for i'th failure (i \equiv Table 6.6)

6.6)  Icmd -- integer in ; absolute sensor type of failed sensor (Table 6.5)

refs:  FINDSCMP --- VMFR, MTLN2
       FINDSI/FINDS2 --- VMFR2, MATNL3
refby:  CLPSBE, RECONF
comm:  SYSXBO, EKBF0, EKFI, CMPSTF, SYSX1, INITVL

In addition, FINDSCMP --- MAIN1

name:  HEALR
func:  Manages the operation of the healer logic. Primary function is to maintain all sensor failed by the FDI logic and determine if they have healed or recovered. Healer decisions are made ONLY at the end of a healer decision window (which in our algorithms is set to be 3 seconds). In FINDSCMP/FINDSI, special logic is employed in order to force the IMU's to heal in a coordinated fashion.

HEALR is operated by computing the running sum, Xsum, of (Xwork - Xfail) over the heater window of length Knxhlr (3 seconds). The value of the sum is reset to zero at the start of a new heater window; a new heater window is started whenever a new sensor is failed by the FDI logic. Xwork and Xfail are defined as follows:

* for input sensors:
  \[ Xwork = \text{measurement from a currently active replicated sensor of the same type as the failed one} \]
  \[ Xfail = \text{measurement from the failed sensor} \]

* for output sensors:
  \[ Xwork = \text{estimate of the observation obtained from the NFF} \]
  \[ Xfail = \text{measurement from the failed sensor} \]

call:  Call HEALR
args:  None
refs:  BUBBL2, TLOUT
       In addition, FINDSCMP --- LRTHLR
refby:  NAV
comm:  SYNC, SYSU1, HEALCM, HFCOM, EKFI, YOBSRV, JUMPCH, NAMES, LOCHEA
       In addition,
       FINDSCMP --- AGOUT, SOUT, MLOUT, RGOUT, LAOUT, PSIR
       FINDS1 --- AGOUT, RGOUT, PSIR
       FINDS2 --- ASOUT, MLOUT, LAOUT

- 55 -
name: LRTHLR (only in FINDSCMP ; this routine is integrated into HEALR in FINDS1/FINDS2)
func: Performs a likelihood ratio test to determine if a sensor has healed at the end of a healer window. The test is performed as follows:
1) a maximum likelihood estimate of the normal operational bias is computed as, Best = Xsum/Length, where Xsum is the running sum from HEALR and Length is the number of samples in the window.
   The estimate is limited by:
   if Best > Bthrsh , Best = Bthrsh
   if Best < -Bthrsh , Best = Bthrsh
   where Bthrsh is the largest expected bias level for this sensor type (set in INITG)
2) a maximum likelihood estimate for a failure level is computed as,
   Fest = Xsum/Length, which is then limited by:
   if Fest > 0 & Fest < Fthrsh , Fest = Fthrsh
   if Fest < 0 & Fest > -Fthrsh , Fest = -Fthrsh
   where Fthrsh is the smallest expected failure level for this sensor type (set in INITG).
3) a decision function is evaluated as,
   Xtmp = 2.0 * (Fest -Best) * Xsum + Length * (Best **2 + Fest **2)
   4) the value of the decision function is compared to a decision threshold, Dthrsh, (set in INITG), and if Xtmp < Dthrsh the sensor is declared "healed."
call: Call LRTHLR (Xsum, J)
args: Xsum -- real in ; sum of (Xwork -Xfail) over healer window
       J -- integer in ; absolute index of failed sensor (refer Table 6.5)
refs: None
refby: HEALR
comm: HEALCM

name: TLOUT
func: Prints a coded message corresponding to an `event` and the status of the NFF estimates in the time-line file.
call: Call TLOUT (Msg, Imsg1, Imsg2)
args: Msg -- integer in ; message number corresponding to specific events.
      Imsg1, Imsg2 -- integers in ; message qualifiers
refs: None
refby: FINDS/FINDS1/FINDS2, DECIDE, RECONF, HEALR
comm: MCONCO, SYNC, EKF1, EKBF0

DESCRIPTION OF LIBRARY (MATRIX/VECTOR) ROUTINES

name: VMPRT/VMPRT2
func: Prints out vectors or diagonals of matrices
call: Call VMPRT (X, Nr, Nc, Name) ←→ FINDSCMP
      Call VMPRT2 (X, Nr, Nc, Name, Ndiml) ←→ FINDS1/FINDS2
args: X -- real in ; vector or matrix to be printed
      Nr -- integer in ; row size of vector/matrix
      Nc -- integer in ; column
      Name -- character in ; character label to be printed
      Ndiml -- integer in ; only in exact dimensioned FINDS1 & FINDS2 --
      max. row dimension of X in calling routine

- 56 -
name: **BUBBLE**
func: Performs a bubble sort on an array of integers where the final ordering is smallest to largest, i.e., increasing in value.
call: Call BUBBLE (Na, n)
args: Na -- integer in out ; array of integers to be sorted
      n -- integer in ; length of array Na
refs: None
refby: INITG, HEALR
comm: None

name: **MAT1A/PMAXB**
func: Forms the matrix product $Z = XY$. No sparseness tests are performed and $Z, Y$ can start at same core locations. PMAXB assumes that $X, Y, Z$ are exact-dimensional in the calling routine while MAT1A assumes a general 'NDIM' row dimension for all matrices.
call: Call MAT1A/PMAXB (nl, n2, n3, X, Y, Z)
args: nl -- integer in ; row dimension of X, Z
      n2 -- integer in ; col. length of X, row length of Y
      n3 -- integer in ; col. length of Y, Z
      X -- real in ; input matrix (nl, n2)
      Y -- real in ; input matrix (n2, n3)
      Z -- real out ; output matrix (nl, n3)
refs: None
refby: EKFN1, BIASF, BLEND, ISOLAT, MAT3/PMABT
comm: MAT1A --> MAIN1
       PMAXB --> None

name: **MAT2/PMABT2**
func: Forms the matrix product $Z = XY^T$ where $Z$ is symmetric. No sparseness tests are done and $Z, Y$ can start at same core locations. PMABT2 assumes that $X, Y, Z$ are exact-dimensional in the calling routine while MAT2 assumes an 'NDIM' row dimension for all matrices.
call: Call MAT2/PMABT2 (nl, n2, X, Y, Z)
args: nl -- integer in ; row dimension of X, Y and col. length of Z
      n2 -- integer in ; row dimension of X, Y
      X -- real in ; input matrix (nl, n2)
      Y -- real in ; input matrix (nl, n2)
      Z -- real out ; output matrix (nl, n2)
refs: None
refby: EKFN1, ISOLAT
comm: MAT2 --> MAIN1
       PMABT2 --> None

name: **MAT3/PMABAT**
func: Forms the symmetric matrix product $Z = X Y Y^T$ where $Y$ is symmetric, and no sparseness tests are done. PMABAT assumes that $X, Y, Z$ are exact-dimensional in the calling routine while MAT3 assumes an 'NDIM' row dimension for all matrices.
call: Call MAT3 (nl, n2, X, Y, Z) --> FINDSCMP
Call PMABAT (nl, X, Y, Z) \texttt{--- FINDS1/FINDS2} (assumes \( n_1 = n_2 \))

**args:**
- \( n_1 \rightarrow \) integer in ; row length of \( X, Z \) and col. length of \( Z \)
- \( n_2 \rightarrow \) integer in ; row length of \( Y \) and col. length of \( X, Y \)
- \( X \rightarrow \) real in ; input matrix \((n_1, n_2)\)
- \( Y \rightarrow \) real in ; input (symmetric) matrix \((n_2, n_2)\)
- \( Z \rightarrow \) real out ; output (symmetric) matrix \((n_1, n_1)\)

**refs:** MAT1A/PMAXB

**refby:** EKFN1

**comm:** MAT3 \texttt{--- MAIN1}

PMABAT \texttt{--- None}

**name:** MAT3B/PMVTAV

**func:** Forms a scalar output from the symmetric vector product

\[ Z = V^T Y V, \] where \( Y \) is symmetric and no sparseness tests are done.

PMVTAV assumes that \( Y \) is exact-dimensioned in the calling routine
while MAT3B assumes an '\text{NDIM}' row dimension for \( Y \).

**call:** Call MAT3B/PMVTAV \((n_1, V, Y, SOUT)\)

**args:**
- \( n_1 \rightarrow \) integer in ; dimension of vector \( V \) and row/col. length of matrix \( Y \)
- \( V \rightarrow \) real in ; input vector \((n_1, 1)\)
- \( Y \rightarrow \) real in ; input (symmetric) matrix \((n_1, n_1)\)
- \( SOUT \rightarrow \) real out ; scalar output

**refs:** None

**refby:** DET01, DET05, DET10, ISOLAT, LRT

**comm:** MAT3B \texttt{--- MAIN1}

PMVTAV \texttt{--- None}

**name:** MATXYT/PMABT

**func:** Forms the matrix product \( Z = X Y^T \), no sparseness test on \( Y \) PMABT assumes that \( X, Y, Z \) are exact-dimensioned in the calling routine
while MATXYT assumes an '\text{NDIM}' row dimension on all matrices.

