24 March 2017

TO: KalmanFilter archive: workflow document
FROM: William Cooper
SUBJECT: workflow for generating the Kalman filter Tech. Note

1 Purpose

This workflow description documents the steps leading to the code in "KalmanFilterTechNote.Rnw" and provides additional detail not in the report "KalmanFilterTechNote.pdf." KalmanFilterTechNote.Rnw contains both text (in LaTeX format) and R processing script for the analyses in the resulting report on the development of a Kalman filter for use with measurements from the NSF/NCAR Gulfstream V research aircraft. The description of workflow provided here includes the process of collecting the observations and processing them to data files, the data archives used, the steps required to generate the plots and other results including the instances where manual intervention is required to identify appropriate subsets of the data, the relevant R code and LaTeX documents, and all the steps leading to the generation of the text in the technical note. "KalmanFilterTechNote.Rnw" is the definitive reference, but this overview and these diagrams will help explain the workflow at a general level and so should substitute for reading the R and LaTeX code in most cases. The intent is to describe the workflow in sufficient detail to support replication of the analysis and figures presented in the report, to facilitate changes based on new data or new analysis approaches, and to make it practical to apply the proposed algorithms to data collected by research aircraft used in atmospheric studies.

The document is intended for publication as an NCAR Technical Note. Some parts may be suitable to accompany a separate journal article that describes alternate and simpler methods for correcting attitude angles. Both this document and that proposed publication augment the material presented in the NCAR Technical Note "Uncertainty in Measurements of Wind from the NSF/NCAR Gulfstream Aircraft" Cooper et al. (2016).

2 Acquisition of the primary data

The measurements used in this report were collected using the NSF/NCAR GV research aircraft during the DEEPWAVE project of 2014. The onboard data-acquisition program 'aeros' recorded the data in digital format, and those data files were then processed by the program 'nimbus' to produce an archive in NetCDF format. The software management group of NCAR/EOL maintains a version-controlled archive of these programs, so if they are of interest they can be obtained by contacting the data-management group of EOL (at this or this address). The data files available from NCAR/EOL can be found at links on this URL. The details of the processing algorithms including those for the calculation of wind are documented in this report on Processing Algorithms and in Lenschow and Spyers-Duran (1989). These procedures as they pertain to the measurement of wind are also documented in Cooper et al. (2016). The resulting data files contain measurements

in scientific units and brief descriptions of each measurement, included as netCDF attributes for the files and for each variable.

One special file was generated and used in KalmanFilterTechNote.Rnw, a netCDF file named DWIRUrf15HR.nc. It was generated by the program nimbus (version of January 2016) by specifying a 25-Hz data rate and by including the variables needed for the "mechanization" described in this report. That file is not part of the standard project archives and so must be obtained separately, as described in the "Reproducibility" appendix to the report, but as used it is also saved as KalmanFilterTechNote.Rdata and can be used in that condensed form in the reference program.

3 The KalmanFilterTechNote.Rnw file

The .Rnw file is basically LaTeX text, generated for simplicity using LyX and exported to .Rnw format and then processed in RStudio (RStudio (2009)). The .lyx file (KalmanFilterTechNote.lyx) will run equivalently and produce a PDF-format version of the manuscript. Within the .Rnw file or within the .lyx file there are "chunks" of R code (R Core Team (2016)), delineated by a header having this format:

```
<<title, var=setting, ...>>=
...R code...
@
```

These chunks generate plots and other results of analyses that are incorporated into the manuscript using 'knitr' (Xie (2013, 2014)). In RStudio, the chunks appear as gray sections in the file when it is edited. Where tasks involve execution of R code, the chunk containing the code is referenced in the discussion below. Any results from the processing can be incorporated into the LaTeX text via "\Sexpr{}" calls embedded in the LaTeX portion of the file.

Two "switches" serve to speed execution of the code: (1) "ReloadData," which when TRUE causes the original archive data files to be read; and the option "CACHE" which, when true, causes previously generated subsets of the results to be re-used as saved in a subdirectory "cache." Both are provided to speed execution in cases where small changes are made. When ReloadData is FALSE subsets of data files previously saved as "Rdata" files are restored to R data.frames, a process that is much faster than re-reading the original archive data file. If cache is FALSE all calculations are performed, but using "cache=TRUE" is usually much faster after the first run. To ensure a clean run, set CACHE to be FALSE and remove all entries from the "cache" subdirectory.

After an introductory section, the program and the report are organized into three main sections: (1) calculation of the derivative of the state vector (position, aircraft velocity, and aircraft attitude angles) from the measured accelerations and rotation rates, leading to a trial "mechanization" for validation by comparison to the INS-provided solution; (2) a set of special topics that deal with component studies like improvement in the variables used for rate of climb and angle of attack

