% TeKLinearizedKalmanFilter.m

% This MATLAB macro computes INS Navigation using Position Fix Updates from

% OKSI Video sensor measurements to compensate for adverse Gyro Drift of

% the IMU. Additional preliminary computations are provided to 1st convert

% the AHRS into an INS (to which the Linearized Kalman Filter is then com-

% patible to represent what is going on internally by using an error model

% to obtain the estimation error). This model-based estimated error (as

% outputted from the Kalman filter) is then added to our result of

% implementing the navigation equations to obtain a final combined NAV

% solution (being the so-designated Aided Navigation result). It is this

% final result that is subsequently compared herein for proximity to the

% GPS data as "TRUTH". The final result reflects use of IMU, magnetometer

% sensor (see DISCLAIMER #1 below), and the OKSI Video Position fix data

% altogether. In making a proximity comparison, the GPS data is utilized.

%

% ASSUMPTION #1:

% In order to be compatible with any MatLab program, all variables and

% filenames generated within MaLab must START with a LETTER and may only

% contain letters, numbers, and the underscore character, "\_". Names are

% case sensitive and may be any length, but must be unique within the first

% 19 characters. Names should consist of only contiguous characters with no

% spaces in between. Thanks to modifications by Chris HolmesParker, we no

% longer need to use the MatLab Import Wizard herein. While the current

% approach avoids its use, we caution that the Import Wizard apparently ex-

% hibits some sensitivities if one attempts to choose a "Select Option" on

% the top right while the mouse pointer cursor is still an hour glass (thus

% indicating that it has not yet finished). The MathWorks should have

% disabled the mouse pointer input until after this operation had completed

% to be safe but evidently The MathWorks did not do so. This is the likely

% reason the MatLab Import Wizard occasionally baulks as a database file is

% being imported. (Cursor can only be viewed as an hour glass while cursor

% is hovering over particular panes; otherwise, it appears to be finished

% when it is not. That is the danger. Alternatively, USER may also view

% message at bottom of MatLab Use Interface screen to know when program is

% "busy" or "ready" for another command.)

%

% ALERT: We recommend (or assume) that operator has already performed a

% quick manual visual scan (as good programming practice) to cross-check

% database text files at an intermediate point (to confirm that there are in

% fact NO missing rows NOR other missing entries that would cause subsequent

% processing problems as a consequence).

%

% We NOW ignore ASSUMPTION #2 below since everything is now handled by

% OKSI beforehand. Our input data now comes with the two additional columns

% already added. We NO longer need to manipulate it and add to it ourselves.

%

% ASSUMPTION #2: During the Import Step using the MatLab Import Wizard

% (labeled Import Data as a menu item on the HOME tab), each file will have

% its name EXPLICITLY changed BY THE OPERATOR to be as follows:

% AA - the internal matrix designator for the AHRS/IMU data set.

% BB - the internal matrix designator for the OKSI Video fix data set.

% CC - the internal matrix designator for the GPS data set (to be used

% as a cross-check for universal validation).

%

% The TeKLinerazedKalmanFilter program assumes that the above procedure has

% already occurred or already been performed when it is initially invoked.

% During the IMPORT step, the original name appears in the middle of a

% slot (that only appears after the Matrix Option is clicked) about seven

% lines down from the top of the MatLab Import Wizard. The OPERATOR is to

% place his cursor pointer there and click in the slot to gain access and

% then (using the keyboard) merely changes the original name to the required

% NEW NAME indicated above. The original database is unaffected. Only the

% database that is imported into MatLab has this new name. (Alternatively,

% if operator forgets to change the filename while he is using the Import

% Wizard, he can always make the name change directly on the file where it

% appears in the Workspace (similar to how filenames may be changed within

% any Microsoft Windows Operating System).

%

% We NOW ignore ASSUMPTION #3 below since the 3 database text files are

% NOW loaded automatically by this program. (A new capability provided by

% Chris HolmesParker.)

%

% ASSUMPTION #3: All 3 database "text" files have already been imported

% into the MatLab Workspace using the MatLab Import Wizard (from the HOME

% tab) by clicking on Import Data, then (near top left of center) choosing

% "Matrix" option by clicking on it [rather than on the "Cell Array" option

% or on the "Column vector" option, which is the automatic default) and

% then, under the "Select Option" (the top right-most menu item) then clicks

% "Import Data". This must have already been performed for all three data-

% base data sets before invoking this TeKLinearizedKalmanFilter program at

% the MatLab prompt by merely typing its name via the keyboard.

%

% TACIT ASSUMPTION #4: Time epoch of AHRS/IMU data brackets OKSI Video Fix

% data but not neccessarily the GPS data. However, we will programmatically

% cross-check to verify this tacit assumption. We we alerted that GPS data

% does not conform to this assumption by occasionally extending beyond

% AHRS data in time. There is no problem with this situation since AHRS/IMU

% data length controls the processing. When AHRS/IMU data runs out our NAV

% processing stops.

%

% GPS data is NOW compared for proximity after AHRS/IMU and Video Position

% fix data processing is complete. If we were attempting to simultaneously handle

% this comparison in real-time, we would have needed to attempt handling GPS comparisons % along with the other prior processing. However, since it is not mandatory that the GPS % validation cross-check be performed in real-time, this GPS cross-comparison can be done % in a more straightforward manner as a subsequent test that is easier to understand and % implement (for greater ease in understanding).

%

% This MatLab program is to be initiated at the MatLab prompt by the USER

% merely typing its name (or starting with any alternate option that MatLab

% provides to RUN the program):

% TeKLinearizedKalmanFilter

% and processing will then commence. The 3 critical data sets are NOW

% automatically loaded into the MatLab Workspace from (or by) this program AFTER

% this program has been started/initiated.

%

% Scope: This MATLAB macro implements an INS and its associated

% Linearized Kalman Filter. Its outputs are:

% NAVOUT, KFCOV, KFEST, GPS, PROX

% (all appended to in MAIN so that they are all indexed on a common time).

%

% Usage:

% TeKLinearizedKalmanFilter

%

% Description of parameters:

%

% Operator is to enter only these following 3 external Inputs via keyboard:

% ndim - input, integer number of rows/columns of the System Matrix.

% (nominally 12, from Eq. 35 & 36).

% m1dim - input, integer number of rows of OKSI Video fix Observation OK,

% Please see further comments below.

% Matrix (nominally 3, from Eq. 54).

% m2dim - input, integer number of rows of magnetometer Observation OK

% Matrix (nominally 3, from Eq. 22 & 24).

%

% INTERNAL DEFINITION of variables utilized:

% t - continuous-time representation of time.

% ntindex - discrete-time index.

% FOUT - being the (ndim by ndim) System Matrix utilized herein,

% as hardwired within this program code (first expressed in

% continuous-time, t, and then converted to discrete-time,

% kindex, within this program code). The matrix entries are

% time-varying but its structural aspects remain constant.

% Transit - Computed discrete-time Transition Matrix (assumes entries of

% F are constant over small time intervals, a standard

% assumption).

% Z1 - OKSI Video Position fix measurement vector (m1dim by 1) OK

% I will use this OK outside to denote

% H1 - being the specific Observation Matrix of the OKSI Video fix OK

% subscript utilized is consistent with what is

% (hardwired within the code, being m1dim by ndim).

% as used in prior Progress Report.

% R1 - being the OKSI position fix measurement noise covariance OK

% At one time, we had a problem with this but no more.

% matrix (m1dim by m1dim symmetric positive definite matrix).

% (we currently do NOT use the attitude information

% that is also availed because we do not know how to do so)

% (m1dim by m1dim symmetric positive definite matrix).

% Z2 - magnetometer measurement vector (r2dim by 1). OK

% H2 - being the specific Observation Matrix of the Magnetometer OK

% fix (hardwired in the code and being m2dim by ndim).

% R2 - being the magnetometer measurement noise covariance matrix OK

% (m2dim by m2dim symmetric positive definite matrix).

%

% NAVOUT - Computed output consisting of Final Navigation Solution

% (and associated plots).

%KFESTAFTER - Kalman Filter Estimates after Measurement Updates are

% incorporated (and associated plots).

% KFCOV - Kalman Filter Covariances before and after sensor measurement

% updates exhibiting classic characteristic saw tooth pattern

% (and associated plots).

% GPS - GPS data to be used in a proximity comparison.

% PROX - results of the proximity comparison between NAVOUT and GPS

% (and associated plots).

% AA - the internal matrix designator for the AHRS/IMU data set.

% BB - the internal matrix designator for the OKSI Video fix data set.

% CC - the internal matrix designator for the GPS data set.

% The original database files that become AA, BB, and CC matrices

% have now been augmented by OKSI to have two additional columns at

% the far right: the first additional column being one that combines

% hours, minutes, and seconds into Total Aggregate Seconds and the

% second (last) additional column being an integer index that

% The above approach works as long as NO day, NO month, NOR year end boundary is crossed.

% This completes the enumeration of all the entries of the particular matrix types.

% These may be easily generalized to handle additional future designators that may arise % within the program. We chose this particular convention for simplicity in accomplishing % this task.

% The above AA, BB, and CC initially have different filenames but are

% converted to these aliases by the OPERATOR filling in the

% appropriate names where indicated within the “datainput.m” subroutine

% used before this program begins its processing.

%

% Pointers utilized:

% iindex - integer pointer for AA.

% jindex - integer pointer for BB

% kindex - integer pointer for CC

% iplusone - iindex + 1

% jplusone - jindex + 1

% kplusone - kindex + 1

% lengthofdata1 - total length of AHRS/IMU data (i.e., AA). This parameter

% controls most of the processing herein.