**call:** Call MATXYT/PMABT \((n_1, n_2, n_3, X, Y, Z)\)

**args:**
- \( n_1 \rightarrow \) integer in ; row dimension of \( X, Z \)
- \( n_2 \rightarrow \) integer in ; col. length of \( X, Y \)
- \( n_3 \rightarrow \) integer in ; row length of \( Y, \) col. length of \( Z \)
- \( X \rightarrow \) real in ; input matrix \((n_1, n_2)\)
- \( Y \rightarrow \) real in ; input matrix \((n_3, n_2)\)
- \( Z \rightarrow \) real out ; output matrix \((n_1, n_3)\)

**refs:** None

**refby:** BIASF, ISOLAT

**comm:** MATXYT \texttt{--- MAIN1}

PMABT \texttt{--- None}

**name:** MEQUAL/PMBEA

**func:** Sets a matrix \( Y \) equal to a matrix \( X \)

PMBEA assumes that \( X, Y \) are exact-dimensioned in the calling routine
while MEQUAL assumes an '\text{NDIM}' row dimension for \( X, Y \).

**call:** Call MEQUAL/PMBEA \((n_1, n_2, X, Y)\)

**args:**
- \( n_1 \rightarrow \) integer in ; row length of \( X, Y \)
- \( n_2 \rightarrow \) integer in ; col. length of \( X, Y \)
- \( X \rightarrow \) real in ; input matrix \((n_1, n_2)\)
- \( Y \rightarrow \) real out ; output in \((n_1, n_2)\)

**refs:** None
**TRANS2** (only in FINDSOMP)

**func:** Transpose a matrix, $X^{\top}$. Assumes 'NDIM' row dimension for all matrices.

**call:** Call TRANS2 ($n_1$, $n_2$, $X$, $X^{\top}$)

**args:**
- $n_1$ -- integer in; row length of $X$, col. length of $X^{\top}$
- $n_2$ -- integer in; col. length of $X$, row length of $X^{\top}$
- $X$ -- real in; input matrix ($n_1$, $n_2$)
- $X^{\top}$ -- real in; output matrix ($n_2$, $n_1$)

**refs:** None

**reby:** ISOLAT

**comm:** MAIN1

---

**PDMINV/PD3NV1/PD4NV1**

**func:** Special matrix inverse routine for positive, symmetric, semi-definite matrices; uses Cholesky $L\cdot U$ decomposition as an intermediate step. PD3NV1 is the special form of PDMINV for 3rd order matrices, used in FINDSI and PD4NV1 inverts 4th order matrices in FINDS2.

**call:**
- Call PDMINV ($n$, $A$, $A^{\text{-1}}$)
- Call PD3NV1/PD4NV1 ($A$, $A^{\text{-1}}$)

**args:**
- $n$ -- integer in; order of matrix to be inverted
- $A$ -- real in; input matrix ($n$, $n$)
- $A^{\text{-1}}$ -- real out; output matrix ($n$, $n$)

**NOTE:** PD3NV1 & PD4NV1 assume $n = 3$ & $n = 4$, respectively.

**refs:** None

**reby:** EKPNL, BIASF, ISOLAT

**comm:** None

---

**MINIM2/MINIM3**

**func:** Searches a vector and determines the minimum value and its corresponding location. Only those elements of the vector are checked which have a corresponding non-zero element in the other input vector.

**call:**
- Call MINIM2 ($\text{Imactv}$, $V$, $Npts$, $Vmin$, $Nmin$)
- Call MINIM3 ($\text{Imactv}$, $V$, $Npts$, $Nmin$)

**args:**
- $\text{Imactv}$ -- integer in; input vector ($Npts$, 1) with 0 or 1 entries corresponding to which entries in $V$ to be checked.
- $V$ -- real in; input vector ($Npts$, 1) to be searched
- $Npts$ -- integer in; length of $V$ (i.e., # of elements to be searched)
- $Vmin$ -- real out; value of minimum element in $V$ (not in MINIM3 which outputs only the location)
- $Nmin$ -- integer out; location of the minimum element in $V$

**refs:** None

**reby:** DECIDE

**comm:** None
name: PNTINV
func: Searches a pointer vector for particular entry. The pointer vector is an integer array with monotonically increasing elements. It will show how a collapsed vector's elements relate to a standard vector, i.e., given the absolute index, this routine reruns the active index in the reduced vector.
call: Call PNTINV (isns, Ipoint, n, index)
args: isns -- integer in ; value searched for in Ipoint (usually in absolute index)
      Ipoint -- integer in ; pointer vector to be searched
      n -- integer in ; length of Ipoint
      index -- integer out ; index in Ipoint where isns was found. If isns was not found, index < 0
refs: None
refby: RECONF, CLPSIO, ADJTBP
comm: None

name: IMTCG2
func: To add or delete a row in an integer matrix or vector, or to add or delete a column in a matrix. If a row or column is added, its elements are set to zero.
args: jflag -- integer in ; operation flag where:
      1 -- add row, 2 -- add column
      -1 -- delete row, -2 -- delete column
      index -- integer in ; pointer to row/column to be added or deleted
      IY -- integer in out ; matrix whose 'index' row/column is to be added or deleted
      nr -- integer in out ; # of rows of Y (incremented or decremented)
      nc -- integer in out ; # of columns of Y (incremented or decremented) (not used in FINDS1/FINDS2)
refs: None
refby: RECONF, CLPSIO, ADJTBP, (CLPSBE)
comm: FINDSCMP --- MAINI
FINDS1/FINDS2 --- None (as no column operations are performed)

name: MATCG2/MATCG3
func: To add or delete a row/column in a 'real' matrix or vector. If a row or column is added, its elements are set to zero. MATCG3 assumes that the matrix is exact-dimensioned in the calling routine while MATCG2 assumes an 'NDIM' row dimension for the matrix.
call: Call MATCG2 (jflag, index, Y, nr, nc)
Call MATCG3 (jflag, index, Y, nr, nc, ndiml)
args: jflag -- integer in ; operation flag where:
      1 -- add row, 2 -- add column
      -1 -- delete row, -2 -- delete column
      index -- integer in ; pointer to row/column to be added or deleted
      Y -- real in out ; matrix whose 'index' row/column is to be added or deleted
      nr -- integer in out ; # of rows of Y (incremented or decremented)
      nc -- integer in out ; # of columns of Y (incremented or decremented)
      ndiml -- integer in ; maximum row dimension of Y in calling routine (used only in FINDS1/FINDS2 for exact-dimensioned matrices)
refs: RECONF, CLPSIO, ADJTBP, (CLPSBE)
refs: None
refby: RECONF, CLPSIO, ADJTBP, (CLPSBE)
comm: FINDSCMP --- MAINI
FINDS1/FINDS2 --- None (as no column operations are performed)
name: MATNL2/MATNL3
func: Initializes columns nl through n2 of a matrix to zero. In addition, if a flag is set, rows nl through n2 can be nulled out, as well. MATNL3 assumes that the matrix is exact-dimensioned in the calling routine while MATNL2 assumes an 'NDIM' row dimension for all matrices.
call: Call MATNL2 (X, nl, n2, ktrig, n3)
Call MATNL3 (X, nl, n2, ktrig, n3, ndiml)
args: X -- real in out; matrix whose rows/columns have to be nulled
   nl -- integer in; first column/row to be nulled
   n2 -- integer in; last column/row to be nulled
   ktrig -- integer in; operation flag: 0 --> only columns
            #0 --> rows & columns
   n3 -- integer in; # of elements in any row to be nulled (used because all column dimensions are exact).
   ndiml -- integer in; maximum row dimension of X in calling routine. (only in exact-dimensioned FINDSI/FINDS2)
refs: None
refby: ISOLAT, RECONP, RC0V
comm: MATNL2 --> MAIN1
       MATNL3 --> None

name: MADD/PMAPB
func: Adds two matrices as Z = X + Y. PMAPB (used in FINDS1/FINDS2 assumes that all matrices are exact-dimensioned in the calling routine while MADD (used in FINDSCOMP) assumes an 'NDIM' row dimension for all matrices.
call: Call MADD/PMAPB (nl, n2, X, Y, Z)
args: nl -- integer in; row length of X, Y, Z
   n2 -- integer in; column length of X, Y, Z
   X -- real in; input matrix (nl, n2)
   Y -- real in; input matrix (nl, n2)
   Z -- real out; output matrix (nl, n2)
refs: None
refby: EKFN1, BIASF, BLEND, ISOLAT
comm: MADD --> MAIN1
       PMAPB -->