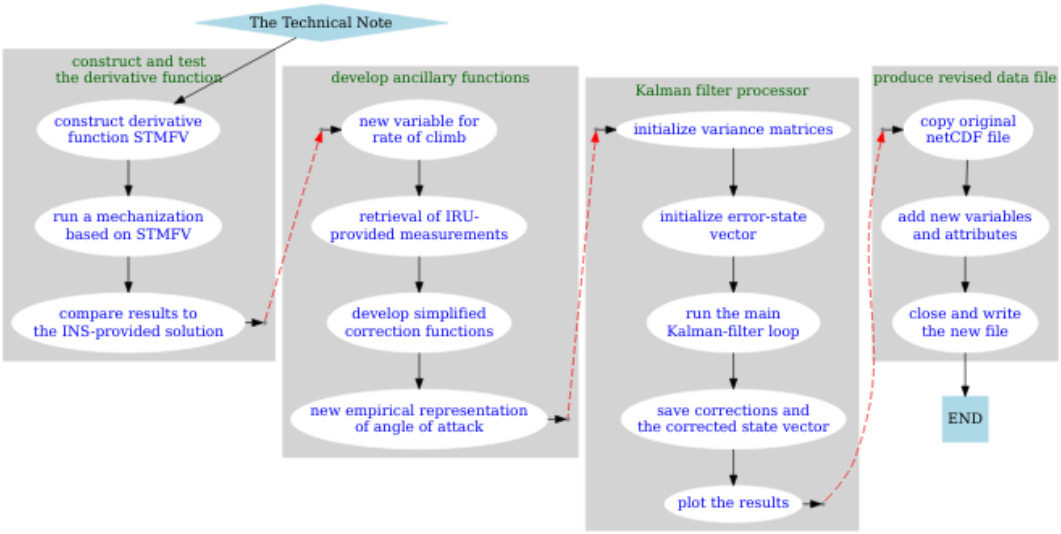


Figure 1: Top-level workflow diagram for the Kalman filter technical note. Some items (e.g., "run the main Kalman-filter loop") are described in additional workflow diagrams below.

and retrieval of the original measurements from data archives where they are missing; and (3) a description of the Kalman filter itself, with illustrative results. These sections are discussed in detail in this workflow document, which is in turn generated by a LyX file ("WorkflowKalmanFilter.lyx"). There is additional material and code at the end that adds the new variables to the data archive after correction by the Kalman filter.

A top-level workflow diagram for the Kalman filter technical is presented in Fig. 1. The Table of Contents in the technical note provides an overview of the structure of the document, while Fig. 1 focuses more on the logical flow of the routine that generates both the technical note and a new data file containing the measurements as corrected by the Kalman filter.

Some sections (mechanization, retrieval of IRU-provided measurements, simplified correction functions, and the Kalman-filter loop) have their own workflow diagrams in the sections where they are discussed.

4 Required R packages including Ranadu

The R code used for analysis reported in this paper relies heavily on a package of routines for R called "Ranadu." This is a set of R scripts for working with the NetCDF archive data files produced by NCAR/EOL/RAF, and it includes some convenience routines for generating plots

and performing other data-analysis tasks with the archived data. The Ranadu package is available at this GitHub address. To run the R code referenced here, that package should be included in the R installation. The KalmanFilterTechNote.Rnw routine requires that package and also some others referenced in the file, including "knitr", "ggplot2", "grid", "ggthemes", "zoo", "signal" and "numDeriv". In addition, Ranadu requires "ncdf4", "tcltk" and "stats". Some parts of "Ranadu" reference additional packages as needed, but they are not used in KalmanFilter.Rnw so do not need to be available for this routine to run.

The data processing for this manuscript involved revising some parts of the Ranadu package, as listed below. The relative timing among measurements is particularly important in this study, so some development of utilities to aid in studies of timing was useful. The netCDF files sometimes have mixed rates, e.g., 5_Hz for GPS-provided measurements and 25 Hz for some others including interpolation to 25 Hz from 13 Hz for IRS-provided measurements. In "Ranadu", this is handled appropriately in the "getNetCDF()" routine, sometimes with the "ShiftInTime()" routine to adjust for delays among the variables and the "SmoothInterp()" routine to interpolate for missing values and smooth the variables. Most of the plots are generated using "ggplotWAC()" or "plotWAC()", which are simple convenience routines that set various options preferred by the author before calling the R "ggplot()" or "plot()" routines.

4.1 Ranadu::getNetCDF () modifications

To handle data files with mixed-rate variables (e.g., 5-Hz GPS but 25-Hz IRS and others at 1 Hz), this script for reading the netCDF data files incorporates code to produce a single-rate R data.frame. For this purpose, there is a function "IntFilter()" in that script that interpolates and filters variables. The RAF data archives follow the convention that each sample is tagged with a time that is the *beginning* of the time interval, not the center. For example, for 50-Hz samples averaged to one second, the recorded variable represents the average of 50 samples beginning at the specified time and so should be interpreted as a sample average at 0.5 s past that specified time. The "IntFilter()" routine observes this convention by interpolating a low-rate sample to the specified higher rate and then shifting the resulting time series forward in time by half the original time interval, with duplication of the first measurement to fill the initial 1/2-period slots and also replication of the trailing 1/2-period slots that are not filled by the linear interpolation routine. Tests verified that this provided an appropriate representation of the measurement as interpolated to the higher rate and shifted forward to match other higher-rate measurements.

4.2 Ranadu::ShiftInTime ()

To shift time series variables forward or backward, this new function was added to the "Ranadu" package. The function interpolates the supplied time series to a higher rate (125 Hz), uses the R function "stats::approx()" for linear interpolation, shifts the interpolated series an appropriate number of bins forward or backward (duplicating or truncating the end values as necessary), optionally

applies a Savitzgky-Golay smoothing filter, and then returns an appropriate subset to represent the shifted time series at the original sample rate.