% lengthofdata2 - total length of Video Fix data (i.e., BB).

% lengthofdata3 - total length of GPS data (i.e., CC).

%

% We use letters for variables that are the most straight-forward and

% closest to their origin appearing in the Progress Report (as fairly

% standard notation), such as "H2" for the Observation matrix (i.e.,

% H\_m in Eq. 22 & 24) and "H1" for OKSI Video fix data (Eq. 54).

%

% Our original intent was to tie into the MatLab Kalman Filter code

% that we had purchased from Lawrence Lupash, as we had mentioned in our

% earlier progress reports. We now provide the entire Kalman Filter code

% ourselves instead as being a more straightforward software development

% path for us(after having had some problems with Lupash's code).

%

% User Defined

% Functions: SYSMAT - Calculates continuous-time System Matrix FOUT

% (alias F). Structure is time-invariant but

% individual elements vary with time.

% TRANSIT - Calculates Transition matrix used in forming

% the discrete-time System Matrix. Can have options

% of 3 term Taylor Series approximation or so-called

% exact, as provided by MalLab (which, ostensibly,

% could have an infinite number of terms, but

% practically uses a stopping condition internally

% that stops after a certain accuracy is achieved).

% (Trade-off: time consumed for each vs. accuracy.)

% TRANSIT is calculated using standard approximation

% which assumes that, over the short time intervals

% involved, the System Matrix, F, is treated as

% being essentially constant. So handling is

% actually as piece-wise constant. Time-varying

% nature is instantaneously captured within the F,

% progressively at each sample time starting each

% evaluation time interval.

% After discussions with Chris HolmesParker, we

% decided to make this a user-defined function as

% well.

% COVBEFORE - calculates P\_k|k-1 a.k.a. Pbefore.

% where COVPROPAGATESTEP: P\_k|k-1 = Transit P\_k-1|k-1 Transit^T

% + GAMMA Q GAMMA^T.

% In matching some literature (i.e., Jay Farrell's

% book) GAMMA is used in the role of "G", used more

% frequently everywhere else. So "G" is an alias of

% GAMMA (to be safe). This is a constant.

% KFGAIN - calculates the Kalman Gain a.k.a. Kgain

% where Kgain = Pafter H^T [H Pafter H^T + R]^(-1).

% COVAFTER - calculates P\_k|k a.k.a. Pafter,

% where COVUPDATESTEP: Pafter = [I-KH] Pbefore [I-KH]^T+K R K^T.

% ESTBEFORE - calculates x\_k]k-1,

% where ESTPROPGATESTEP: x\_k|k-1 = Transit x\_k-1}k-1.

% ESTAFTER - calculates x\_k|k,

% where ESTUPDATESTEP: x\_k|k = x\_k|k-1 + Kgain [z - H x\_k|k-1].

% datainput - automatically loads the three databases in the

% Workspace that is accessible to this program to

% read. This program does not modify their contents.

% appendMatrix - appends one additional matrix to the previous time-

% series of this same type.

% appendScalar - appends one additional scalar (this is time here)

% to the previous time-series of this same type.

% appendVector - appends one additional vector to the previous time

% -series of this same type.

%

% Last update: 04/18/2013

% Copyright (C) 2012-13 by TeK Associates. All Rights Reserved.

% Programmer: Thomas H. Kerr III, Ph.D. in E.E., TeK Associates,

% Lexington, Massachusetts, 8 April 2013.

%==========================================================================

% disp(' ')

% disp('REMINDER: To be compatible with any MatLab program, all variables')

% disp('and filenames generated within MaLab must START with a LETTER and')

% disp('may only contain letters, numbers, and the underscore character,')

% disp('"\_". Names are case sensitive and may be any length, but must be')

% disp('unique within the first 19 characters. Names should consist of only')

% disp('contiguous characters with no spaces in between. Thanks to modifi-')

% disp('cations by Chris HolmesParker, we no longer need to use the MatLab')

% disp('Import Wizard herein. While the current approach avoids its use, we')

% disp('caution that the Import Wizard apparently exhibits some sensitivi-')

% disp('ties if one attempts to choose a "Select Option" on the top right')

% disp('while the mouse pointer cursor is still an hour glass (thus indica-')

% disp('ting that it has not yet finished). The MathWorks should have dis-')

% disp('abled the mouse pointer input until after this operation had com-')

% disp('pleted to be safe but evidently The MathWorks did not do so. This')

% disp('is the likely reason the MatLab Import Wizard occasionally baulks')

% disp('as a database file is being imported. (Cursor can only be viewed as')

% disp('an hour glass while cursor hovering over particular panes; other-')

% disp('wise, it appears to be finished when it is not. That is the danger.')

% disp('Alternatively, USER may also view message at bottom of MatLab User')

% disp('Interface Screen to know when program is "busy" or "ready" for')

% disp('another command. Hit ENTER to continue, otherwise "exit".')

% pause % Program pauses here until ENTER key is pressed.

%

% %=================== UP-FRONT Full DISCLOSURE =============================

% %== 28 March 2013 TWO CAUTIONARY DISCLAIMERS (to be rectified soon) =====

% %

% These disclaimers will be deleted when they no longer apply (target is by 30 Apr).

% %=== DISCLAIMER #1:

% % In the above idealized discussion, HH1 should be compatible with the

% % states chosen to represent the system, where the first three states are

% % currently 3 coordinate POSITION. What I currently found in Jay A Far-

% % rell's textbook is instead compatible only with the first three states

% % being ATTITUDE states. I need to bridge the gap between what we have and

% % what we need. Similar statement applies for R1 for the same reason. It

% % may be easy to bridge this gap but I haven't done it yet. In the mean-

% % time, the safest course is to develop the proper structure in code herin

% % that will use it when posed compatibly (correctly) yet NOT yet use it as

% % it currently exists in the WRONG FORM. That means that, rather than

% % propagate and update using magnetometer measurements, I will temporarily

% % only be propagating the state at the higher periodic rate (shorter time-

% % step) which is tantamount to effectively assuming that magnetometer has

% % already been sufficiently accounted for in the AHRS data. This later

% % assumption was less far fetched when OKSI was sending me AHRS mode 3 data

% % (that included "everything" but recent data sent seems to be a more

% % parsimonious proper subset (perhaps obtained from a different data AHRS

% % mode now). I need to check back with OKSI about this issue. I brought it

% % up before but not recently. I didn't want to surprise them (with a bad

% % surprise) but I jumped over it in order to continue with other aspects

% % that needed to be addressed rather than get bogged down on this one

% % issue. It may still be quickly solved when I bring it up again. In the

% % remainder of this coding exercise, I will treat it as though everything

% % on this nit has already been successfully resolved.

% %

% % A useful temporary work-around was settled upon by TeK Associates: we

% % program the use of H1 and R1 (defined above), representing the magneto-

% % meter, so that this program code structure is exactly as it should be for

% % an Observation Matrix totally compatible with our current state selection

% % (reminder: with 3 position states leading off) but with the contents of

% % the H1 Observation Matrix to be identically zero (until we get the

% % correct replacement). We will use the current diagonal R1 being non-zero

% % (so that we do not encounter or incur attempts to invert singular

% % matrices). This is a nice compromise since everything is correctly in

% % place structurally but the sensor update step (for both Kalman Filter

% % estimate and its associated covariance) still occurs at the correct times

% % but, instead, effectively corresponds merely to a Propagate Step

% % (because the Update Step is effectively zeroed out due to H1 consisting

% % entirely of zeros).

% %

% % The current temporary bogus H1 needs to merely be replaced by the

% % compatibly-corrected version when/if it becomes available and then

% % everything will be satisfactorily resolved without any further program

% % architectural change being needed. This is a nice resolution with but a

% % mere change in parameter values.

% %

% %===== DISCLAIMER #2:

% % Another issue is units consistency. I first focus on structural correct- % The good news on this issue is that if we

% % ness of calculations, then go back later and thoroughly check units % properly handle units into and out of each

% % throughout. It is just a matter of proper scaling. Most (but not all) % subroutine and user-defined function, this

% % have already been followed through on. ALL must be done along these lines % will automatically be corrected and units

% % before it may be deemed correctly implemented. An easy but onerous task % will be consistent throughout.

% % but which has, so far, taken a back seat. Details, details. The devil is

% % always in the details.

% %=================== TIME-TABLE TO FIX ABOVE TWO DISCLAIMERS =============

% % Within the next week by Sunday 28 April 2013

% %

% % Once these two issues are successfully resolved, this entire FULL

% % DISCLOSURE box will be deleted from this code. Otherwise, it will

% % continue to appear here to truthfully convey the status.