name: MSUB/PMAMB
func: Matrix subtraction as Z = X - Y. PMAMB (used in FINDS1/FINDS2) assumes that all matrices are exact-dimensioned in the calling routine while MSUB (used in FINDSCMP) assumes an 'NDIM' row dimension for all matrices.
call: Call MSUB/PMAMB (nl, n2, X, Y, Z)
args: refer args. for MADD/PMAPB
refs: None
refby: BIASF, ISOLAT
comm: MSUB --> MAIN1
       PMAMB -->
name: MATVEC/PMAXV
func: Performs matrix vector multiplication as \( V2 = X \cdot V1 \). PMAXV (used in FINDS1/FINDS2) assumes that the matrix \( X \) is exact-dimensioned in the calling routine while MATVEC (used in FINDSCMP) assumes an 'NDIM' row dimension for all matrices.
call: Call MATVEC/PMAXV (nl, n2, X, V1, V2)
args: nl -- integer in ; row length of \( X \), length of \( V2 \)
n2 -- integer in ; column length of \( X \), length of \( V1 \)
X -- real in ; input matrix (nl, n2)
V1 -- real in ; input vector (n2)
V2 -- real out ; output vector (nl)
refs: None
refby: BIASF, ISOLAT
comm: MATVEC --> MAIN1
       PMAXV --> None

name: MATVC2/PMAXV2
func: Computes matrix-vector-product-sum as \( V3 = X \cdot V1 + V2 \) (an extension of MATVEC/PMAXV)
call: Call MATVC2/PMAXV2 (nl, n2, X, V1, V2, V3)
args: same as MATVEC/PMAXV with exception of
       V2 -- real in ; input vector (nl)
       V3 -- real out ; output vector (nl)
refs: None
refby: BLEND
comm: MATVC2 --> MAIN1
       PMAXV2 --> None

name: VSCALE
func: Performs vector scaling as \( V2 = s \cdot V1 \)
call: Call VSCALE (nl, stmp, V1, V2)
args: nl -- integer in ; length of vectors \( V1, V2 \)
stmp -- real in ; scale factor
       V1 -- real in ; input vector to be scaled
       V2 -- real out ; output vector
refs: None
refby: EKFM1, BIASF
comm: None

name: VEQUAL
func: Equates vectors as, \( V2 = V1 \)
call: Call VEQUAL (nl, V1, V2)
args: nl -- integer in ; length of \( V1, V2 \)
       V1 -- real in ; input vector
       V2 -- real out ; output vector
refs: None
refby: INITG, ISOLAT, DECIDE
comm: None

name: VADD
func: Performs vector addition as \( V3 = V1 + V2 \)
call: Call VADD (nl, V1, V2, V3)
args:  n1 -- integer in ; length of V1, V2, V3
       V1 -- real in ; input vector
       V2 -- real in ; input vector
       V3 -- in out ; output vector (result of addition)
refs:  None
refby:  ISOLAT
comm:  None

name:  VSUB
func:  Performs vector subtraction as, V3 = V1 - V2
call:  Call VSUB (n1, V1, V2, V3)
args:  same as VADD
refs:  None
refby:  BIASF, ISOLAT
comm:  None
8. COMMON BLOCK DESCRIPTION AND TABLES

This section contains a list of FINDS program variables as partitioned by various common blocks. Table 8.1 is a "short form" list of each common block in FINDS and the various subprograms which refer to it. Supporting Table 8.1 is a detailed description of the variables contained in each block.

**TABLE 8.1**

<table>
<thead>
<tr>
<th>Common Block</th>
<th>Referenced by Subprogram(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABRTOt</td>
<td>FINDS (main), NAV, RECONF</td>
</tr>
<tr>
<td>AGOUT</td>
<td>READFL, INITXF, SUMOUT, RESCMP, HEALR</td>
</tr>
<tr>
<td>ASOUT</td>
<td>READFL, INITG, INITXF, SUMOUT, RESCMP, HEALR</td>
</tr>
<tr>
<td>BIASF</td>
<td>BNSAV1, BNSAV2, SORTER</td>
</tr>
<tr>
<td>BLNDWK</td>
<td>BNSAV2, BNSAV2, SORTER</td>
</tr>
<tr>
<td>BNSRT1</td>
<td>BNSAV1, BNSAV2, SORTER</td>
</tr>
<tr>
<td>BNSRT2</td>
<td>BNSAV2, BNSAV2, SORTER</td>
</tr>
<tr>
<td>BNSRT2</td>
<td>BNSAV2, SORTER</td>
</tr>
<tr>
<td>BSFWFK</td>
<td></td>
</tr>
<tr>
<td>CMPSTF</td>
<td>INITG, BLEND, UPDPH, ISOLAT, RCOV</td>
</tr>
<tr>
<td>CNTROL</td>
<td>FINDS (main), NAV, INITG, EKFN1, BIASF, DFT01, DFT05, DET10</td>
</tr>
<tr>
<td>DCTIDEI</td>
<td>ISOLAT, DECIDE, RECONF</td>
</tr>
<tr>
<td>DETCOV</td>
<td>BIASF, ISOLAT</td>
</tr>
<tr>
<td>DETINF</td>
<td>INITG, BLEND, SPTISN, RESCMP, ISOLAT, BNSAV2, LKF, LRT, DECIDE, RECONF, CLPSIO</td>
</tr>
<tr>
<td>DETLC2</td>
<td>ISOLAT</td>
</tr>
<tr>
<td>DETLC3</td>
<td>ISOLAT, LKF, LRT</td>
</tr>
<tr>
<td>DETPRI</td>
<td>DET01, DET05, DET10</td>
</tr>
<tr>
<td>DETWRK</td>
<td></td>
</tr>
<tr>
<td>DETXBI</td>
<td>INITG, SUMOUT, SETISN, ISOLAT, LKF, DECIDE, RECONF, CLPSIO</td>
</tr>
<tr>
<td>DETYBI</td>
<td>ISOLAT, LKF, LRT</td>
</tr>
<tr>
<td>DCTC01</td>
<td>DET01, BNSAVI</td>
</tr>
<tr>
<td>DCTC05</td>
<td>DET05, BNSAVI</td>
</tr>
<tr>
<td>DCTC10</td>
<td>DET10, BNSAVI</td>
</tr>
<tr>
<td>DTSYNC</td>
<td>NAV, RESCMP, ISOLAT, BNSAV2</td>
</tr>
<tr>
<td>EARTH</td>
<td>FINDS (main), SUMIN, GTOI</td>
</tr>
<tr>
<td>EKBFO</td>
<td>INITG, SUMIN, BIASF, BLEND, DET01, DET05, DET10, UPDH, ISOLAT, RECONF, ILOUT, CLPSBE, RCOV, BNSAV1, BNSAV2</td>
</tr>
<tr>
<td>EKFI</td>
<td>NAV, INITG, INITXF, GT08, EKFN1, BIASF, BLEND, UPDB, UPDH, RESCMP, ILOUT, BNSAV2, ISOLAT, RECONF, RCOV, HEALR</td>
</tr>
<tr>
<td>EKFBIA</td>
<td>EKFN1, BIASF</td>
</tr>
<tr>
<td>EKFBLN</td>
<td></td>
</tr>
<tr>
<td>EKFWRK</td>
<td></td>
</tr>
<tr>
<td>EULER</td>
<td>UPDB</td>
</tr>
<tr>
<td>FILTIC</td>
<td>INITG</td>
</tr>
</tbody>
</table>