4.3 Ranadu::XformLA ()

At many places the mechanization and Kalman-filter algorithms require transformation from the aircraft or *a*-frame reference coordinates to a local-level Earth-relative frame or *l*-frame in which the {x, y, z} coordinates are respectively east, north, and up. This transformation was coded as a new "Ranadu" function. This function takes as input a data.frame containing the attitude angles measured by the INS in variables named PITCH, ROLL, and THDG (true heading), all in units of degrees. It transforms an *a*-frame vector to an *l*-frame vector and returns the result. It would be preferable to do this in vectorized form, but such a representation hasn't yet been found, so the function uses a loop. As noted in the routine, the approach using R 'apply()' functions was slower than the loop, but that is because the specific approach tried here is likely unnecessarily convoluted. This is an area of needed improvement, but the loop as coded is practical and adequate for this study. The routine also accepts an "inverse" argument with default value FALSE; when true, the transformation is from the *l*-frame to the *a*-frame instead.

The transformation coded in this routine was obtained in standard ways using rotations ϕ about the roll axis, then θ about the pitch axis, and then ψ about the heading axis. The text lists sources for this transformation, but they differ from the matrix used here in that this transformation starts from what is called here the *a*-frame or aircraft reference frame with x forward, y starboard, and z downward and transforms to the local frame or *l*-frame with x east, y north, and z upward. The transformation matrix $\mathbf{R}_a^l$ is obtained as follows, where the first matrix changes the axis definitions:

$$\mathbf{R}_a^l = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}$$

$$R_a^l = \begin{bmatrix} \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\phi - \sin\psi\sin\theta\cos\phi \\ \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & -\cos\psi\sin\theta\cos\phi - \sin\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & -\cos\theta\cos\phi \end{bmatrix} \quad (1)$$

Two references for this transformation are listed in the manuscript. As checks, the transformations given in those two references (Eq. 2.6 in Lenschow and Spyers-Duran (1989) and Eq. 2-82 in Noureldin et al. (2013)) were further transformed to give directly the *a*-frame-to-*l*-frame transformation, with signs of rotation angles as required, and each led to (1).

It seemed useful to check that this transformation gives *l*-frame accelerations in reasonable agreement with those measured by GPS, so several checks were performed. One, reported in the manuscript, used a circle maneuver flown during DEEPWAVE research flight 15. The circle maneuvers are discussed in considerable detail in Sect. 7.1 of Cooper et al. (2016), where plots of the flight track in a coordinate frame drifting with the wind show that the Lagrangian tracks are close

approximations to circles. They were flown under control of the flight management system of the aircraft, which was able to maintain constant roll angle to a tolerance of a fraction of a degree. Two circles were flown in left turns, then two additional circles were flown in right turns. The ground-speed components measured by GPS were differentiated to obtain reference measurements of the corresponding acceleration components, again using the approach of replacing missing values by interpolation and then using fitted Savitzgy-Golay polynomials to find the derivatives. In this case, because the conditions changed too rapidly for the long extents used in the pitch-correction algorithm, the length of the polynomials was reduced to 21 1-Hz measurements for the horizontal components and 7 1-Hz measurements for the vertical component of acceleration. The results are shown in Fig. 1 of the technical note. The agreement through several maneuvers supports the validity of the transformation used to determing *l*-frame transformations from the measured accelerations in the *a*-frame.

Several other checks were performed also but have not been described in the manuscript. One important case was to consider a pitch maneuver in which the pitch was varied cyclically with about a 20-s period, with wings level but with airspeed and altitude varying. The maneuver flown on DEEPWAVE flight 15, 4:25:00–4:28:30 UTC, showed strong oscillations in the vertical acceleration, and again the body accelerations transformed to the *l*-frame matched will the accelerations deduced from the changes in vertical speed measured by the GPS receiver.

5 Comments on "The components of the derivatives"

The calculations are explained in the document, but it may be useful to elaborate on some of the choices that were made in regard to corrections for inertial effects. Such corrections enter during calculation of the accelerations and again during calculations of the rate of change in attitude angles.

5.1 Position derivative

The position state variables are latitude, longitude, and altitude. Their derivatives are related to the velocity state variable as follows (cf. Noureldin et al. (2013, pages 47–48 and 175)):

1. $d\lambda/dt = v_n/(R_m + z)$ where λ is the latitude, v_n is the northward ground speed of the aircraft, z the altitude, and R_m the radius of the Earth about a meridional circle, taken to be

$$R_m = R_e \frac{(1-\varepsilon^2)}{(1-\varepsilon^2 \sin^2 \lambda)^{3/2}} \tag{2}$$

   with the equatorial radius $R_e = 6378137\,\mathrm{m}$ and the Earth's eccentricity $\varepsilon = 0.08181919$.

   (a) $d\Psi/dt = v_e/((R_n + z)\cos\lambda)$ where Ψ is the longitude, v_e is the eastward ground speed of the aircraft, and R_n the normal radius of the Earth's ellipsoid, taken to be

$$R_n = R_e \frac{1}{(1-\varepsilon^2 \sin^2 \lambda)^{1/2}} \quad . \tag{3}$$

(b) $dz/dt = v_u$ where v_u is the vertical component of the aircraft motion. For the Kalman filter, the new variable "ROC" will be used for v_u; for the trial mechanization to duplicate the INS integration, the INS-provided variable VSPD was used.