% %

% %=====================END OF TWO CAUTIONARY DISCLAIMERS====================

%

% % This program is double precision by default - Covariance aspects handled

% % within therefore do not require use of U-D-U^T Squareroot filter

% % formulation (such is necessary only for situations involving extremely

% % long run times, which then requires a numerically stable implementation,

% % which is only availed by a squareroot filter formulation). For less

% % arduous applications, the two different formulations of "standard" and

% % "squareroot" produce identical results. Any attempted real-time implemen-

% % tations of these computations for navigation applications should seek to

% % use a squareroot filter formulation. Within this present algorithm herein

% % we forgo use of a squareroot formulation for simplicity and greater

% % readability since non-squareroot filter mechanizations are easier to

% % understand by being closer to what is presented in the majority of

% % standard historical textbooks as the constituent components of a standard

% % Kalman filter. To ameliorate somewhat for not using a squareroot formu-

% % lation, the Covariance calculations provided herein are still improved by

% % using a so-called "stabilized conventional formulation" (consisting

% % merely of adding it to its own transpose and dividing by 2 or, equiva-

% % lently, by multiplying by 0.5). Summarizing again, a conventional formu-

% % lation is easier to understand, code, and debug since it matches what is

% % presented in most textbooks as a Kalman Filter. The system matrix that is

% % used herein corresponds to a linearized model so we ultimately implement

% % a Linearized Kalman Filter herein. We found out that we did NOT need to

% % resort to use of an Extended Kalman Filter (EKF) after all so life is

% % easier in this regard and the interpretation of Kalman Filtering results

% % is more rigorous and achieved without having to invoke the numerous other

% % approximations associated with implementing an EKF (and worries about its

% % convergence (or lack thereof) or stability (or lack thereof) or less

% % rigor in the interpretation of covariances of an EKF. Computed covariances

% % of a KF have full rigor.

% disp(' ')

% disp('ALERT: Of necessity, there is INITIALLY some fundamental operator')

% disp('screen interaction. Later in the processing, the program takes over')

% disp('and everything is automated. Hit ENTER to continue.')

% pause % This pause allows operator to read prior message before proceeding.

% % Please forgive this initial overkill in explanations. Once operator

% % becomes accustomed to this programming style, the subsequent

% % comments can/will be breifer or merely abbreviated since their

% % purpose should be obvious (but I include them just in case they are

% % needed later).

% clc %Clear work pane.

% %clear %clears all memory (so nothing is left over from previous run).

% close all %Good programming practice.

% disp(' ')

% disp('Our pausing here gives USER an opportunity for saving previously')

% disp('generated figures before they get clobbered or over-written by')

% disp('figures generated within this computer run. We will next clear away')

% disp('any previously generated figures as a conservative step to avoid')

% disp('incurring any confusion (by avoiding having two figures with the')

% disp('same figure number). Click upper right (in figure frame that will')

% disp('appear next) to close it. Hit ENTER afterwards to continue.')

% pause % Program pauses here until ENTER key is pressed. This gives operator

% % an opportunity to collect and save any existing MaTlab figures from

% % a prior run to a safe folder that operator wishes to retain.

% clf %Clears all figures (to avoid confusion with results from earlier runs)

% disp(' ')

% disp('After closing the figure frame, hit ENTER to continue.')

% pause % Program pauses here until ENTER key is pressed.

% disp(' ')

% disp('If there is a need to ABORT the processing at any time,')

% disp('simultaneously press "CONTROL" and "c". Hit ENTER to continue.')

% pause % Program pauses here until ENTER key is pressed.

%

% % Linearized Kalman Filter for OKSI including further preprocessing of AHRS

% % to become an INS (rather than merely an AHRS) to which the Linearized

% % Kalman Filter then corresponds. This is a discrete-time Mechanization but

% % some entities such as System Matrix F and Process Noise Covariance

% % Intensity Matrix Q start out as being continuous-time. All 3 input

% % datasets are assumed to be periodic but not necessarily synchronized.

% % This software will properly account and compensate for the lack of

% % synchronization between these three input files.

%

% % The IMU within the AHRS (being a Crossbow AHRS400CD-200), is a strapdown

% % mechanized MEMS, and the model utilized for the Kalman-like Filter herein

% % is the full INS Error Model of Eq. 11.80, p. 396 of Jay A. Farrell's text

% % -book; with matrix entries as spelled out in Table 11.1 p. 397 (and with

% % gyro random-walk error terms included [beyond what is shown in Eq. 11.80]

% % in each of the pertinent input channels of the 3 gyros). This VERSION was

% % coded by Thomas H. Kerr III (TeK Associates) and much was carried over

% % from his preliminary work on this code that he performed for this project

% % on his other more limited capability 32-bit Vista OS HP Laptop PC. This

% % current version is running on a 64-bit Windows 7 OS Toshiba Laptop PC

% % and, as such, can accommodate much longer input files (2 Gigs.). (All 3

% % input files were originally provided to us by OKSI.)

%

% % A FilterOut matrix will be defined early on and be built up by adding

% % additional rows as processing progresses. The dimension of FilterOut will

% % ultimately be altered to accommodate processing associated with both IMU/

% % magnetometer and OKSI Video position fixes together. Once completed and

% % Kalman Filtering is finished, the pertinent outputs contained within

% % FilterOut will be automatically plotted.

%

% % This code assumes that OKSI Video Position Fix measurements are available

% % periodically and that there are no missing rows or missing columns. TeK

% % visually confirmed this assumption for the three OKSI-provided data files

% % that it used in testing this software for data processing.

%

% % The IMU data is also assumed to be periodic but more frequently available

% % at a higher rate (with a shorter period than OKSI Video data fixes) and

% % that, similarly, there are no missing rows or columns.

%

% % The total number of "Kalman Filter Update Steps" is exactly the number of

% % OKSI Video Position fixes (in time) availed as Input Data (obtained from

% % OKSI) plus each AHRS data step to account for magnetometer measurement

% % (also obtained from OKSI) unless a video fix and a magnetometer fix are

% % simultaneous (which we explicitly check for and properly take into

% % account by then having only one update step for both sensors).

%

% % REPEAT OF THE INFORMATION OF THE ABOVE PARAGRAPH FOR EMPHASIS (with

% % slightly revised wording):

% % The total number of Kalman Propagate Steps is the number of ARHRS data

% % (in time) PLUS 2 times the number of OKSI Video Position fixes (assuming

% % none are exactly simultaneous with the times of the AHRS data). In case

% % any are simultaneous, we built in a cross-check to detect this situation

% % if it occurs so that we still do the right thing (in this unlikely case)

% % and still obtain the correct total length of post-processed data).

%

% % We also have GPS data to be used as "TRUTH" (that can be input now at the

% % beginning before processing actually starts but NOW will not be used

% % until all the Kalman-like Filter Processing has been completed over

% % the entire time epoch covered by both the AHRS and OKSI Video position

% % fix data. There is no reason why GPS TRUTH data would be synchronous with

% % AHRS or OKSI Video Position fix data without being explicitly forced to

% % be so by use of a common clock for timing and being synchronized. We were

% % told that this is not the case. Therefore, in order to obtain a proper

% % "apples-to-apples" comparison of final Navigation results to test for

% % proximity to what GPS indicates is TRUTH, it would have to be done at a

% % common time for both. This can be accomplished by interpolating GPS data

% % to match the same points in time for which the processed Nav data is

% % available. A reasonable question to ask is why not instead interpolate

% % the computed Navigation data to match the times at which GPS TRUTH data

% % is available. The quick answer is that if we ever want to use some of the

% % creative ideas of Section 5 of the Progress Report to gauge proximity

% % then the associated computed covariances are also needed within the

% % comparison (as indicated in Section 5). Since covariances correspond to

% % ellipsoids with principal axes that move around with time, it is not

% % obvious how to interpolate within covariances properly so interpolation

% % needs to be performed on the GPS data instead as the most convenient and

% % practical option. Again, to be slightly redundant for emphasis, the GPS

% % input data was originally obtained from OKSI.

%

% % Contributing to the above decision to interpolate GPS truth data to the

% % designated comparison times, recall from Sec. 6 of TeK Associates Progress

% % Report that the computed Nav solution (of Eqs. 1 to 20) is to be combined

% % with (i.e., added to or subtracted from) the delta solution outputted by

% % the Kalman-like filter in order to obtain the properly corrected

% % Navigation solution to which the GPS data is eventually compared for

% % proximity.

% %

% % Restating as a summary of sorts:

% % The most straightforward way to perform a proximity test to GPS Truth

% % would be to wait until all the navigation data has been processed for the

% % entire time epoch of interest. This requires that we also output whole

% % covariances for the quantities that are to participate in the cross-

% % comparison for each time at which a comparison is to take place. We now

% % opt to output all this time-tagged covariance data and wait to apply the

% % proximity tests discussed above, at the end rather than apply it

% % incrementally, step-by-step, as we proceed with the NAV and Kalman Filter

% % processing. In all cases, we output results as files to be plotted at the

% % end for convenience in human interpretation. This appears to be the

% % clearer path that, unfortunately, involves larger output files and

% % additional processing.

% %

% % Additionally, a valid comparison to GPS data as a TRUTH reference must

% % only be performed for 4 or more satellites being available and further

% % only for good geometry existing between the GPS receiver and the 4 (or

% % more) satellites being instantaneously relied upon, which is evidenced in

% % high level summary fashion by the GPS data file exhibiting "good"

% % associated GDOP. Therefore, we must check for these benign conditions

% % being satisfied before endeavoring to cross-check processing results

% % against GPS as "TRUTH".