- 64 -
<table>
<thead>
<tr>
<th>Common Block</th>
<th>Referenced by Subprogram(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILTRT</td>
<td>INITG, INITXF, SUMIN, SUMOUT, GTOI, EKFN1, BIAS, BLEND, DET01, DET05, DET10, NOISR, RECONF, DECIDE, SETISN, RESCMP, ISOLAT</td>
</tr>
<tr>
<td>FLTIN</td>
<td></td>
</tr>
<tr>
<td>GBLEND</td>
<td>BIASF, BLEND, RECONF, CLPSBE</td>
</tr>
<tr>
<td>GRYVTC</td>
<td>GTOI</td>
</tr>
<tr>
<td>GTOI01C</td>
<td>GTOI</td>
</tr>
<tr>
<td>HEALCM</td>
<td>NAV, INITG, RECONF, HEALR, LRTHLR</td>
</tr>
<tr>
<td>HPCom</td>
<td>NAV, DECIDE, RECONF, HEALR</td>
</tr>
<tr>
<td>IMLS</td>
<td>FINDS (main), GTOI</td>
</tr>
<tr>
<td>INITVL</td>
<td>INITG, ISOLAT, RECONF, CLPSIO, CLPSBE, RCOV</td>
</tr>
<tr>
<td>JUMPCH</td>
<td>NAV, EKFN1, BIASF, BLEND, DET01, DET10, DECIDE, HEALR</td>
</tr>
<tr>
<td>LOUT</td>
<td>READPL, SUMIN, HEALR</td>
</tr>
<tr>
<td>LAT10N</td>
<td>SUMIN, GTOI, BNSAV2</td>
</tr>
<tr>
<td>LCOM21</td>
<td>BNSAV1, BNSAV2, HEALR</td>
</tr>
<tr>
<td>LOCHEA</td>
<td>BNSAV2, TLOUT</td>
</tr>
<tr>
<td>LR1ENV</td>
<td>BIASF, DET01, DET05, DET10</td>
</tr>
<tr>
<td>LR1MAX</td>
<td>FINDS (main), DET01, DET05, DET10</td>
</tr>
<tr>
<td>MAIN1</td>
<td>INITG, SUMIN, GTOI, BIASF, UPDB, UPDph, ISOLAT, LKF, LRT, RCOV, BNSAV2</td>
</tr>
<tr>
<td>MAIN2</td>
<td>EKFN1, BIASF, BLEND, ISOLAT, BNSAV2</td>
</tr>
<tr>
<td>MCONCO</td>
<td>FINDS (main), READPL, INITG, INITXF, GTOI, DECIDE, BNSAV1, BNSAV2, TLOUT</td>
</tr>
<tr>
<td>ML1OUT</td>
<td>READPL, INITXF, SUMOUT, RESCMP, HEALR</td>
</tr>
<tr>
<td>MS1ALL</td>
<td>FINDS (main), INITXF, UPDh, UPDph,</td>
</tr>
<tr>
<td>MUL1DT</td>
<td>ISOLAT, DECIDE, RECONF</td>
</tr>
<tr>
<td>NAMES</td>
<td>READPL, DECIDE, HEALR</td>
</tr>
<tr>
<td>PSIR</td>
<td>FINDS (main), INITXF, SUMIN, SUMOUT, GTOI, RESCMP, HEALR, BNSAV2</td>
</tr>
<tr>
<td>PQR DEG</td>
<td>GTOI</td>
</tr>
<tr>
<td>RD1OCL</td>
<td>READPL</td>
</tr>
<tr>
<td>RGOUT</td>
<td>READPL, SUMIN, GTOI, HEALR</td>
</tr>
<tr>
<td>S1GTAU</td>
<td>FINDS (main), INITG, DECIDE, NOISR</td>
</tr>
<tr>
<td>S1MLOC</td>
<td>SUMIN</td>
</tr>
<tr>
<td>S1NC</td>
<td>FINDS (main), READPL, NAV, INITG, SUMIN, DET01, DET05, DET10, UPDB, DECIDE, RECONF, HEALR, BNSAV1, TLOUT</td>
</tr>
<tr>
<td>S1SY1</td>
<td>INITG, SUMIN, GTOI, EKFN1, BIASF, BLEND, DET01, DET05, DET10, SETISN, UPDB, UPDh, BNSAV2, HEALR, ADJTBP, NOISR, CLPSIO, RECONF, DECIDE, UPDPh, RESCMP, ISOLAT</td>
</tr>
<tr>
<td>S1SYXI</td>
<td>INITG, EKFN1, BIASF, BLEND, UPDB, UPDPh, UPDPh, ISOLAT, RECONF, CLPSIO, ADJTBP, RCOV, BNSAV2</td>
</tr>
<tr>
<td>S1SXBO</td>
<td>NAV, INITG, EKFN1, BIASF, BLEND, UPDB, UPDPh, ISOLAT, RECONF, CLPSIO, CLPSBe, ADJTBP, RCOV, BNSAV2</td>
</tr>
<tr>
<td>S1SYBO</td>
<td>EKFN1, BIASF, RECONF,</td>
</tr>
<tr>
<td>S1SYWI</td>
<td>INITG, SUMOUT, EKFN1, BIASF, BLEND, DET01, DET05, DET10, UPDPh, ISOLAT, CLPSIO, NOISR, ADJTBP, BNSAV2</td>
</tr>
<tr>
<td>TR1BER</td>
<td>SUMIN, GTOI, UPDB</td>
</tr>
<tr>
<td>TSTORE</td>
<td>EKFN1, BIASF, BLEND, ISOLAT</td>
</tr>
<tr>
<td>UD1QLC</td>
<td></td>
</tr>
<tr>
<td>YOBSRV</td>
<td>INITG, SUMOUT, BIASF, UPDPh, UPDPh, RESCMP, ISOLAT, HEALR, BNSAVI</td>
</tr>
</tbody>
</table>
II. DESCRIPTION OF COMMON BLOCKS

NOTE: (1) All vector/matrix dimensions are specified here in three separate parentheses corresponding to their use in FINDSCMP, FINDS1 and FINDS2, respectively. A single parentheses implies that all three versions use the same dimensions.

(2) Notation used is as follows: 
cont: contains (i.e., brief description of common block)
vars: variables contained in common block

name: ABRTCM
cont: System status flag
vars: iabort -- integer, unitless; program abort flag which is activated (i.e., set from 0 to 1) when too many sensors are failed by the FDI logic and filter cannot operate with the remaining sensor complement.
refby: FINDS/FINDS1/FINDS2, RECONF

name: AGOUT (not in FINDS2)
cont: IMU sensor measurements from flight data in program units
vars: Phim -- real, radians, (2) (2) (-); dual replicated IMU roll measurements
      Them -- real, radians, (2) (2) (-); dual replicated IMU pitch measurements
      Psim -- real, radians, (2) (2) (-); dual replicated IMU yaw measurements (w.r.t. North)
refby: READFL, INITXF, SUMOUT, RESCMP, HEALR

name: ASOUT (not in FINDS1)
cont: IAS measurements from flight data in program units.
vars: Airsm -- real, m/s, (2) (-) (2); dual replicated airspeed measurements
refby: READFL, INITXF, SUMOUT, RESCMP, HEALR

name: BLNDWK (only in FINDS2)
cont: Temporary working variable(s) in subroutine BLEND
vars: Vtmp1 -- real, mixed units, (-) (-) (8); temp. vector used in propagation
refby: BLEND

name: BSFWRK (not in FINDSCMP)
cont: Local working variables/arrays in subroutine BIASF
vars: CbfO -- real, mixed units, (-) (3,3) (4,3); bias filter observation matrix
     Com2 -- real, mixed units, (-) (-) (8,8); temporary local matrix
     Tmp1 -- real, mixed units, (-) (-) (8,3); temporary local matrix
     Tmp2 -- real, mixed units, (-) (-) (8,3); temporary local matrix
     Tmp3 -- real, mixed units, (-) (-) (3,4); temporary local matrix
     Tmp4 -- real, mixed units, (-) (-) (4,4); temporary local matrix
     Tmp5 -- real, mixed units, (-) (-) (3,3); temporary local matrix
refby: BIASF

name: CMPSTF
cont: Quantities associated with composite NFF (bias free + bias)
vars: nxb -- integer, unitless ; total states + bias states in NFF
value = (17) (6) (11)
Pxml -- real, mixed, (17,17) (6,6) (11,11) ; combined NFF estimation error covariance

refby: INITG, BLEND, UPDPH, ISOLAT, CLPSBE/ADJTBP, RCOV

name: CTRL
cont: Option flag to activate/deactivate FDI logic
vars: icntr -- boolean, unitless ; false => run NFF only.
true => run FDI portion of algorithm also

refby: FINDS1/FINDS1/FINDS2, NAV, INITG, EKFPI, BIASF, DET01, DET05, DET10

name: DCIDEI
cont: Quantities relevant to the LR computations and the decision logic
vars: Priori -- real, unitless, (20) (9) (11) ; vector of log of prior probabilities of failure
-- one for each sensor, ordered by replicated sensor index of Table 6.6 but assumes dual MLS replication.