5.2 Velocity derivative (acceleration)

There are two corrections needed if the accelerations measured in the a-frame are to be used in the l-frame. First, a correction must be made for the Coriolis acceleration that arises from motion relative to a spherical Earch. In addition, a correction is needed because the l-frame itself moves and remains oriented to match the local-level orientation. Evaluation of the typical magnitudes arising from the inertial terms indicates that the corrections are minor but not negligible, and their omission leads to significant accumulated errors during mechanization.

The Coriolis acceleration is

$$C_c = -2\Omega_e \times V \tag{4}$$

where Ω_e is the angular rotation rate of the Earch and V is the velocity of the aircraft relative to the Earth.

In the R code, this equation and others involving vector cross products are represented by skew-symmetric matrices, so this choice merits some explanation. Strangely, core R mathematical functions do not include the cross product needed by (4). Instead, in R, the "cross product" has a different meaning. Of course, there are many solutions that contributors have produced that could be used, but it seemed awkward to use a non-standard package for such a basic function and it seemed appropriate to select a solution that ensured reproducibility in the future. The representation of the cross product via a skew-symmetric matrix provides such a solution, but it is likely to be unfamiliar to many readers so an elaboration is provided here: If an angular-rotation vector has coordinates $\omega = (\omega_1, \omega_2, \omega_3)$ then the skew-symmetric representation is

$$S = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{5}$$

Then the cross produce $\omega \times \mathbf{V}$ can be found by matrix multiplication via SV with V the column matrix representing the velocity. This equality can be verified by comparing the result of the matrix multiplication to the standard formula for the vector product. In particular, the vector representing the rotation of the Earth (ω_{ie}^l, where the notation denotes the rotation rate of the Earth relative to an inertial frame, as observed in the l-frame) has components $(0,\ \omega_e \cos\lambda,\ \omega_e \sin\lambda)$ where ω_e is the rotation rate and λ is the latitude, so the Coriolis acceleration can be represented by

$$C_c = -2\omega_e \begin{pmatrix} 0 & -\sin\lambda & \cos\lambda \\ \sin\lambda & 0 & 0 \\ -\cos\lambda & 0 & 0 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} \tag{6}$$

This is the form chosen to represent the Coriolis acceleration in the derivative function.

Noureldin et al. (2013), p. 179, give the components of the rotation rate of the l-frame relative to the Earth, as expressed in the l-frame, as $\omega^l_{el} = (-V_2/(R_m+h), V_1/(R_n+h), V_1 \tan\lambda/(R_n+h)$ where R_m and R_n are respectively the meridional and normal radii of the earth, and h is the altitude above the Earth. Expressed as a skew-symmetric matrix, this is the other component used to correct the measured accelerations when transforming from the a-frame to the l-frame. The resulting equation for the required correction can be expressed as follows:

$$\blacksquare\dot{\mathbf{v}} = -(2\Omega^l_{ie} + \Omega^l_{el})\mathbf{v}^{(l)} \tag{7}$$

where the rotation matrices, respectively representing the Earth's rotation and the l-frame rotation, are

$$\Omega^l_{ie} = \begin{bmatrix} 0 & -\omega^e \sin\lambda & \omega^e \cos\lambda \\ \omega^e \sin\lambda & 0 & 0 \\ -\omega^e \cos\lambda & 0 & 0 \end{bmatrix} \tag{8}$$

$$\Omega^l_{el} = \begin{bmatrix} 0 & \frac{-v_e \tan\lambda}{R_N+z} & \frac{v_e}{R_N+z} \\ \frac{v_e \tan\lambda}{R_N+z} & 0 & \frac{v_n}{R_M+z} \\ \frac{-v_e}{R_N+z} & \frac{-v_n}{R_M+z} & 0 \end{bmatrix} \tag{9}$$

with $\omega^e = 7.292 \times 10^{-5}$ the angular rate of rotation of the Earth.

Without some feedback, the vertical component is unstable, so the INS uses feedback based on the pressure altitude to control the third component of aircraft velocity. To obtain an analogous variable for comparison, a similar feedback loop was used for the third component arising from this new mechanization. A conventional feedback loop uses three coefficients specified in terms of a parameter ζ as $\{C_0, C_1, C_2) = \{3\zeta, 4\zeta^2, 2\zeta^3)$. In an attempt to duplicate the INS reference source, the reference altitude was calculated as the pressure altitude corresponding to the avionics-supplied value of the pressure, but this still gave a small offset from the INS values so it appears that a different pressure source must be used by the INS. The loop was implemented in the conventional way according to the algorithm in the next box:

```
initialize:
   wp3F=0; hxF=0; hxxF=0; hi3F=PALT[1];
for each ith time step of time change dt:
   wp3F += (sv[6] - C1*hxF - C2*hxxF) * dt; ## sv is the state vector; sv[6] is vertical speed
   hi3F += (wp3F - C0*hxF) * dt;
   hxF = HI3F-PALT[i];
   hxxF += hxF*dt;
   sv[6] = (sv[6] + wp3F)/2;
   sv[3] = hi3F; ## sv[3] is then the altitude of the aircraft
   [save sv[3] and sv[6] as the values of the aircraft-state vector vs time]
```