% %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%

% %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% % UTC Time in each of the three data sets supplied by OKSI includes \*

% % Year, Month, Day, Hour, Minutes, and Seconds as complete Time Tags, \*

% % each with its own separate column within the data, viewed as a matrix. \*

% % For most situations, this may, perhaps, be viewed as "overkill" unless \*

% % these boundaries are actually crossed during the course of a mission. \*

% % To simplify the situation, TeK Associates will limit its attention to \*

% % properly handling merely Hours, Minutes, and Seconds as sufficient \*

% % time-tags (after having 1st viewed each of the 3 data sets to confirm \*

% % that no Year, Month, and Day boundaries are crossed and that this more \*

% % limited approximation of ignoring Year, Month, and Day is apparently \*

% % adequate for this situation of the specific data utilized here). For \*

% % any missions where any of these Year, Month, or Day boundaries is \*

% % actually crossed during a mission, TeK Associates cautions that \*

% % additional coding needs to be provided and inserted to properly handle \*

% % this more challenging situation (that is still routine). We chose to \*

% % use this approach to both simplify the situation and to keep the size \*

% % of numbers encountered herein for "total number of seconds" included \*

% % within the new auxiliary time-tags more manageable by merely using: \*

% % 1 Hour = 60 minutes = 60 x 60 secs = 3600 secs. \*

% % 1 Minute = 60 secs. \*

% % We requested that OKSI use "Hours x 3600 + Minutes x 60 + Secs" as a \*

% % single time-tag of Total Seconds: \*

% % 3600\*Hours\_slot + 60\*Minutes\_slot + Seconds\_slot, as in the MatLab \*

% % snippet next below (pleas notice that columns for CC is incremented by \*

% % one since OKSI added a new first column without informing TeK): \*

% % AA(:,21)=3600\*AA(:,4)+ 60\*AA(:,5)+AA(:,6); \*

% % BB(:,49)=3600\*BB(:,4)+ 60\*BB(:,5)+BB(:,6); \*

% % CC(:,13)=3600\*CC(:,5)+ 60\*CC(:,6)+CC(:,7);- % CC has a new C(:,1) \*

% % OKSI now appends this new time-tag to each dataset Matrix as a "new" \*

% % next-to-last column to enable TeK to make proper cross-comparison \*

% % decisions (for the particular data at hand in these 3 input files from \*

% % OKSI). TeK also provide code-based internal screen cross-checks for \*

% % assessing sufficient simultaneity of the three data sets. Specifically,\*

% % that they each have time epochs (from beginning to end) that overlap \*

% % sufficiently and that the AHRS/IMU data encompasses at least all the \*

% % OKSI Video fix data (otherwise processing problems are encountered). \*

% % The last sentence is no longer a problem as of 26 April 2013 \*

% %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%

% %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% % OKSI NOW also appends a "new" final column of sequential integers as \*

% % separate indeces for each of the three data set matrices to aid us in \*

% % making decisions as we loop through the data. \*

% % \*

% % We are aware that Matrix-based MatLab abhors nested loops since such a \*

% % structure can cause MatLab computations to be extremely time consuming.\*

% % We try to limit it to only one control loop (at a time) appearing in \*

% % this program. \*

% % To enforce this constraint we instead rely on use of case statements: \*

% % "while...else...end" and "if...then, elseif..., elseif..., else... \*

% % end", and "while...end" and "for....end". \*

% %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%

% % The following inputs are NOT really necessary here since the mathematical

% % model is hard-wired within this version of the program. We do it as a

% % little reminder to the USER in case the model is changed later. This

% % constitutes immediate feedback to the USER on dimensions of current

% % internal model.

% disp(' ')

% ndim = input('Specify state dimension (ndim) i.e., it should be 12 --> '); % We probably want to fix these so that code

% % while (ndim ~= 12)

% complains here if nothing is provided by

% % disp(' ')

% USER when these numbers are requested.

% % disp('Input was NOT entered correctly. Try again!');

% This should be properly viewed as

% % ndim = input('Specify state dimension (ndim) i.e., it should be 12 --> ');

% "robustifying" this USER interaction.

% % end

% So far, it does not complain enough

% % disp(' ')

% when user merely hits ENTER only.

% % disp('Specify number of rows of magnetometer');

% % m2dim = input('Observation Matrix (m2dim) i.e., it should be 3 --> ');

% OK

% % while (m2dim ~= 3)

% % disp(' ')

% S/W should complain here if nothing is provided by USER when these numbers are

% % requested.

% % disp('Input was NOT entered correctly. Try again!');

% % disp('Specify number of rows of magnetometer');

% % m2dim = input('Observation Matrix (m2dim) i.e., it should be 3 --> ');

% % end

% % disp(' ')

% % disp('Specify number of rows of OKSI Video fix');

% % m1dim = input('Observation Maatrix (m1dim) i.e., it should be 3 --> ');

% % while (m1dim ~= 3)

% % S/W should Complain here if nothing is provided by USER when these numbers are

% % requested.

% % disp(' ')

% % disp('Input was NOT entered correctly. Try again!');

% % disp('Specify number of rows of OKSI Video fix');

% % m1dim = input('Observation Maatrix (m1dim) i.e., it should be 3 --> ');

% % end

%

% format long %attempting to handle a wide range of matrix values

%

% disp(' ')

% disp('Check to confirm that matrices AA, BB, and CC appear in the MatLab.')

% disp('Workspace a short time after USER hits RETURN key to continue.')

% pause % pauses will USER reads above message and responds.

% %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% % This is Chris HolmesParker's neat utility for inputting all 3 data- \*

% % bases into the MatLab Workspace and automatically changes the \*

% % names to what this MAIN program expects, namely AA, BB. and CC. \*

% % When USER gets a new mission to process with different file names \*

% % all USER needs to do is change the new names within the subroutine \*

% % named "datainput.m", then it will all be handled automatically with- \*

% % out any further USER intervention being required. \*

% datainput\_version2 % <------Version 2 of what is discussed here. \*

% % it converts degrees to radians too so it manipulates these files. \*

% %datainput %<---This useful utility was provided by Chris HolmesParker. \*

% % Thanks to OKSI, the 3 database files already have the last two new \*

% % colmns alreadt present so we no loner need to append them within \*

% % this program. \*

% %\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%

% disp(' ')

% % As a reasonableness cross-check use screen display to simultaneously

% % provide display screen overview of Year, Month, and Day for all 3

% % datasets.

% disp(' ')

% disp('Compare UTC Year, Month, and Day between datasets for compatibility')

% disp('For AA:');

% fprintf('%i %i %i ',AA(1,1),AA(1,2),AA(1,3));

% disp(' ')

% disp('For BB:');

% fprintf('%i %i %i ',BB(1,1),BB(1,2),BB(1,3));

% disp(' ')

% disp('For CC:');

% fprintf('%i %i %i ',CC(1,2),CC(1,3),CC(1,4));

%

% disp(' ')

% disp('Please scroll up if necessary to compare the UTC year, month, and')

% disp('day associated with all three databases that we have just read.')

% disp('Even though they have not yet appeared in the Workspace. They will')

% disp('appear soon. Hit ENTER to continue.')

% pause %check to see that the printouts from all 3 databases correspond

% lengthofdata1= length(AA);

% lengthofdata2 = length(BB);

% diag(' ')

% disp('First sample time step for AA in total seconds:');

% disp(' ')

% fprintf('%f ',AA(1,21));

% disp(' ')

% disp('First sample time step for BB in total seconds:');

% disp(' ')

% fprintf('%f ',BB(1,49));

% disp(' ')

% disp('Last sample time step for AA in total seconds:');

% disp(' ')

% fprintf('%f ',AA(lengthofdata1,21));

% disp(' ')

% disp('Last sample time step for BB in total seconds:');

% disp(' ')

% fprintf('%f ',BB(lengthofdata2,49));

% disp(' ')

% disp('Now check first 2 items above to see if first time step of AHRS/IMU')

% disp('(AA) data occurs before first time step of OKSI Video fix data')

% disp('(BB)? This condition holding had previously been a fundamental')

% disp('assumption for this program to work properly but now we can')

% disp('accommodate any situation without any problem as long as we have')

% disp('some time overlap between the two records AA and BB. If the')

% disp('assumption of possessing some time overlap is not met, program will')

% disp('automatically stop and tell USER the reason why. Please hit ENTER')

% disp('to continue.')

% pause %Check to see that printouts start time assumptions are satisfied.

%

% % \*\*\*\*\*\* Was originally going to Provide Global Constants here below: \*\*\*

% % Rather than rely on Global constants, TeK decided to explicitly pass\*

% % such variables directly as parameters to subroutines so units may be\*

% % changed as necessary within the routine. This would serve as a \*

% % systematic units cross-check as well. This was a needed cross-check.\*

% % Use of Globals can be problematic. Now we avoid the issue entirely. \*

% % Chris may decide to re-introduce Globals. He can handle them well. \*

% % End Global Constants \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% % The following were originally candidates to be Globals:

% % OMEGA = 15.041067 % degrees/hour. See Eq. 6 of Progress Report.

% % Careful! Chris now converts all degrees into % radians upon reading data the first time.

% % He actually modifies what is in our copy of the databases.

datainput;

OMEGA = Omega\_ie;

omega\_ie = Omega\_ie;

R0 = R\_0;

%OMEGA = 7.292115\*(10)^(-5); % radians/sec. See Eq. 6 of Progress Report

%R0 = 6378137; % meters. See Eq. 7 of Progress Report.

% Initial Constant Values being utilized before entering the Loop \*\*\*\*\*\*\*

% \*

% Initial conditions to be read from AHRS database to start NAV \*

% calculations (utilized further in Main Loop below): \*

% initial attitude, initial velocities, initial position (?) maybe from \*

% different AHRS mode. \*

% \*

% Since they are constant, specify H1, H2, R1, R2, Q, GAMMA (= G), \*

H2 = [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;

% Notice that by making this matrix all zeroes,

% 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;

% it serves as a placeholder. Update Step using

1. 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0];

% Eq. 24 (this does nothing). At least it doesn't hurt

% and suffices for now since process noise covariance intensity matrix is

% positive definite and invers to be encountered in Update Step is well-behaved.

%

% % If, as Chis has suggested, we use AHRS data that has already been

% % temperature-compensated and, moreover, already vertically damped with

% % magnetometer data, we never need to change this but it is a slightly

% % heavier computational burden. (So you will likely want to eventually

% % remove it in the long term.)