Alamda -- real, unitless, (20) (9) (11) ; vector of log-likelihood of sensor failing -- one for each sensor, ordered by replicated sensor index of Table 6.6 but assumes dual MLS replication.

refby: ISOLAT, DECIDE, RECONF

name: DETCOV
cont: Quantity needed in isolation routine to form total covariance
vars: Afbv -- real, unitless, (17,6) (3,3) (8,3) ; intermediate storage matrix which saves the computation AFI*VBO + BFI

refby: BIASF, ISOLAT

name: DETINF
cont: Information pertinent to the bank of first order filters in ISOLAT
vars: nft -- integer, unitless ; total # of replicated sensors (considered for FDI) value = (17) (9) (8)
nyf -- integer, unitless ; current # of replicated measurement sensors value = (11) (6) (5) from Table 6.7
Inoryp -- integer, unitless, (17) (6) (11) ; pointer vector to measurement sensor type

Icntsn -- integer, unitless, (20) (9) (11) ; determines if a particular sensor type/replication is being used and which element of the input/meas. vector it corresponds to. Null entry implies an inactive sensor. (Table 6.6)

Resboc -- real, mixed, (14,10) (6,10) (8,10) ; expanded residual vector from the NFF saved over the last 10 iterations

refby: INITG, BLEND, SETISN, RESCMP, ISOLAT, LKF, LRT, DECIDE, RECONF, CLPSIO

name: DETLC2 (not in FINDS1)
cont: Variables relevant to multiple MLS sensor failures
vars: Dobs - real, mixed, (17,6) (-) (5,6) ; observation matrix to generate dual failure conditioned residuals
Best -- real, mixed, (6) ; estimated magnitude of multiple replicated MLS failure

refby: ISOLAT

name: DETLC3
cont: Quantities local to the isolation logic which are temporarily stored
vars: Detinv -- real, mixed, (17,14) (6,6) (11,5) ; inverse of expanded innovations covariance
Hpaf -- real, mixed, (17,11) (-) (4,8) ; computed HP1*AF1
Hpbf -- real, mixed, (17,6) (6,3) (11,3) ; computed HP1*BF1
Augm -- real, mixed, (17,6) (6,3) (11,3) ; intermediate augmented matrix
Hdpd -- real, mixed, (17,6) (6,3) (11,3) ; computed HP1*BF1 + D
Bmghb -- real, mixed, (17,6) (6,3) (11,3) ; BF1 - GAINKX*HP1*BF1

refby: ISOLAT, LKF, LRT

name: DETPRI
cont: Flag to check if a failure has already been detected in current iteration ; hierarchy of detectors is DETO1, DETO5, DETO5
vars: idfail -- integer, unitless ; set to 1 by any detector which flags a sensor failure -- remaining detectors will be deactivated during current iteration (default value = 0)

refby: DETO1, DETO5, DETO10
name: DETWRK (not in FINDSCMP)
cont: Local working arrays in subroutine ISOLAT
vars: Vtmp1 -- real, mixed, (-) (6) (11) ; temporary working vector
Vtmp2 -- real, mixed, (-) (6) (11) ; temporary working vector
Tmp1 -- real, mixed, (-) (3,3) (8,8) ; temporary working matrix
Tmp2 -- real, mixed, (-) (6,6) (5,11) ; temporary working matrix
Tmp3 -- real, mixed, (-) (6,6) (5,5) ; temporary working matrix
Tmp4 -- real, mixed (-) (-) (5,5) ; temporary working matrix
Com2 -- real, mixed, (-) (-) (8,3) ; temporary working matrix
Hpic -- real, mixed, (-) (6,6) (5,11) ; composite observation matrix
Gnkxd -- real, mixed, (-) (6,3) (11,4) ; augmented NFF gain matrix
[GAINKX/GAINBO]
refby: ISOLAT

name: DETXBI
cont: Quantities associated with the sensor failure isolation & estimation logic
vars: nfmax -- integer, unitless ; maximum possible # of sensor types to be considered (has value = 13, 6, 7)
nymax -- integer, unitless ; maximum possible # of measurement sensor types to be considered (has value = 7, 3, 4)
xbfi -- real, mixed, (20) (9) (11) ; vector of current failure level estimates -- one for each type & replication using absolute indexing (Table 6.6)
Phbfi -- real, mixed, (20) (9) (11) ; vector of estimation information for each estimated failure (ordered as per Table 6.6)
Vbi -- real, mixed, (17,13) (6,6) (11,7) ; matrix of blender gain vectors
refby: INITG, SUMOUT, SETISN, ISOLAT, LKF, DECIDE, RECONF, CLPSIO, ADJTBP

name: DETYBI
cont: Observation matrices and compensated residual vectors for the bank of filters in the isolation logic
vars: Resbi -- real, mixed, (17,20) (6,9) (11,11) ; matrix of failure compensated residuals vectors - cols. are ordered by replicated sensor index (Table 6.6)
Chfi -- real, mixed, (17,13) (6,6) (11,7) ; observation matrix where each col. is an observations vector for a filter. Cols. are ordered by replicated sensor index (Table 6.6)
refby: ISOLAT, LKF
name:  DTCT01
cont:  Quantities associated with the detector of window length 1 sample
vars:  vlrt01 -- real, unitless ; Chi-square test failure likelihood
       ratio
       Rti01 -- real, mixed, (17,7) (3,3) (4,4) ; NFF innovations inverse
       matrix compensated for residual window length of 1 sample
refby:  DET01

name:  DTCT05
cont:  Quantities associated with the detector of window length 5 samples
vars:  vlrt05 -- real, unitless ; Chi-square test failure likelihood
       ratio
       Ravg05 -- real, mixed, (7) (3) (4) ; five sample moving window
       average of NFF residuals
       Rsav05 -- real, mixed, (7,5) (3,5) (4,5) ; saved RESBO over last
       five iterations (moving window)
       Rti05 -- real, mixed, (17,7) (3,3) (4,4) ; NFF innovations inverse
       matrix compensated for residual window length of 5 samples
refby:  DET05

name:  DTCT10
cont:  Quantities associated with the detector of window length 10 samples
vars:  vlrt10 -- real, unitless ; Chi-square test failure likelihood
       ratio
       Ravg10 -- real, mixed, (7) (3) (4) ; ten sample moving window
       average of NFF residuals
       Rsav10 -- real, mixed, (7,10) (3,10) (4,10) ; saved RESBO over
       last ten iterations (moving window)
       Rti10 -- real, mixed, (17,7) (3,3) (4,4) ; NFF innovations inverse
       matrix compensated for residual window length of 5 samples
refby:  DET10

name:  DTSYNC
cont:  Pointer to current location in saved array of NFF expanded residuals
vars:  icurnt -- integer, unitless ; [1,10] location in saved RESBOC
       used in ISOLAT to go back either 5 or 10 iterations and run isolation logic
refby:  NAV, RESCMP, ISOLAT
name: EARTH (not in FINDS2)
cont: Quantities associated with earth's rotation -- used in GTOI to compute a/c latitude, longitude and rate gyro compensation terms
vars: omegt -- real, radians; computed WE * TIME to give angular change between I-frame and E-frame
sint -- real, unitless; sine of omegt
cosm -- real, unitless; cosine of omegt
re -- real, meters; radius of earth
we -- real, rad/s; earth rotation rate
refby: FINDS/FINDS1, SUMIN, GTOI

name: EKBFO
cont: Arrays used in the bias filter portion of the NFF
vars: XbfO -- real, mixed, (17) (3) (3); vector of current normal operating bias estimates (Table 6.3)
ResbO -- real, mixed, (7) (3) (4); vector of NFF residuals (Table 6.2)
GainbO -- real, mixed, (17,7) (3,3) (3,4); Kalman gain for bias filter
PbfO -- real, mixed, (17,6) (3,3) (3,3); bias filter estimation error covariance
refby: INITG, SUMIN, BIASF, BLEND, DET01, DET05, DET10, UPDH, ISOLAT, RECONF, CLPSBE, RCOV, TLOUT

name: EKF1
cont: Arrays used in the bias free portion of the NFF
vars: Xfl -- real, mixed, (11) (3) (8); vector of current NFF state estimates (Table 6.1)
Hxkpl -- real, mixed, (7) (3) (4); vector of NFF observations (Table 6.2)
Gainkx -- real, mixed, (17,7) (3,3) (8,4); Kalman gain for EKF (bias and bias-free)
Pfl -- real, mixed, (17,11) (3,3) (8,8); bias free filter estimation error covariance
refby: NAV, INITG, INITXF, GTOI, EKFN1, BIASF, BLEND, UPDB, UPDH, UPDPH, RESCMP, ISOLAT, RECONF, RCOV, HEALR, TLOUT

name: EKFBIA
cont: arrays common to the bias and bias-free filters
vars: Ximgh -- real, mixed, (17,11) (3,3) (8,8); saved computed I-GAIN "HPI
Tmp1 -- real, mixed, (-) (3,3) (-); temporary working matrix
Tmp2 -- real, mixed, (-) (3,3) (-); temporary working matrix
RbfO -- real, mixed, (-) (-) (4,4); saved HPL * PF2 * HPL' + R computed in EKFN1 and used also in BIASF
refby: EKFN1, BIASF

name: EKFBLN (only in FINDS2)
cont: Working arrays common to subroutines EKFN1 & BLEND
vars: Tmp3 -- real, mixed (-) (-) (8,4) ; temporary working matrix
refby: EKFHl, BLEND

name: EKFWRK (only in FINDS2)
cont: Working arrays local to subroutine EKFHl
vars: Tmp1 -- real, mixed, (-) (-) (8,8) ; temporary working matrix
      Tmp2 -- real, mixed, (-) (-) (8,8) ; temporary working matrix
      Gktmp -- real, mixed, (-) (-) (4,4) ; intermediate gain matrix
refby: EKFHl

name: EULER
cont: Sine/cosine values of a/c Euler angles
vars: s1 -- real, unitless ; sine of roll attitude
      c1 -- real, unitless ; cosine of roll attitude
      s2 -- real, unitless ; sine of pitch attitude
      c2 -- real, unitless ; cosine in pitch attitude
      t2 -- real, unitless ; tangent of pitch attitude
      s3 -- real, unitless ; sine of yaw attitude
      c3 -- real, unitless ; cosine of yaw attitude
refby: UPDB

name: FILTIC
cont: Variables associated with NFF initial conditions
vars: Sdpic -- real, mixed, (11) (3) (8) ; vector of s.d. of the
diagonal elements of the NFF state initial estimation
      error covariance.