5.3 Attitude angles

The approach to finding the derivatives of the attitude angles is to find the derivative ($\dot{R}^l_a$) of the transformation matrix from the a-frame to the l-frame and then find the attitude-angle derivatives from the definition of the transformation matrix.[1] The steps are these:

1. The IRU measures the rotation rate ω^a_{ia} of the a-frame relative to an inertial frame, as observed in the a-frame. In the a-frame, the roll axis is the x-axis forward along the longitudinal axis of the aircraft, the pitch axis is the y-axis to starboard, and the yaw axis is the downward z-axis but with a sign reversal arising from the inverse relationship between yaw angle and heading angle, so $\omega^a_{ia} = (\dot{\theta}, \dot{\phi}, \dot{\Psi}) = (\text{BROLLR}, \text{BPITCHR}, -\text{BYAWR})$ where the latter are the variables as recorded in the data file. As a skew-symmetric matrix such as discussed above, this rotation rate is represented as

$$\Omega^a_{ia} = \begin{bmatrix} 0 & -\dot{\Psi} & -\dot{\theta} \\ \dot{\Psi} & 0 & \dot{\phi} \\ \dot{\theta} & -\dot{\phi} & 0 \end{bmatrix} \tag{10}$$

2. The rotation rate needed to find the derivative of the transformation matrix ($\dot{R}^l_a$) is instead the rotation rate $\omega^a_{la}$of the a-frame relative to the l-frame, as observed in the a-frame. This differs from ω^a_{ia} by the rotation rate of the l-frame relative to the a-frame, as observed in an inertial frame (ω^a_{il}): $\omega^a_{la} = \omega^a_{ia} - \omega^a_{il}$. Transforming ω^a_{la} from the a-frame to the l-frame then gives the desired derivative $\dot{R}^l_a$.

3. As for velocity, the term ω^a_{il} contains two contributions, one from the rotation of the Earth (ω^a_{ie}) and the second from the rotation of the l-frame relative to an inertial frame (ω^a_{el}). These can be obtained by transforming ω^l_{ie} and ω^l_{el} as defined in the previous subsection to the a-frame. The appropriate transformation is the inverse of R^l_a, denoted $R^a_l = t(R^l_a)$ where "$t()$" denotes the matrix transpose operator: $\omega^a_{ie} = R^a_l \omega^l_{ie}$ and $\omega^a_{el} = R^a_l \omega^l_{el}$. In the code, this is replaced by the skew-symmetric equivalent: If Ω^l is the skew-symmetric representation

[1] This material relies heavily on the development in Noureldin et al. (2013), pp. 178–180.

of $\omega_{ie}^l + \omega_{el}^l$, with skew-symmetric representations given by (8) and (9) above. The skew-symmetric representation in the a-frame, following the model for transformation of skew-symmetric matrices, is then $\Omega^a = R_l^a \Omega^l R_a^l$.

4. Expressing ω_{ia}^a also as a skew-symmetric matrix, following (5), and subtracting Ω^a from it, gives a skew-symmetric matrix that, when multiplied by R_a^l, gives the desired derivative $\dot{R}_a^l = \mathbf{R}_a^l(\Omega_{ia}^a - \Omega^a)$. (See Noureldin et al. (2013), p. 180.)

5. The attitude angles can be found from the definitions of the transformation matrix, as specified by (1):

$$\begin{aligned} \theta &= \arcsin(-R_l^a[3,1]) \\ \phi &= \arctan 2(R_l^a[3,2], -R_l^a[3,3]) \\ \Psi &= \arctan 2(R_l^a[1,1], R_l^a[2,1]) \end{aligned} \tag{11}$$

The derivatives of the attitude angles are then given by the derivatives of the arcsin and arctan functions:

$$\begin{aligned} \frac{d(\arcsin(x))}{dx} &= \frac{1}{\sqrt{1-x^2}} \\ \frac{d(\arctan(x))}{dx} &= \frac{1}{1+x^2} \\ \dot{\theta} &= \frac{-1}{\sqrt{1-R_l^a[3,1]^2}} \dot{R}_l^a[3,1] \qquad (12) \\ \dot{\phi} &= \frac{1}{1+\left(R_l^a[3,2]/R_l^a[3.3]\right)^2} \frac{R_l^a[3,2]}{R_l^a[3,3]} \left(\frac{\dot{R}_l^a[3,3]}{R_l^a[3,3]} - \frac{\dot{R}_l^a[3,2]}{R_l^a[3,2]} \right) \qquad (13) \\ \dot{\psi} &= \frac{1}{1+\left(R_l^a[1,1]/R_l^a[2,1]\right)^2} \frac{R_l^a[1,1]}{R_l^a[2,1]} \left(\frac{\dot{R}_l^a[1,1]}{R_l^a[1,1]} - \frac{\dot{R}_l^a[2,1]}{R_l^a[2,1]} \right) \\ &= \frac{1}{1+\left(R_l^a[1,1]/R_l^a[2,1]\right)^2} \left(\frac{\dot{R}_l^a[1,1]}{R_l^a[2,1]} - \frac{\dot{R}_l^a[2,1] R_l^a[1,1]}{R_l^a[2,1]^2} \right) \qquad (14) \end{aligned}$$