%

%

%

%

R2 = [4.0 0.0 0.0; % Magnetometer Measurement noise Covariance intensity matrix.

0.0 4.0 0.0;

0.0 0.0 4.0];

% Eq. 25 units are (nt/rt-Hz)^2 (check these units with

% Alex, who provided them).

H1 = [1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0; % OKSI Video Fix

0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0; % Observation Matrix

0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0];

% Eq. 55 (check units).

% x\_1|0

xhat = [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]';

% Eq. 52 any % Use something other than "x" to avoid confusing with Chris’

% computed NAV.

% Initial x0 will suffice since Kalman Filter will quickly converge to right

% answer anyway (i.e., being exponentially asymptotically fast,

% when system is "Controllable" and "Observable"). Not true for arbitrary

% general Ordinary Differential Equations (ODE).

% P\_1|0 or Initial P or P0 next: \*

Pinit = [1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0; % Initial P Eq. 51.

0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;

0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;

0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;

0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0;

0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0];% Any positive

% definite initial P0

% suffices since it

% will quickly

% converge to the

% right answer anyway.

G = [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0;

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0]';%Eq. 53.

% (Thanks Chris!)

%

GAMMA = G; % In matching one notation, we used G. In matching another

% standard notation, we used GAMMA. They are the same and are constant

% in this application.

%

Q = [20.25 0.0 0.0; %Process (or Plant) Noise Covariance Intensity Matrix.

0.0 20.25 0.0;

0.0 0.0 20.25]; % Eq. 53.

%

% Need to adaptively calculate Delta within the loop since it will be

% different when Video Fix data is encountered but otherwise the same.\*

% \*

%Since System F matrix is time-varying it must be called inside the loop\*

% System matrix F is handled or resides in SYSMAT. \*

% Perform Loop initializations last. \*

%\*\*End Initialization of System Parameters that are Constant throughout\*\*

% \*\*\*\*\*\*\*\*\*\* Placeholder for Chris' NAV code if BATCH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% This section of code does not need to exist

% \* % if Chris' NAV solution is within the Loop

% Candidate location for Chris' ODE code (or function call) for solving \*

% below (as recommended).

% simultaneous system of ODE's representing velocity equations of NAV \*

% to be an INS using R-K 3rd order (or whatever method he settles on). \*

% It should either be before the main loop for a batch solution or within the main loop

% for an incremental solution.

% If only solved as BATCH, this should appear before the main loop, and \*

% it will cover the entire mission epoch provided. \*solution but it % shouldn't appear at both locations. Just at one place

% He had discussed use of Simpson's Rule, Euler, and possibly switching\*

% The better location is within the loop because there the Video fix can be used

% to damp the vertical channel.

% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Chris' NAv code if BATCH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% I will provide my own Kalman Filter code throughout rather than have to

% debug someone else's code too (referring to that of Lupash that I had

% been attempting to utilize before). I have no worries about Chris' code.

% I have not yet finished but should finish this week of 28 April 2013.

% Yes, I know that you have heard that before but now all that remains is

% completing this MAIN and I know how to finish it off now. The simple user

% defined functions are now completed (and provided except for this one and

% for Proximity test to be done next in line, then we are through).

%\*\*\*\* Initialize counters and pointers and read initial conditions \*\*\*\*\*\*\*\*

% Reminder here of definitions and symbols to be used, as defined at start \*

lengthofdata1 = length(AA);

lengthofdata2 = length(BB);

% iindex - integer pointer for AA.

% jindex - integer pointer for BB

% kindex - integer pointer for CC

% iplusone - iindex + 1

% jplusone - jindex + 1

% kplusone - kindex + 1

% lengthofdata1 - total length of AHRS/IMU data (i.e., AA). This parameter

% controls most of the processing herein.

% lengthofdata2 - total length of Video Fix data (i.e., BB).

% lengthofdata3 - total length of GPS data (i.e., CC).

%

%\*\*\*End of Initialize counters and pointers and read initial conditions \*\*\*

% % Move down BB until the first time sample (in Total Seconds) where BB time

% % sample is later or higher/greater than AA time sample.

% jindex = 1;

% while AA(1,21) >= BB(jindex,49)% comparing data start times and moving % This approach works as long as we don't

% if AA(1,21) < BB(jindex,49) % down BB until an acceptable 1st sample % cross any day, month, or year boundary.

% continue % time is found so that the processing can % It may be easily generalized by merely.

% end % commence/begin. Results displayed after % adding the requisite time to Total Seconds

% jindex = jindex + 1; % following CASE statements (accomplished % currently placed as next to last column

% end % with "if" and "elseif" statements below). % of each of the 3 datasets to be compared.

%

% disp('A graphic depiction of the original relationship between AHRS/IMU')

% disp('data (AA) and OKSI Video fix data (BB) follows next:')

% if (BB(1,49) < AA(1,21)) && (AA(lengthofdata1,21)< BB(lengthofdata2,49)) % Condition #1 AA nested in BB

% disp(' ')

% disp(' \_\_\_\_\_\_\_\_\_\_BB\_\_\_\_\_\_\_\_ ')

% disp('| |')

% disp(' \_\_\_\_\_\_AA\_\_\_\_\_ ')

% disp(' | | ')

% elseif(BB(1,49) > AA(1,21)) && (AA(lengthofdata1,21)> BB(lengthofdata2,49))% Condition #2: BB nested in AA

% disp(' ')

% disp(' \_\_\_\_\_BB\_\_\_\_\_\_ ')

% disp(' | | ')

% disp(' \_\_\_\_\_\_\_\_\_\_AA\_\_\_\_\_\_\_\_ ')

% disp('| |')

% elseif(BB(1,49) < AA(1,21)) && (AA(lengthofdata1,21)> BB(lengthofdata2,49))% Condition #3: BB start shifted left of AA

% disp(' ')

% disp(' \_\_\_\_\_BB\_\_\_\_\_\_')

% disp('| |')

% disp(' \_\_\_\_\_\_AA\_\_\_\_\_ ')

% disp(' | | ')

% elseif(BB(1,49) > AA(1,21)) && (AA(lengthofdata1,21)< BB(lengthofdata2,49))% Condition #4: AA start shifted left of BB

% disp(' ')

% disp(' \_\_\_\_\_\_BB\_\_\_\_\_')

% disp(' | |')

% disp(' \_\_\_\_\_\_AA\_\_\_\_\_ ')

% disp('| | ')

% elseif(AA(1,21) > BB(lengthofdata2,49)) % Condition #5: BB entirely left of AA

% disp(' ')

% disp(' \_\_\_\_\_\_\_\_\_\_BB\_\_\_\_\_\_\_\_\_')

% disp('| |')

% disp(' \_\_\_\_\_\_AA\_\_\_\_\_ ')

% disp(' | | ')

% disp('Time segments of AA&BB fail to overlap at all. Nothing can be done')

% disp('Time sements are disjoint so no further processing is possible now.')

% return

% elseif(AA(lengthofdata1,21) < BB(1,49)) % Condition #6: AA entirely left of BB

% disp(' ')

% disp(' \_\_\_\_\_\_\_\_\_\_BB\_\_\_\_\_\_\_\_')

% disp(' | |')

% disp(' \_\_\_\_\_\_AA\_\_\_\_\_ ')

% disp(' | | ')

% disp('Time segments of AA&BB fail to overlap at all. Nothing can be done')

% disp('Time sements are disjoint so no further processing is possible now.')

% return

% end

%

% %=========Cleaned up above results prior to entering fundamental ==========

% %=========for...if...elseif...elseif block below==========================

% diag(' ')

% disp('We have now used a neat loop to advance a pointer (jindex) down the')

% disp('rows of BB to the first UTC time sample encountered after the first')

% disp('UTC time sample of AA. The first measurement in BB at this time')

% disp('will be used to update AA and both will be properly advanced')

% disp('subsequently until AA runs out (or BB runs out before AA) but')

% disp('processing of AA will continue to its very end.')

% disp('Coarse Cross-Check: First sample time step for AA in total seconds:')

% disp(' ')

% fprintf('%f ',AA(1,21));

% disp(' ')

% disp('Value obtained for jindex:')

% disp(' ')

% fprintf('%i ',jindex);

% disp(' ')

% disp('First BB(jindex,49):');

% disp(' ')

% fprintf('%f ',BB(jindex,49));

% disp(' ')

% %==========================================================================

% Current value (not original value) of jindex is what we start with in the

% Main Loop below we found the proper intial value for jindex via the

% simple loop above. The adjusted start within BB is the first time point

% within BB following the start of AA.

% We place necessary "if" statements and logic here to invoke Kalman Filter

% operations and handling of actual mission data. DONE!!! This critical

% portion, which is the centerpiece of our solution approach, follows NEXT.

%=========for...if...elseif...elseif block follows now====================

% Weighting - weighting (>=0) used:

% use weighting = 1 for AHRS/IMU to AHRS/IMU time-step.

% use weighting = MU for ARDS/IMU to Video fix time-step.

% use weighting = OneMinusMU for Video fix to AHRS/IMU time-step.

% Delta - scalar time step provided (>= 0).

nN = length(AA);

% Intial Condition for Kalman filter state estimate already established

% above as xhat.

xbefore = xhat;

xafter = xhat;

P = Pinit; % Initial condition for associated Kalman filter covariance.

for iindex = a\_index:nN-1 % This is the main loop where all important

% logic will reside.

% nN = lengthofdata1 = length(AA).