name: FILTRT
cont: Pointing vectors used by NFF
vars: mxrplf -- integer, unitless ; max. # sensor replications used in
      the NFF & FDI logic -- currently
      limited to 2.
      Ireplf -- integer, unitless, (13) (6) (7) ; vector of sensor
      replications used by
      the NFF (absolute
      sensor indexing)
      (Table 6.5)
      Inoutf -- integer, unitless, (17,2) (6,2) (7,2) ; matrix
      indicating
      status of all
      sensors in the
      NFF. Row index
      corresponds to
      absolute sensor
      type and col.
      index is
      replication of
      sensor. 1 =>
      active, -1 =>
      standby, 0 =>
      failed

- 72 -
refby: INITG, SUMIN, SUMOUT, GTOI, EKFNl, BIASF, BLEND, DET01, DET05, DET0, SETISN, ISOLAT, DECIDE, RECONF

ame: FLTIN (only in FINDS2)
cont: Vector array of sensor flight data
vars: Readin -- real, mixed, (-) (-) (26) ; dual replicated sensor flight data.
refby: READFL, INITXF

ame: GBLEND
cont: NFF blender gain matrix
vars: Vb0 -- real, mixed, (17,6) (3,3) (8,3) ; NFF blender gain
refby: BIASF, BLEND

ame: GRVYTC (not in FINDS2)
cont: Arrays needed to compute gravity vector which is appended to the input vector UFl
vars: GRavlc -- real, m/s , (3) (3) (-) ; skew symmetric compensation terms for runway frame w.r.t. inertial frame
TlcprT -- real, unitless, (3) (3) (-);
refby: GTOI

ame: GTOILC (not in FINDS2)
cont: Saved local variables in subroutine GTOI
vars: aloni -- real, radians ; constant longitude offset
alati -- real, radians ; constant latitude offset
ticpl -- real, unitless ; constant term in transformation matrix Tic
ticp2 -- real, unitless ; constant term in transformation matrix Tic
ticp3 -- real, unitless ; constant term in transformation matrix Tic
ticp4 -- real, unitless ; constant term in transformation matrix Tic
ticp5 -- real, unitless ; constant term in transformation matrix Tic
ticp6 -- real, unitless ; constant term in transformation matrix Tic
refby: GTOI

name: HEALCM
cont: Quantities used by the healer logic
vars: kcthir -- integer, unitless ; running count of elapsed samples since start of current healer window; value = [1, 60]
kmaxhrl -- integer, unitless ; total # of samples in (i.e., length of) healer window; value = 60
confbd -- real, unitless ; log of initial confidence bound (1/19) for the healer test
bthrsh -- real, mixed, (13) (6) (7) ; vector of largest expected normal operating biases for each sensor type -- absolute sensor index, Table 6.5

Pthrsh -- real, mixed, (13) (6) (7) ; vector of smallest expected failure levels for each sensor type (Table 6.5)

Dthrsh -- real, mixed, (13) (6) (7) ; vector of decision thresholds to be applied to each healer process. Dthrsh (i) = 2*Confbd*Phealt (i)**2 where Phealt contains s.d. of expected noise to be used only by healers (Table 6.5)

refby: INITG, NAV, RECONF, HEALR, LRTHLR

total # of sensors determined to be 'failed'
nfail -- integer, unitless ; # of new failures, i.e., incremental # of sensors just detected as failed in current iteration

nhealm -- integer, unitless ; max. # of sensors which can heal in one instant (i.e., dimension of Ihealp)

nheal -- integer, unitless ; total # of sensors which the healer logic has declared healthy at the end of a healer window

Ifailt -- integer, unitless, (13) (6) (7) ; vector containing absolute sensor type for each failed sensor. (Table 6.5) Whenever a sensor fails, its absolute sensor type is added to Ifailt -- hence, this vector is ordered by relative time of occurrence of failure.

Ifailr -- integer, unitless, (13) (6) (7) ; vector containing replication # for each failed sensor -- ordered same as Ifailt

Ihealp -- integer, unitless, (10) (6) (7) ; vector containing list of failed sensors which have healed. The value of an element is the index in Ifailt/Ifailr of the healed sensor.
name: IMLS (not in FINDS2)
cont: Quantities associated with earth rotation & thus on MLS frame rotation.
vars: rmagor -- real, m; radius of earth added to mean sea level altitude of MLS frame origin
splat -- real, radians; latitude of MLS frame origin
slon -- real, radians; longitude of MLS frame origin
sinlac -- real, unitless; sine of splat
coslac -- real, unitless; cosine of splat
Wrws -- real, unitless; skew symmetric form of angular vel. of runway w.r.t. inertial frame.

refby: FINDS/FINDS1, GTOI

name: INITVL
cont: Initial values for the NFF
vars: Inobps -- integer, unitless, (13) (6) (7); INOBPS=INOBP at start of run (showing which sensor biases are to be estimated) Table 6.5
Pbf0i -- real, mixed, (13) (6) (7); initial s.d. of bias estimation error (user units) Table 6.5
Pbfic -- real, mixed, (13) (6) (7); initial s.d. of isolator filters error information. (user units) (absolute sensor index) Table 6.5

refby: INITG, RECONF, CLPSIO, CLPSBE, RCOV

name: JUMPCM
cont: Variables for multi-frequency implementation of NFF.
vars: jmpcvx -- integer, unitless; # of iterations after which bias free covariance has to be computed
jmpcvb -- integer, unitless; # of iterations after which bias covariance has to be computed
jmpgnx -- integer, unitless; # of iterations after which bias free gain has to be computed
jmpgnb -- integer, unitless; # of iterations after which bias gain has to be computed
jiter -- integer, unitless; running counter of iterations or elapsed time ticks
jmdcx -- integer, unitless; mod (jiter, jmpcvx) = 0 ==> perform computations
jmdcb -- integer, unitless; mod (jiter, jmpcvb) = 0 ==> perform computations
jmdgx -- integer, unitless; mod (jiter, jmpgnx) = 0 ==> perform computations
jmdgb -- integer, unitless; mod (jiter, jmpgnb) = 0 ==> perform computations

refby: NAV, EKFM1, BIASF, BLEND, DET01, DET05, DET10, DECIDE, HEALR
name: LAOUT (not in FINDS1)
cont: Replicated accelerometer sensor measurements from flight data
vars: Axm -- real, m/s, (2) (-) (2) ; dual longitudinal accelerometer meas.
     Aym -- real, m/s, (2) (-) (2) ; dual lateral accelerometer meas.
     Azm -- real, m/s, (2) (-) (2) ; dual vertical accelerometer meas.
refby: READFL, SUMIN, HEALR

name: LATLON (not in FINDS2)
cont: Information regarding a/c latitude and longitude
vars: alat -- real, radians ; current estimate of a/c latitude
     alon -- real, radians ; current estimate of a/c longitude
     alatd -- real, rad/s ; current estimate of rate of latitude change
     alond -- real, rad/s ; current estimate of rate of longitude change
     csalat -- real, unitless ; cosine of alat
     snalat -- real, unitless ; sine of alat
refby: GTOI, SUMIN

name: LOCHEA
cont: Quantities local to subroutine HEALR
vars: nfail -- integer, unitless ; local snapshot of 'nfail'
     Ifailp -- integer, unitless, (20) (9) (11) ; local snapshot of 'Ifailt'
     Xsum -- real, mixed, (20) (9) (11) ; running sum over healing window length of difference between failed sensor and "working" sensor.
     Itest -- integer, unitless, (3) (3) (-) ; Local pointer vector for IMU healing logic
     Itestp -- integer, unitless, (3,3) (3,3) (-) ; Local pointer to store which parts of the IMU have healed
     Itest2 -- integer, unitless, (9) (9) (-) ; pointer vector to check that entire IMU heals as a unit
refby: HEALR

name: LRTINV
cont: Saved part of Kalman gain calculations from bias filter to be used by the detectors in the Chi-square test.
vars: Rtinv -- real, mixed, (17,7) (3,3) (4,4) ; saved \[CBFO*PBFO*CBFO + RBFO \]^**-1
refby: BIASF, DETO1, DETO5, DET10

name: LRTMAX
cont: Maximum Chi-square test thresholds to trip detectors
vars: vmax01 -- real, unitless ; max. threshold to trip DET01
     vmax05 -- real, unitless ; max. threshold to trip DET05
     vmax10 -- real, unitless ; max. threshold to trip DET10
refby: DETO1, DETO5, DET10