Because $R_l^1[2,1]$ (equal to $\cos\psi\cos\theta$) can pass through zero for heading of 90 or 270°, some numerical problems arise when the last equation is used. (Avoiding similar problems at the zero for $R_l^a[1,1]$ is the justification for using the second form of the last equation.) For small $R_l^a[2,1]$, the last equation becomes

$$\dot{\psi} \approx \left(\frac{R_l^a[2,1]\dot{R}_l^a[1,1]}{R_l^a[1,1]^2} - \frac{\dot{R}_l^a[2,1]}{R_l^a[1,1]} \right) \tag{15}$$

which avoids the singularity, so this form is used when the absolute value of $R_l^a[2,1]$ is smaller than $R_l^a[1,1]/10$.

5.4 The function "STMFV()"

To standardize calculation of the derivatives of the state vector for both mechanization and the Kalman filter, the code was incorporated into a function `STMFV(D, .components, .aaframe)`. The first argument is either a data.frame, vector, or matrix containing the nine components of the state vector and the six a-frame measurements from the IRU. If multiple rows are included, all are processed and a matrix of derivatives is returned that has the original number of rows and `.components` components; this is included to accommodate either the mechanization section that uses a nine-component state vector or the Kalman-filter section that uses a 15-component error-state vector. The argument `.aaframe` defaults to 'a' and attitude angles are assumed to be in the a-frame. The other option 'l' is not used; this was a provision for calculating all derivatives in terms of a state vector with all components in the l-frame, which was used during testing but not in the final program. As coded, STMFV () assumes that the measurements from the IRU (accelerations and rotation rates) have been added to the aircraft-state vector for convenience. This function is called for each time step during mechanization, and it then serves as the function argument to the Jacobian calculation that finds the error-state transition matrix used by the Kalman filter. The code is in the R chunk called "chunks/STMFV.R" to make it available to the processing program for the Technical Note and also for subsequent use in the data processing script "KalmanFilter.R" that applies the Kalman filter to other data files.

5.5 Discussion of the "Tests of the derivatives" section

Time shifts

An important part of the work that is background to this section was determining the time shifts to apply to various measurements from the INS. The "initialization" chunk of R code applies these time shifts, using the "Ranadu::ShiftInTime()" function. The time shifts quoted in the text were determined by varying the shifts specified in that section and observing the result by plotting the plot generated in the "plot-acc" chunk but modified to display only the pitch maneuver (3::16:00 to 3:18:00) and observing the difference between the measured values and the values determined from the derivatives of the velocity measurements. This was not done systematically but rather by trial-and-error to find a combination of time shifts that showed small fluctuations during the maneuver. These results are sensitive to pitch and roll as well because, in the case of pitch, gravity is resolved into a longitudinal component of acceleration that affects the comparison to BLONGA, and in the case of roll small roll angles created similar contributions to the lateral acceleration represented by BLATA. In the exploration of time shifts, the measured accelerations were assumed to have the same time delays, and similar assumptions were made about the pitch-roll pair and the set of velocities. With the listed shifts, the residual errors (shown in Tables 1 and 2 of the Technical Note) were so small that further refinement did not seem warranted.

Tables 1 and 2

These tables show results from regression fits obtained using the R function "lm()". Fit results including the coefficients, the standard deviation of residuals from the fit, and the square of the correlation coefficient are entered directly into the tables from the results of the fits, via the "\Sexpr()" function. The values included in the tables are calculated in the R-code chunk named "check-accelerations". The details for obtaining the entries in Table 2, which are used as calibrations of the accelerometers, are as follows:

1. Differentiate the GPS-provided velocity components (GGVEW, GGVNS, GGVSPD) via the R routine signal::sgolayfilt() with appropriate argument that returns the first derivative of the filtered function. Cubic Savitzky-Golay polynomials spanning 11 25-Hz measurements were used, which effectively imposes a high-pass cutoff of about 8 Hz and so applies only minor smoothing while retaining high temporal resolution in the derivatives.

2. Apply an appropriate rotation correction as discussed above, but with reversed sign, to these accelerations.

3. Transform the resulting accelerations to the *a*-frame using the same function XformLA() discussed above but with the argument ".inverse" set TRUE to give the transformation from the *l*-frame to the *a*-frame instead.

4. Use the R routine "lm()" to find the linear-model fit relating the IRU-measured accelerations to the GPS-derived accelerations. The coefficients from that fit are those quoted in the document.

5. One subtle circularity needs mention here: The coefficients obtained are obtained from the original measurements as stored in the data.frame Data, but the resulting calibration was already applied earlier and saved in the data.frame SP that is used for the mechanization loop. These calibrations were inserted manually into the statements near the end of R-code chunk "INS-data", in the section following the comment "adjustments". As mentioned in the main report, no calibration was applied to the measurements "BLATA" because the IRU-provided and GPS-derived lateral accelerations were consistently small and the calibration did not appear sufficiently reliable to trust, especially because the result was significantly different from the identity calibration that reasonably characterized the other two components of measured acceleration. check this

Generation of Figure 1

Many of the figures in the Technical Note were generated following the same model, so that model is described here. The plot is generated by a call to the Ranadu function `Ranadu::ggplotWAC()`, which uses the faceting capability of the R package ggplot2 to construct multiple plots with the same time scale. The code is in R chunk "plot-acc".