% Subtract 1 since we will be using

% iPlusOne below and, otherwise, this will

% hit lengthofdata1 since we need to avoid

% going beyond the end of data

% (i.e., out of bounds).

iPlusOne = iindex + 1;

%disp(' ') % For test only. Remove afterwards %===================

%disp('iplusone is:') % For test Only. Remove afterwards. % || Optional Sanity Check.

%fprintf('%i ',iplusone); % For test Only. Remove afterwards. %===================

if (AA(iPlusOne,21)-AA(iindex,21)) > (BB(jindex,49)-AA(iindex,21)) % This condition being True indicates Next

% time sample is from OKSI Video fix.

% Propagate Step then Update Step (above) using OKSI fix data then

% do computations indicated in existing user-defined functions

% Append appropriate data after processing using Chris' 3 append

% utilities.% and and Propagate Step & Update Step twice in succession ONLY for this "If" case.

Mu = (BB(jindex,49)-AA(iindex,21))/(AA(iPlusOne,21)-AA(iindex,21)); % This variable is mu, appearing in Eqs.

% 87 and 89.

% PROPAGATE STEP comes now:

% Evaluate System Model in F at the new start time (SYSMAT which contains

% continuous-Time system dynamics capturing the time variation) that will be used further.

% Define all inputs for what is called next.

% phi - current latitude expressed in radians (be careful since elsewhere

% it is likely in degrees). Remember to Scale by pi/180 herein.

% In some equations within the Status Report, we let L represent

% latitude (as in Equations 1, 5, 11).

% Need to calculate OMEGA\_N and OMEGA\_D from appropriate Equations (not

% yet specified in Progress Report. Need to do so. It will use OMEGA from

% Eq. 6 (needs units)

% R0 - radius of the earth (needs units) from Eq. 7.

% v\_N - already computed (elsewhere) North velocity (needs units).

% V\_E - already computed (elsewhere) East velocity (needs units).

% V\_D - already computed (elsewhere) Down velocity (needs units).

% omega\_ie is as labled in the subscript as in Eq. 5 (needs units).

% f\_N - North component of specific force (needs units specified).

% f\_E - East component of specific force (needs units specified).

% f\_D - Down component of specific force (needs units specified).

% g - local gravity in meters/seconds^2 (check units) from Eq. 10.

[Lat\_IMU(iindex,1), Long\_IMU(iindex,1), h\_IMU(iindex,1), v\_N\_IMU(iindex,1), v\_E\_IMU(iindex,1), v\_D\_IMU(iindex,1)] = ...

IMU\_to\_INS\_single\_step(f\_N\_IMU, f\_D\_IMU, f\_E\_IMU, v\_N\_IMU, v\_E\_IMU, v\_D\_IMU, Omega\_ie, zeta, eta, R\_0, h\_IMU, Lat\_IMU, Long\_IMU,time\_IMU,iindex);

phi = Lat\_IMU(iindex,1);

[FOUT] = SYSMAT(phi,R0,v\_N,v\_E,v\_D,omega\_ie,f\_N,f\_E,f\_D,g);

% Calculate Delta (best to actually use time points for this from AA in case there is a hiccup).

delta = (AA(iPlusOne,21)-AA(iindex,21));

weighting = Mu;

F = FOUT;

% Evaluate Transition matrix used in discrete-time System matrix (uses results from SYSMAT).

[Transit] = TRANSIT(F,weighting,delta);

% Propagate Step for estimate (ESTBEFORE):

[xbefore] = ESTBEFORE(Transit,xafter);

% Propagate Step for covariance (COVBEFORE):

[Pbefore] = COVBEFORE(Transit,Pafter,G,Q,weighting,delta);

% Compute Kalman Gain (KFGAIN):

[Kgain] = KFGAIN(H1,Pafter,R1);

% Update Step goes here.

% Upgate Step for estimate (ESTAFTER):

[xafter] = ESTAFTER(xbefore,Kgain,z1,H1);

% Update Step for covariance (COVAFTER):

[Pafter] = COVAFTER(Pbefore,Kgain,H1,R1);

% NOW Chris' 3 APPEND Utilities need to be invoked, as appropriate:

% Matrix Append (invoked more than once) for sawthooth P+ & P- and for P+ alone.

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

% Vector Append (invoked more than once) for same reasons as above.

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

% Scalar Append (invoked more than once) for accompanying time (one for sawtooth and one for P+ and xhat+ only).

[timeVector] = appendScalar(time, timeVector);

[timeVector] = appendScalar(time, timeVector);

%disp(' ') % For test only. Remove afterwards. %===================

%disp('MU is:') % For test Only. Remove afterwards. % || Optional Sanity Check.

%fprintf('%f ',Mu); % For test only. Remove afterwards. %===================

% with different alterations of scalar time step parameter delta.

% \*\*\*\*\*\*\*\*\*\* Placeholder for Chris' NAV code if incremental \*\*\*\*\*\*\*\*\*\*\*\*\*

% (This is currently its temporary resting place at this Candidate location for Chris' % % ODE code (or function call) for solving \* % moved up to its place within the if

% simultaneous system of ODE's representing velocity equations of NAV above.)

% to be an INS using R-K 3rd order (or whatever method he settles on). \*

% It should either be within this loop for incremental solution or

% If only solved incrementally, by the end of this MAIN loop, before this loop

% it will cover the entire mission epoch provided. It shouldn't appear at both locations. % Just at one place or the other.

%

%

% We need to subtract KF estimate from Chris' NAV solution after \*

% OKSI Video update. This may suffice in damping the vertical channel \*

% as well or else use GPS or an altimeter does (success depends on its relative \*

% accuracy). \*

% He had discussed use of Simpson's Rule, Euler, and possibly switching between them\*

% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Chris' NAv code if incremental \*\*\*\*\*\*\*\*\*\*\*\*\*

OneMinusMu =(1-Mu); % Weight factor, appearing in Eqs. 88 and 90.

%disp(' ') % For test only. Remove afterwards. %===================

%disp('OneMinusOne is:') % For test only. Remove afterwards. % || Optional Sanity Check.

%fprintf('%f ',OneMinusMu); % For test only. Remove afterwards. %===================

% PROPAGATE STEP comes now:

% Evaluate System Model in F at the new start time (SYSMAT which contains

% continuous-Time system dynamics capturing the time variation) that will be used further.

% Define all inputs for what is called next. Nix for this case! See next comment.

% Since it doesn't change from what it was above, this doesn't need to be called for this case:

% [FOUT] = SYSMAT(phi,R0,v\_N,v\_E,v\_D,omega\_ie,f\_N,f\_E,f\_D,g);

% Calculate Delta (best to actually use time points for this from AA in case there is a hiccup).

delta = (AA(iPlusOne,21)-AA(iindex,21));

weighting = OneMinusMu;

% Evaluate Transition matrix used in discrete-time System matrix (uses results from SYSMAT).

[Transit] = TRANSIT(FOUT,weighting,delta); %THis needs to be present here since now weighting is different.

% Propagate Step for estimate (ESTBEFORE):

[xbefore] = ESTBEFORE(Transit,xafter);

% Propagate Step for covariance (COVBEFORE):

[Pbefore] = COVBEFORE(Transit,Pafter,G,Q,weighting,delta);

% Compute Kalman Gain (KFGAIN):

[Kgain] = KFGAIN(H1,Pafter,R1);

% Update Step comes now:

% Upgate Step for estimate (ESTAFTER):

[xafter] = ESTAFTER(xbefore,Kgain,z1,H1);

% Update Step for covariance (COVAFTER):

[Pafter] = COVAFTER(Pbefore,Kgain,H1,R1);

% NOW Chris' 3 APPEND Utilities need to be invoked, as appropriate:

% Matrix Append (invoked more than once) for sawthooth P+ & P- and for P+ alone.

% THK needs to specify input and output file names for each of 6 Append utilities below.

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

% Vector Append (invoked more than once) for same reasons as above.

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

% Scalar Append (invoked more than once) for accompanying time (one for sawtooth and one % for P+ and xhat+ only).

[timeVector] = appendScalar(time, timeVector);

[timeVector] = appendScalar(time, timeVector);

% Then propagate from this Video fix time point to very next AA sample

% This Propagate Step uses different scalar

% time point at BB(jindex,49) then perform and Update Step.

jindex = jindex + 1;

elseif (AA(iPlusOne,21)-AA(iindex,21)) < (BB(jindex,49)-AA(iindex,21))

% Being True indicates Next time sample is

% Propagate Step then Update Step using magnetometer fix data then

% from magnetometer (AHRS/IMU) data Only.

% do computations indicated in existing user-defined functions.

% Append appropriate data after processing using Chris' 3 append

% utilities.

% PROPAGATE STEP comes now:

% Evaluate System Model in F at the new start time (SYSMAT which contains

% continuous-Time system dynamics capturing the time variation) that will be used further.

% Define all inputs for what is called next.

% Define all inputs for what is called next.

% phi - current latitude expressed in radians (be careful since elsewhere

% it is likely in degrees). Remember to Scale by pi/180 herein.

% In some equations within the Status Report, we let L represent

% latitude (as in Equations 1, 5, 11).

% Need to calculate OMEGA\_N and OMEGA\_D from appropriate Equations (not

% yet specified in Progress Report. Need to do so. It will use OMEGA from

% Eq. 6 (needs units)

% R0 - radius of the earth (needs units) from Eq. 7.

% v\_N - already computed (elsewhere) North velocity (needs units).

% V\_E - already computed (elsewhere) East velocity (needs units).

% V\_D - already computed (elsewhere) Down velocity (needs units).

% omega\_ie is as labled in the subscript as in Eq. 5 (needs units).