- 76 -
name: MAIN1
cont: Provides common dimensioning information for all 2-dimensional arrays and a scratch array for temporary use by all routines.
vars: ndim -- integer, unitless ; common row dimension for all arrays, value = (17) (6) (11)
      ndiml -- integer, unitless ; ndim + 1
      Dmfx -- real, temporary, (17,17) (6,6) (11,11) ; scratch area dimensioned 'ndim x ndim'
refby: INITG, SUMIN, GTOI, BIASF, UPDB, UPDQ, UPDP, ISOLAT, RCOV, VMPRT, MAT1A, MAT2, MAT3, MAT3B, MATXYT, MEQUAL, TRANS2, IMTCG2, MATCG2, MAINL2, MADD, MSUB, MATVEC, MATVC2

name: MAIN2
cont: Provides a temporary scratch area for use by all routines
vars: Com2 -- real, temporary, (17,17) (6,6) (11,11) ; scratch array dimensioned 'ndim x ndim'
refby: EKFN1, BIASF, BLEND, ISOLAT

name: MCONCO
cont: Conversion factors from user units to program units & vice versa
vars: Radian -- real, unitless ; conversion factor from degrees to radians
      Cnvrf -- real, unitless, (13) (-) (7) ; conversion factors from program units to user units for sensor signals - absolute sensor index. Table 6.5 (not used in FINDS1 because all conversions are radians to degrees)
refby: FINDS/FINDS1/FINDS2, READFL, INITG, INITXF, GTOI, UPDQ, DECIDE, TLOUT

name: MLOUT (not in FIND21)
cont: Replicated MLS sensor measurements from flight data
vars: Azim -- real, radians, (2) (-) (2) ; dual azimuth measurements
      Elem -- real, radians, (2) (-) (2) ; dual elevation measurements
      Rngm -- real, radians, (2) (-) (2) ; dual range measurements
refby: READFL, INITXF, SUMOUT, RESCMP, HEALR

name: MLSALL (not in FINDS1)
cont: Information regarding MLS antenna locations.
vars: Xaz -- real, m, (3) (-) (3) ; location of azimuth/DME antenna in the runway frame
      Xel -- real, m, (3) (-) (3) ; location of elevation/DME antenna in the runway frame
      xO -- real, m ; x-location of elev. antenna in MLS frame
      xO -- real, m ; y-location of elev. antenna in MLS frame
      zO -- real, m ; altitude offset between azimuth & elev. antennae
refby: FINDS/FINDS2, UPDH, UPDPH
name: MULTDT (not in FINDSl)
cont: Quantities used in detecting multiple simultaneous failures.
vars: Priorj -- real, mixed, (3) (-) (3) ; vector of log. of the prior
probability of more than one
MLS sensor of the same type
to fail in the same instant
(common mode failure).
(ordered MLS azimuth,
elevation, range)
Alandj -- real, mixed, (3) (-) (3) ; vector of log-likelihood of a
multiple MLS sensor failure.
(ordered same as Priorj)
Resbj -- real, mixed, (17,3) (-) (11,3) ; matrix of multiple MLS
failure compensated
residuals vectors.
Cols. are ordered as
azim., elev., rng.
refby: ISOLAT, DECIDE

name: NAMES
cont: Character variables which are vectors of sensor names & units
vars: Iyname -- character *9, (13) (6) (7) ; vector of sensor types,
Table 6.5
Iyunit character *5, (13) (6) (7) ; vector of sensor types,
Table 6.5
refby: READFL, DECIDE, HEALR

name: PSIR (not in FINDS2)
cont: Quantities associated with runway yaw
vars: psiru -- real, radians ; runway yaw w.r.t North
simpsr -- real, unitless ; sine of psiru
cospsr -- real, unitless ; cosine of psiru
refby: FINDS/FINDSl, INITXF, SUMIN, SUMOUT, GTOI, RESCMP, HEALR

name: PQRDEG (not in FINDS2)
cont: Computed "best" estimate of P, Q, R (in degrees) as average of all
available rate sensors, including standby equipment
vars: apdeg -- real, degrees ; roll rate estimate [(rep1 + rep2)/2]
tagdeg -- real, degrees ; pitch rate estimate
ardeg -- real, degrees ; yaw rate estimate
refby: GTOI, UPDQ

name: RDLOCL
cont: Saved local variables in subroutine READFL. In particular, the
saved variables are current sensor measurements to be used at the
next iteration and the maximum sensor differences for the data drop-
out tests.
vars: Axmold -- real, m/s , (2) (-) (2) ; longitudinal accel. previous
measurements
Aymold -- real, m/s , (2) (-) (2) ; lateral accel. previous
measurements
Azmold -- real, m/s$^2$, (2) (-) (2) ; vertical accel. previous measurements

Pmold -- real, rad/s, (2) (2) (-) ; roll rate gyro previous measurements

Qmold -- real, rad/s, (2) (2) (-) ; pitch rate gyro previous measurements

Rmold -- real, rad/s, (2) (2) (-) ; yaw rate gyro previous measurements

Aziold -- real, rad, (2) (-) (2) ; MLS azimuth previous measurements

Eleold -- real, rad, (2) (-) (2) ; MLS elevation previous measurements

Rngold -- real, m, (2) (-) (2) ; MLS range previous measurements

Airold -- real, m/s, (2) (-) (2) ; IAS previous measurements

Ph iod -- real, rad, (2) (2) (-) ; IMU roll previous measurements

Theold -- real, rad, (2) (2) (-) ; IMU pitch previous measurements

Psiold -- real, rad, (2) (2) (-) ; IMU yaw previous measurements

Axmax -- real, m/s$^2$ ; longitudinal accel. dropout threshold

Aymax -- real, m/s$^2$ ; lateral accel. dropout threshold

Azmax -- real, m/s$^2$ ; vertical accel. dropout threshold

Pmax -- real, rad/s ; roll rate gyro dropout threshold

Qmax -- real, rad/s ; pitch rate gyro dropout threshold

Rmax -- real, rad/s ; yaw rate gyro dropout threshold

Azimax -- real, rad ; MLS azimuth dropout threshold

Elemax -- real, rad ; MLS elevation dropout threshold

Rngmax -- real, m ; MLS range dropout threshold

Airmax -- real, m/s ; IAS dropout threshold

Phimax -- real, rad ; IMU roll dropout threshold

Themax -- real, rad ; IMU pitch dropout threshold

Psimax -- real, rad ; IMU yaw dropout threshold

name: RGOU T (not in FINDS2)
cont: Replicated rate gyro measurements from flight data
vars: Pm -- real, rad/s (2) (2) (-) ; dual roll rate gyro measurements
Qm -- real, rad/s (2) (2) (-) ; dual pitch rate gyro measurements
Rm -- real, rad/s (2) (2) (-) ; dual yaw rate gyro measurements
refby: READFL, SUMIN, GTOI, HEALR

name: SIGT AU (SIG in FINDS1)
cont: Design values for noise parameters used by NFF and detectors
vars: Sig -- real, mixed, (15) (6) (9) s.d. of sensor noise used by NFF (ordered as input sensors, winds, output sensors). Tables D, B
Tau -- real, seconds, (2) (-) (2) ; time constant for horizontal winds in wind model used by NFF
Sigd01 -- real, mixed, (15) (6) (9) ; s.d. of sensor noise for DET01, ordered same as SIG
Sigd05 -- real, mixed, (15) (6) (9) ; s.d. of sensor noise for DET05, ordered same as SIG
Sigd10 -- real, mixed, (15) (6) (9) ; s.d. of sensor noise for DET10, ordered same as SIG
refby: INITG, UPDQ, DECIDE, NOISR
Saved local variables in subroutine SUMIN. In particular, the input sensor measurements from the current iteration are saved to perform trapezoidal integration at the next iteration.