Many of the plots were generated in multiple-panel displays. This was straightforward in standard R graphics, using the "layout()" and "par()" functions, so the standard R plot function was used in first drafts. However, plotting with ggplot2 produced plots with better appearance. The construction of multiple-panel figures was tried in two ways, first using viewports and then using facets. With viewports, it was difficult to get the panels to align vertically without much tailoring of margins, but facets handled this automatically so that was the method used in the final version. The procedure is worth documenting here because it took some exploration. This was built into a revision of the function "Ranadu::ggplotWAC()," which is used as follows:

1. Construct a new data.frame that contains only the variables that will appear on the plot, along with the "Time" variable (which should be first). Variables should be in lines/panel format, i.e., all variables for the top panel, all for the next-to-top, etc.
2. Call ggplotWAC() with these new arguments:
   (a) panels: the number of panels or facets, aligned vertically.
   (b) labelL: the names to appear in the legend for the lines. Only one legend appears for all panels. Example: c("IRU", "GPS").
   (c) labelP: the names to appear at the right of each panel. Example: c("longitudinal", "lateral", "upward").
   (d) theme.version: 1 to use a theme with minor adaptations for the faceted plots.
3. The procedure that ggplotWAC() uses is as follows:
   (a) Define two groups, VarGroup and PanelGroup, to describe the structure specified by the new arguments.
   (b) "melt" the data.frame to a long-format data.frame, with "Time" as the id.var, and include the two new groups in the data.frame definition.
   (c) Use the ggplotWAC() arguments for cols, lwd, and lty in groups for each panel, with replication as needed. Each panel must be the same.
   (d) Construct the initial plot definition using ggplot, using x=Time, y=value, colour=VarGroup, linetype=VarGroup in the aesthetic specification.
   (e) Add "geom_line(aes(size=VarGroup))" to this definition to plot the lines.
   (f) Use appropriate manual scale definitions to set the desired colors, line types, and line widths in order to get consistent plot lines and legend.
   (g) Add "facet_grid()" with first argument "PanelGroup" as included in the data.frame definition.
   (h) Tailor desired aspects of the plotting theme.

The routine then returns a plot definition, and that definition can be tailored further using "theme()" elements if desired or printed without further modification.

An example is this code that generates the plot of attitude angles (generating Fig. 2 that appears in Sect. 2.4):

```
d <- with(Data[setRange(Data, 32000, 35500), ],
          data.frame(Time, PITCH, PITCHX, 10*DPITCH, ROLL, ROLLX, 10*DROLL,
                     THDG, THDGX, 10*DTHDG))
ggplotWAC (d, col=c('blue', 'red', 'forestgreen'),
          ylab=expression (paste ('attitude angles [', degree, ']')),
          lwd=c(1.4, 0.8, 1), lty=c(1, 42, 1), panels=3,
          labelL=c('original', 'new', '10*diff'),
          labelP=c('pitch', 'roll', 'heading'),
          legend.position=c(0.8, 0.97), theme.version=1)
```

5.6 Discussion of the "Mechanization" section

The algorithm

The following box contains an algorithmic flow chart describing the mechanization scheme, which is also diagrammed in Fig. 2:

```
Initialize a time-series matrix of aircraft-state vectors;

    ## Each row: three position coordinates,
    ##           three velocity coordinates,
    ##           three attitude-angle coordinates

Add the measured accelerations, rotations (M) to the same matrix;
Define (STMFV(sv,M)) that returns the matrix of derivatives D;

    ## each row D_i is the derivative of sv_i
    ##            with respect to time, given M_i

For each time (index i) in the data set, at intervals Δt:

    Propagate the state vector forward one time step,
              via sv[i+1]←sv[i]+D[i]Δt;
    Save the resulting new state vector sv[i+1]
```

The core of the mechanization is the function STMFV() discussed in Sect. 5.3 above. Various references have provided "cookbook" equations for the required derivatives. In particular, the approach followed was that described by Noureldin et al. (2013), but with some differences arising primarily from use of the aircraft-coordinate frame or *a*-frame instead of the *b*-frame (body frame) used in that source. In addition, the derivatives of the attitude-angle components were calculated differently, so some detail regarding this function is appropriate.