% f\_N - North component of specific force (needs units specified).

% f\_E - East component of specific force (needs units specified).

% f\_D - Down component of specific force (needs units specified).

% g - local gravity in meters/seconds^2 (check units) from Eq. 10.

[Lat\_IMU(iindex,1), Long\_IMU(iindex,1), h\_IMU(iindex,1), v\_N\_IMU(iindex,1), v\_E\_IMU(iindex,1), v\_D\_IMU(iindex,1)] = ...

IMU\_to\_INS\_single\_step(f\_N\_IMU, f\_D\_IMU, f\_E\_IMU, v\_N\_IMU, v\_E\_IMU, v\_D\_IMU, Omega\_ie, zeta, eta, R\_0, h\_IMU, Lat\_IMU, Long\_IMU,time\_IMU,iindex);

phi = Lat\_IMU(iindex,1);

[FOUT] = SYSMAT(phi,R0,v\_N,v\_E,v\_D,omega\_ie,f\_N,f\_E,f\_D,g);

% Calculate Delta (best to actually use time points for this from AA in case there is a hiccup).

delta = (AA(iPlusOne,21)-AA(iindex,21));

weighting = 1;

% Evaluate Transition matrix used in discrete-time System matrix (uses results from SYSMAT).

[Transit] = TRANSIT(FOUT,Weighting,Delta);

% Propagate Step for estimate (ESTBEFORE):

[xbefore] = ESTBEFORE(Transit,xafter);

% Propagate Step for covariance (COVBEFORE):

[Pbefore] = COVBEFORE(Transit,Pafter,G,Q,Weighting,Delta);

% Compute Kalman Gain (KFGAIN):

[Kgain] = KFGAIN(H2,Pafter,R2);

% Update Step comes now:

% Upgate Step for estimate (ESTAFTER):

[xafter] = ESTAFTER(xbefore,Kgain,z2,H2);

% Update Step for covariance (COVAFTER):

[Pafter] = COVAFTER(Pbefore,Kgain,H2,R2);

% NOW Chris' 3 APPEND Utilities need to be invoked, as appropriate:

% Matrix Append (invoked more than once) for sawthooth P+ & P- and for P+ alone.

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

% Vector Append (invoked more than once) for same reasons as above.

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

% Scalar Append (invoked more than once) for accompanying time (one for sawtooth and one for P+ and xhat+ only).

[timeVector] = appendScalar(time, timeVector);

[timeVector] = appendScalar(time, timeVector);

elseif (AA(iPlusOne,21)-AA(iindex,21)) == (BB(jindex,49)-AA(iindex,21))

% Two simultaneous fixes are called for

% One Propagate Step then Update Step from both (in either order) then

% from magnetometer and from OKSI video Fix.

% do computations indicated in existing user-defined functions.

% (i.e., they are simultaneous so they occur

% Append appropriate data after processing using Chris' 3 append

% at the same time). This is less likely to

% utilities.

% actually occur but should not be overlooked.

% Anything can happen with real data.

% PROPAGATE STEP comes now:

% Evaluate System Model in F at the new start time (SYSMAT which contains

% continuous-Time system dynamics capturing the time variation) that will be used further.

% Define all inputs for what is called next.

% Define all inputs for what is called next.

% Define all inputs for what is called next.

% phi - current latitude expressed in radians (be careful since elsewhere

% it is likely in degrees). Remember to Scale by pi/180 herein.

% In some equations within the Status Report, we let L represent

% latitude (as in Equations 1, 5, 11).

% Need to calculate OMEGA\_N and OMEGA\_D from appropriate Equations (not

% yet specified in Progress Report. Need to do so. It will use OMEGA from

% Eq. 6 (needs units)

% R0 - radius of the earth (needs units) from Eq. 7.

% v\_N - already computed (elsewhere) North velocity (needs units).

% V\_E - already computed (elsewhere) East velocity (needs units).

% V\_D - already computed (elsewhere) Down velocity (needs units).

% omega\_ie is as labled in the subscript as in Eq. 5 (needs units).

% f\_N - North component of specific force (needs units specified).

% f\_E - East component of specific force (needs units specified).

% f\_D - Down component of specific force (needs units specified).

% g - local gravity in meters/seconds^2 (check units) from Eq. 10.

[Lat\_IMU(iindex,1), Long\_IMU(iindex,1), h\_IMU(iindex,1), v\_N\_IMU(iindex,1), v\_E\_IMU(iindex,1), v\_D\_IMU(iindex,1)] = ...

IMU\_to\_INS\_single\_step(f\_N\_IMU, f\_D\_IMU, f\_E\_IMU, v\_N\_IMU, v\_E\_IMU, v\_D\_IMU, Omega\_ie, zeta, eta, R\_0, h\_IMU, Lat\_IMU, Long\_IMU,time\_IMU,iindex);

phi = Lat\_IMU(iindex,1);

[FOUT] = SYSMAT(phi,R0,v\_N,v\_E,v\_D,omega\_ie,f\_N,f\_E,f\_D,g);

F = FOUT;

% Calculate Delta (best to actually use time points for this from AA in case there is a hiccup).

delta = (AA(iPlusOne,21)-AA(iindex,21));

weighting = 1;

% Evaluate Transition matrix used in discrete-time System matrix (uses results from SYSMAT).

[Transit] = TRANSIT(F,weighting,delta);

% Propagate Step for estimate (ESTBEFORE):

[xbefore] = ESTBEFORE(Transit,xafter);

% Propagate Step for covariance (COVBEFORE):

[Pbefore] = COVBEFORE(Transit,Pafter,G,Q,weighting,delta);

% Compute Kalman Gain (KFGAIN):

[Kgain] = KFGAIN(H1,Pafter,R1);

% Update Step comes now:

% Upgate Step for estimate (ESTAFTER):

[xafter] = ESTAFTER(xbefore,Kgain,z1,H1);

% Update Step for covariance (COVAFTER):

[Pafter] = COVAFTER(Pbefore,Kgain,H1,R1);

% Compute Kalman Gain (KFGAIN):

[Kgain] = KFGAIN(H1,Pafter,R1);

% Update Step comes now:

% Upgate Step for estimate (ESTAFTER):

% Compute Kalman Gain (KFGAIN):

[Kgain] = KFGAIN(H2,Pafter,R2);

[xafter] = ESTAFTER(xbefore,Kgain,z2,H2);

% Update Step for covariance (COVAFTER):

[Pafter] = COVAFTER(Pbefore,Kgain,H2,R2);

% NOW Chris' 3 APPEND Utilities need to be invoked, as appropriate:

% Matrix Append (invoked more than once) for sawthooth P+ & P- and for P+ alone.

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

[N\_by\_N\_by\_t\_Matrix] = appendMatrix(N\_by\_N\_Matrix, N\_by\_N\_by\_t\_Matrix,N);

% Vector Append (invoked more than once) for same reasons as above.

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

[N\_by\_1\_by\_t\_Vector] = appendVector(N\_by\_1\_Vector, N\_by\_1\_by\_t\_Vector,N);

% Scalar Append (invoked more than once) for accompanying time (one for sawtooth and one for P+ and xhat+ only).

[timeVector] = appendScalar(time, timeVector);

[timeVector] = appendScalar(time, timeVector);

end

end

return % temporary end while debugging above handling of 3 database inputs % Temporary End (this will be removed

% (to eventually be removed from final version). % afterwards).

% (We successfully verified up to this point.)

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* REMINDER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Now that Kalman Filter results have been computed for the entire time

% epoch of concern (i.e., the mission length as conveyed in the ARHs/IMU

% data, we now have some appended processed data that is extra long:

% lehgth(AA)+length(BB). With time (in Total seconds) appended in its own

% accompanying file that is just as long. Before we subtract Kalman Filter % This section of code does not need to exist

% Estimates from Chris's NAV solution, the addend and the subtrahend must % if Chris' NAV solution is within the above

% be of the same length. Chris said that I should leave that operation to % loop (as recommended).

% him. I gave him a few candidate ways to accomplish that task in an e-mail

% but will leave it up to him to implement it in the way he sees fit.

%\*\*\*\*\*\*This is but a placeholder and reminder to do so \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

disp('Starting the proximity test as a comparison against GPS as "TRUTH".')

disp('Hit RETURN to continue.')

pause % Briefly pause to allow USER to read message above.

%\*\*\*\* Initialize counters and pointers and read initial conditions \*\*\*\*\*\*

% \*

lengthofdata3 = length(CC);

% iindex - integer pointer for AA.

% jindex - integer pointer for BB

% kindex - integer pointer for CC

% iplusone - iindex + 1

% jplusone - jindex + 1

% kplusone - kindex + 1

% lengthofdata1 - total length of AHRS/IMU data (i.e., AA). This parameter

% controls most of the processing herein.

% lengthofdata2 - total length of Video Fix data (i.e., BB).

% lengthofdata3 - total length of GPS data (i.e., CC).

%

% \*\*End of Initialize counters and pointers and read initial conditions \*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Initiate working with GPS data (to be used as a Truth in subsequent \*

% cross-comparisons) and groom to use in this program, as prompted by \*

% output display. \*

% (Take Transpose, Unpack, and Avoid Discriptive Designator Labels) \*

% (Modified to fit our situation) \*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% After inspecting the GPS data, it was noticed that "Number of \*

% Satellites" is also included as well as GDOP. This is important to \*

% utilize before relying on GPS as a TRUTH reference. \*

% \*

% In order to have good accuracy from GPS, it should be the case that \*

% 1 < GDOP < 6. HDOP (a measure of horizontal position accuracy) is a \*

% constituent component of GDOP and so is VDOP (a measure of vertical \*

% position accuracy). As is TDOP for TIME. \*

% \*

% Insights relating to these measures are availed in: \*

% http://en.wikipedia.org/wiki/GDOP \*

% \*

% GPS position needs a minimum of 4 satellites and even with 4, if the \*

% geometry is bad (as gauged by associated GDOP being bad) then GPS \*

% position estimates should not be relied upon as "TRUTH". \*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Now unpack it only as necessary.