- **Axmo**: real, m/s, (2) (-) (2); saved longitudinal accel. measurements
- **Aymo**: real, m/s, (2) (-) (2); saved lateral accel. measurements
- **Azmo**: real, m/s, (2) (-) (2); saved vertical accel. measurements
- **Rmo**: real, rad/s, (2) (2); saved roll rate gyro measurements
- **Qmo**: real, rad/s, (2) (2); saved pitch rate gyro measurements
- **Rmo**: real, rad/s, (2) (2); saved yaw rate gyro measurements

Quantities associated with the program timing and synchronization.

- **dttime**: real, s; program integration step size \((1/20)\)
- **idtime**: integer, unitless; counter incremented at each iteration to compute \('time\).
- **time**: real, s; elapsed time from start of program
- **tstart**: real, s; program starting time (default = 0)
- **tstop**: real, s; program final time (estimated)
- **dt22**: real, s; saved \(dttime^2/2\)
- **idst05**: real, unitless; counter to stop/start DET05 after system reconfiguration following failure/healing.
- **idst10**: real, unitless; counter to stop/start DET10 after system reconfiguration following failure/healing.

Quantities associated with the inputs to the NFF

- **nu** : integer, unitless; total # of inputs to NFF including gravity inputs (default value = 9,3,6)
- **nul**: integer, unitless; total # of inputs to NFF associated with an input sensor (i.e., \(nu - ng\)), value = 6, 3, 3
- **nulpl**: integer, unitless; \(nul + 1\) = 7, 4, 4
- **nulc**: integer, unitless; \((nul) - (# of inputs not currently active)\)
- **Inoup**: integer, unitless (17) (6) (11); pointer vector to absolute input measurements used by NFF (Table 6.3). The array index corresponds to the location in uFl and the value is the abs. input meas. type index.
Ufl -- real, mixed, (9) (6) (6) ; vector of compensated inputs used by NFF (computed in SUMIN)  

refby: INITG, SUMIN, GTOI, EKFN1, BIASF, BLEND, DET01, DET05, DET10, SETISH, UPDB, UPDH, UPDPH, RESCMP, ISOLAT, DECIDE, RECONF, CLPSIO, NOISR, ADJTBP, HEALR  

name: SYSX1  
cont: Bias free filter state dimensions and system matrices  
vars: nx -- integer, unitless ; total # of states in bias free portion of NFF, value = 11, 3, 8 

nx1 -- integer, unitless ; nx + 1, value = 12, 4, 9 

Afl -- real, mixed, (11,11) (-) (8,8) ; constant state transition matrix. (Not defined in FINDS1 as it is an identity matrix there). 

Bfl -- real, mixed, (17,9) (3,3) (8,6) ; nonlinear input transition matrix (function of states). 

Ef1 -- real, mixed, (17,11) (3,3) (8,8) ; discrete process noise covariance matrix.  

refby: INITG, EKFN1, BIASF, BLEND, UPDB, UPDQ, UPDH, UPDPH, ISOLAT, RECONF, CLPSIO, ADJTBP, RCOV  

name: SYSXBO  
cont: Quantities associated with the bias filter portion of the NFF.  
vars: nb -- integer, unitless ; current # of biases estimated by NFF (nb = nub + nyb), value 6, 3, 3 

nub -- integer, unitless ; current # of input sensor biases estimated by NFF, value 6, 3, 3 

nyb -- integer, unitless ; current # of measurement biases estimated by NFF, value 0, 0, 0 

nub1 -- integer, unitless ; nub + 1, value = 7, 4, 4 

Inobp -- integer, unitless, (13) (6) (7) ; pointer vector to sensor type of each bias estimated. (absolute sensor index) (from Table 6.5)  

refby: NAV, INITG, SUMIN, EKFN1, BIASF, BLEND, UPDH, UPDPH, ISOLAT, RECONF, CLPSIO, CLPSBE, ADJTBP  

name: SYSYBO (not in FINDS2)  
cont: Variables common to subroutines EKFN1 and BIASF  
vars: Rbf0 -- real, mixed, (17),12) (3,3) (-) ; saved HPl*PFl*HPl T + R from EKFN1 

Cbf0 -- real, mixed, (17,6) (-) (-) ; bias filter observation matrix.  

refby: EKFN1, BIASF  

name: SYSWL  
cont: Quantities associated with the NFF observation and process noises.  

- 81 -
vars:      
  ny -- integer, unitless ; total # of averaged (or collapsed)
  measurements presented to the NFF, value
  = 7, 3, 4
  nymxi -- integer, unitless ; initial max. # of avgd. meas. to NFF,
  value = 7, 3, 4
  Inoyp -- integer, unitless, (17) (6) (11) ; pointer vector to
  active avgd. outputs
  used by NFF.  (array
  index corresponds to
  the elements of the
  measurement array &
  value of each element
  corresponds to
  absolute meas. index.)
  Table 6.2
  Inoypi -- integer, unitless, (17) (6) (11) ; inverse mapping of
  Inoyp, i.e., array
  index is abs. meas.
  index and value is
  the corresponding
  index in current
  meas. vector to NFF.
  If a particular meas.
  type is not used, its
  value entry will be
  zero.

Yfl -- real, mixed, (7) (3) (4) ; vector of avgd. meas. used by
  NFF (abs. meas. sensor indexing) Table 6.2
Qfl -- real, mixed, (8) (3) (5) ; vector of process noise
covariances organized by
absolute input index, Table 6.4
Hpl -- real, mixed, (17,17) (3,3) (4,8) ; effective observation
matrix for NFF (partial
of h w.r.t. x)
Rfld01 -- real, mixed, (7) (3) (4) ; vector of meas. noise
covariances used by DET01
(abs. meas. index). Table 6.2
Rfld05 -- real, mixed, (7) (3) (4) ; vector of meas. noise
covariances used by DET05
(abs. meas. index). Table
6.2
Rfld10 -- real, mixed, (7) (3) (4) ; vector of meas. noise
covariances used by DET10
(abs. meas. index).

refby:    INITG, SUMOUT, EFKM1, BIASF, BLENBD, DET01, DET05, DET10, UPDQ, UPDHH,
UPDPh, ISOLAT, CLPSIO, NOISR, ADJTBP
name:     TRBER
cont:     Transformation matrices for various reference frames
vars:     Trb -- real, unitless, (3,3) (3,3) (3,3) ; transformation matrix
from body axes into the
G-frame (for accel.
inputs).
Ter -- real, unitless, (3,3) (3,3) (-) ; matrix relating the body rates to the Euler angles (for gyro inputs).

 Tic -- real, unitless, (3,3) (3,3) (-) ; transformation matrix from runway frame to inertial frame.

refby: SUMIN, GTOI, UPDB, UPDQ

name: TSTORE (only in FINDSCMP)
cont: Temporary scratch areas (matrices) in EKFN1 and BIASF
vars: Tmp1 -- real, mixed, (17,17) (-) (-) ; local working array
      Tmp2 -- real, mixed, (17,17) (-) (-) ; local working array
refby: EKFN1, BIASF

name: UPDQLC
cont: Saved local variables in subroutine UPDQ
vars: scalef -- real, unitless ; s.d. of scale factor for rate gyro compensation
      spm -- real, unitless ; average error variance for rate gyro compensation (includes scale factor and misalignment errors)
      dt3 -- real, s ; saved dtime 3/3
refby: UPDQ

name: YOBSRV
cont: Scaling array for the filter observations
vars: Yscale -- real, mixed, (7) (3) (4) ; vector of scale factors used to scale each avgd. meas. into the NFF. Scaling is performed to ensure that the meas. noise variance is unity for each sensor. (indexed as per Table 6.2)

refby: INITG, SUMOUT, UPDH, UPDPH, RESCMP, ISOLAT, HEALR
9. REFERENCES


This report describes the operation and internal structure of the computer program FINDS (Fault Inferring Nonlinear Detection System). The FINDS algorithm is designed to provide reliable estimates for aircraft position, velocity, attitude, and horizontal winds to be used for guidance and control laws in the presence of possible failures in the avionics sensors.

The FINDS algorithm was developed with the use of a digital simulation of a commercial transport aircraft and tested with flight recorded data. The algorithm was then modified to meet the size constraints and real-time execution requirements on a flight computer. For the real-time operation, a multi-rate implementation of the FINDS algorithm has been partitioned to execute on a dual parallel processor configuration: one based on the translational dynamics and the other on the rotational kinematics. The report presents an overview of the FINDS algorithm, the implemented equations, the flow charts for the key subprograms, the input and output files, program variable indexing convention, subprogram descriptions, and the common block descriptions used in the program.
End of Document