% UTC Year, UTC Month, UTC Day, UTC Hour, UTC Minute, UTC Sec,

for lidendex = 1:lengthofprocessedAppendeddata % This is secondary main

% loop for comparing our NAV and

% Kalman Filter processing

% results to GPS for proximity.

% "lengthofproccessedAppendeddata"

% is Not length of processed co-

% variance, which can be longer

% due to doubleing up, which

% occurs with the saw tooth

% behavior.

% We need to make certain that 4 or more satellites are utilized and that

% GDOP is good. If we only have HDOP, then rely on that instead.

% We have been warned that GPS data may extend beyond AHRS/INS Data, but

% that should not hurt us since our data processing should still be no

% longer than AHRS/IMU data final time and automatically stop when this

% dataset ends.

end

% Now that all the processing is completed, we need to avail plotting here.

% Alternatively, if we don't clear Workspace upon ending the program,

% we can allow the user to pick and choose what to plot.

% To ease USER's burden, we should offer candidte suggestions.

% This is where we should output commands that will allow USER to

% "copy and paste" in order to plot what USER wants!

disp(' ')

disp ('Processing is almost finished now. Once it finishes, USER can')

disp('again access MatLab Prompt to obtain plots since we do not clear')

disp('out the Workspace. To be USER-friendly, we will output commands')

disp('that USER can copy-and-paste to obtain specific plots of interest.')

disp('Program may display commands that we request that USER copy and')

disp('paste at the prompt in order to proceed a-step-at-a-time.')

disp('All USER needs to do is follow what is explicitly requested on the')

disp('screen at the prompt and this program will automatically handle the')

disp('rest internally, otherwise, there would be too much danger in')

disp('personal USER interpretation of how best to proceed. It is')

disp('easier and safer for the program to have these steps already worked')

disp('out beforehand and built-in for the USER in order to avoid likely')

disp('ambiguity in handling. Please hit ENTER key AGAIN to continue.')

disp(' ')

disp(' ')

disp(' ')

pause % Program pauses here until ENTER key is pressed.

fclose('all');

disp('Linearized Kalman Filter Program now ending. Hit ENTER to finish.')

disp('Afterwards, plotting can begin using our suggestions. Hit RETURN to')

disp('end program.')

pause

return %Ends the linearized Kalman Filter program

% [Pbefore] = COVBEFORE(Transit,Pafter,G,Q,Weighting,Delta)

% Calculates the appropriate (12 by 12) Kalman Filter covariance matrix

% P\_k|k-1 at the Propagate Step.

% P\_k|k-1 = Transit\*Pafter\*Transit^T+GAMMA\*Q\*GAMMA^T\*Delta (pseudo code)

% Inputs:

% Transit - Transition matrix for the indicated time-step.

% Pafter - Updated covariance at previous time step k-1|k-1.

% G - Gain Matrix for Process noise.

% Q - Process noise covariance intensity matrix.

% Weighting - weighting (>=0) used:

% use weighting = 1 for AHRS/IMU to AHRS/IMU time-step.

% use weighting = MU for ARDS/IMU to Video fix time-step.

% use weighting = OneMinusMU for Video fix to AHRS/IMU time-step.

% Delta - scalar time step provided (>= 0).

% Outputs:

% P\_k|k-1 - Predicted covariance one time step ahead. Units correspond to

% those of state variable selection; however, same units are

% squared herein.

function [Pbefore] = COVBEFORE(Transit,Pafter,G,Q,Weighting,Delta)

P = (Transit\*Pafter\*Transit') + (G\*Q\*G')\*Weighting\*Delta;

Pbefore = (0.5)\*(P+P'); %Stabilized covariance to slightly improve behavior

end

% [Pafter] = COVAFTER(Pbefore,Kgain,H,R)

% Covariance Update Step

% Output:

% Pafter - P\_k|k (12 by 12) Computed Kalman Filter Update Step Covariance,

% where COVUPDATESTEP: Pafter = [I-KH]\*Pbefore\*[I-KH]^T+K\*R\*K^T.

% Inputs:

% H - Observation Matrix (3 by 12).

% Pbefore - Covariance (12 by 12) after Propagation Step being P\_k|k-1.

% R - Sensor Measurement Noise Covariance Intensity Matrix (3 by 3)

% must be symmetric, positive definite (>0).

% Kgain - Kalman Filter gain matrix (12 by 3).

function [Pafter] = COVAFTER(Pbefore,Kgain,H,R)

I = eyes(12,12);

FACTOR = (I-K\*H);

P = (FACTOR\*Pbefore\*FACTOR')+(Kgain\*R\*Kgain');

Pafter = (0.5)\*(P+P'); % Stabilized covariance to slightly improve behavior

end

% [xbefore] = ESTBEFORE(Transit,xafter)

% Calculates ESTPROPGATESTEP: x\_k]k-1,

% where x\_k|k-1 = Transit\*x\_k-1}k-1. (pseudo-

% code).

% Inputs:

% Transit - discrete-time System Matrix (consisting of continuous-time

% transition matrix).

% xafter - Kalman Filter estimate following the last Update Step.

% Output:

% xbefore - Calculates Kalman Filter estimate at the Propagate Step.

function [xbefore] = ESTBEFORE(Transit,xafter)

xbefore = Transit\*xafter;

end

% [xafter] = ESTAFTER(xbefore,Kgain,z,H)

% ESTAFTER - calculates x\_k|k (12 by 1) vector,

% where ESTUPDATESTEP: x\_k|k = x\_k|k-1 + Kgain [z - H x\_k|k-1]

% (pseudo-code).

% Inputs:

% xbefore - Kalman Filter state estimate (12 by 1) state vector obtained

% as previous propagate step.

% Kgain - Kalman gain matrix (12 by 3) calculated at previous time-step.

% H - Sensor Observation Matrix (3 by 12).

% z - sensor measurement (3 by 1) vector realization at this time

% point.

% Output:

% xafter - Kalman Filter state estimate (12 by 1) after incoporating the

% sensor measurement realization.

function [xafter] = ESTAFTER(xbefore,Kgain,z,H)

xafter = xbefore + (Kgain\*(z - H\* xbefore));

end

% [Kgain] = KFGAIN(H,Pafter,R)

% Calculates the Kalman Gain a.k.a. Kgain where

% Kgain = Pafter\*H^(T)\*[H\*Pafter\*H^T + R]^(-1). (pseudo-code);

% where ^T is matrix transpose and ^(-1) is matrix inverse.

% Output:

% Kgain - Computed Kalman Filter gain matrix (12 by 3).

% Inputs:

% H - Observation Matrix (3 by 12).

% Pafter - Covariance (12 by 12) after Propagation Step being P\_k|k-1.

% R - Sensor Measurement Noise Covariance Intensity Matrix (3 by 3)

% must be symmetric, positive definite.

function [Kgain] = KFGAIN(H,Pafter,R)

Intermediate1 = (H\*Pafter\*H'+R);

Intermediate2 = inv(Intermediate1);

Kgain = Pafter\*H'\*Intermediate2;

end

% [Transit] = TRANSIT(FOUT,Weighting,Delta) - Calculates Transition matrix

% used in forming the discrete-time System Matrix.

% Can have options of 3 term Taylor Series approxi-

% mation or so-called "exact", as provided by MalLab

% (which, ostensibly, could have an infinite number

% of terms, but practically uses a stopping condi-

% tion internally that stops after a certain accu-

% racy is achieved).

% (Trade-off: time consumed for each vs. accuracy.)

% TRANSIT is calculated using standard approximation

% which assumes that, over the short time intervals

% involved, the System Matrix, F, is treated as

% being essentially constant. So handling is

% actually as piece-wise constant in time. The true

% Time-varying nature is instantaneously captured

% within the F, progressively at each sample time

% starting each evaluation time interval.

% After discussions with Chris HolmesParker, we

% decided to make this a user-defined function as

% treated herein. Chris may convert MatLab version

% to C-language to enable considerable speed-up.

% Output:

% Transit - Computes Transition matrix (using the Matrix Exponential).

% Inputs:

% FOUT - Continuous-time System matrix (computed within SYSMAT).

% Weighting - weighting (>=0) used:

% use weighting = 1 for AHRS/IMU to AHRS/IMU time-step.

% use weighting = MU for ARDS/IMU to Video fix time-step.

% use weighting = OneMinusMU for Video fix to AHRS/IMU time-step.

% Delta - scalar time step provided (>= 0).

% Pseudo-code: Transit = I + (FOUT)\*Delta + (0.5)\* F\*F\*(Weighting\*Delta)^2.

function [Transit] = TRANSIT(FOUT,Weighting,Delta)

IntermediateCalc1 = FOUT\*Weighting\*Delta;

IntermediateCalc2 = IntermediateCalc1\*IntermediateCalc1;

IntermediateCalc3 = eyes(12,12);

Transit = IntermediateCalc3 + IntermediateCalc1 + IntermediateCalc2\*(0.5);

% If MatLab internal capability were used instead, then ...

% Transit = expm(IntermediateCalc1);

% We expect FOUT to be (12 by 12) so Transit will likewise be (12 by 12).

end