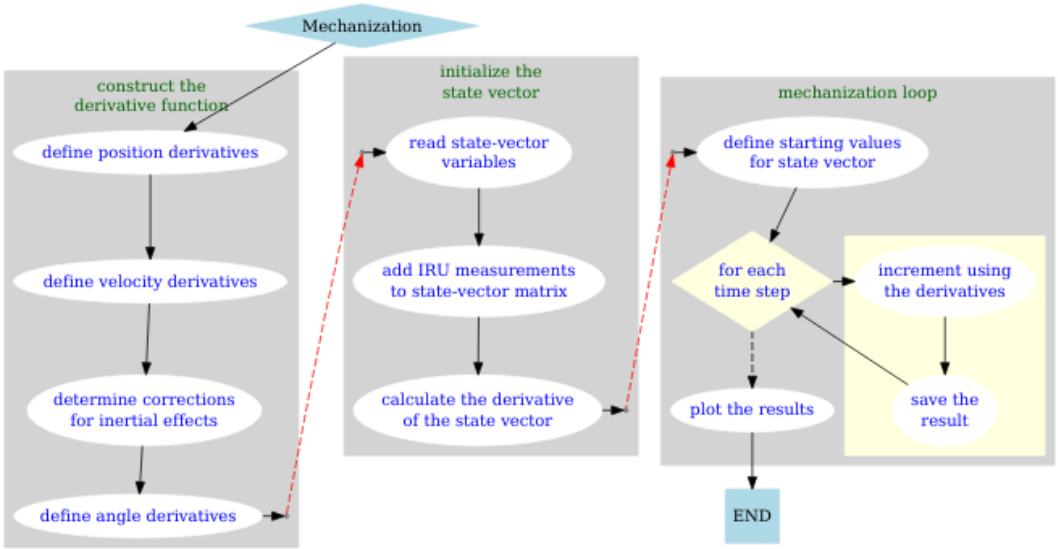


Figure 2: Workflow diagram for the mechanization test in the technical note. Some items (e.g., "run the main Kalman-filter loop") are described in additional workflow diagrams below.

Time shifts

Before the step-wise integration, some adjustments were made to the measurements. To ensure against missing values, interpolation was used to fill in a very small number of missing values in the measurements from the IRU. In addition, some of the measurements were shifted in time to compensate for delays in recording of the variables. This shifting is a part of normal processing, but the mechanization is very sensitive to timing errors so some fine tuning was used. The procedure was discussed in connection with the Ranadu::ShiftInTime() function discussed above. In addition, a few errors in interpolation were introduced to the heading variable by this process, so a special loop searched for these errors (four in the data set used) and corrected them. The problem arose from a deficiency in the ShiftInTime() function that still needs to be corrected.

The integration

After initializing the aircraft-state vector to match that from the INS at a point near the start of the flight, the mechanization proceeded by taking time steps where the changes in the aircraft-state vector were determined by the preceding derivative function. The Technical Note describes the initialization and the sequence of time steps. For convenience, the measurements of acceleration and rotation rate from the IRU were added to the state vector with each time step so that those measurements were transferred to the function `STMFV()` when it was called. The integration was done either by simple Euler integration or via a fourth order Runge-Kutta scheme, with indistinguishable results. A special test and correction were used to ensure that the heading remained in the range from 0 to 2π.

The results

Like Fig. 1, Figs. 2 and 3 showing results of the mechanization were also generated using the "Ranadu::ggplotWAC()" convenience function, which calls functions that are part of ggplot2 with some assigned settings. Details are presented above (on page 12). All three of these figures were generated during the same processing step that produced the text document, and they were incorporated into that document using the R package "knitr" to place the figures appropriately in the resulting LaTeX file. During the mechanization, the aircraft-state vector was saved in variables in an R data.frame called SP, and at the end these variables were added back to the original data.frame (called Data) as new variables with names like LATX, LONX, ALTX, ... that correspond to the original INS-produced variables LAT, LON, ALT. These variables were then used in plot commands to generate the figures in the subsections that compare the results for attitude angles, velocity components, and position.

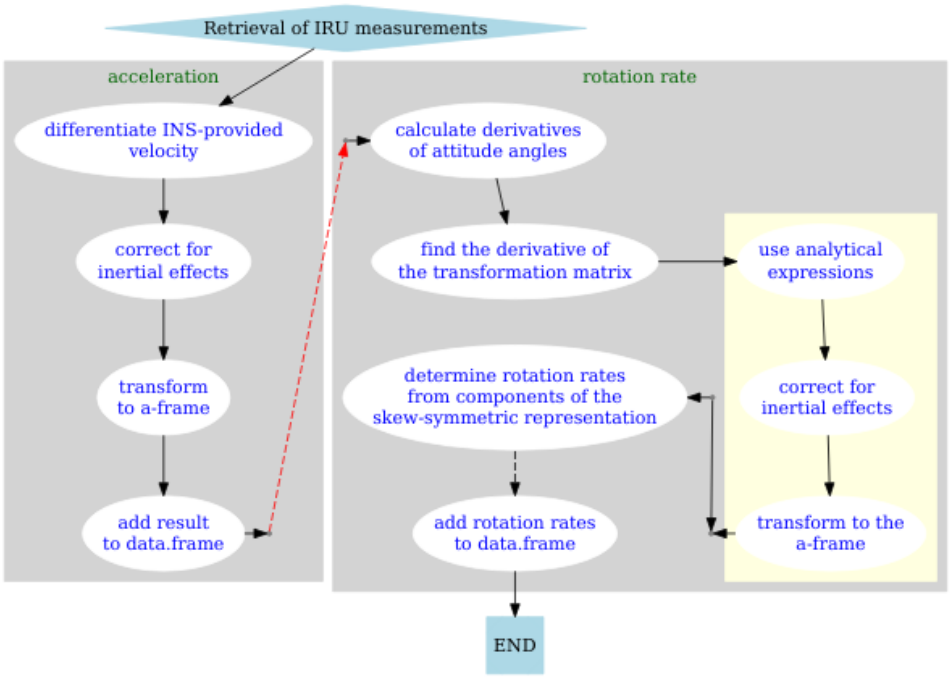


Figure 3: Workflow diagram for the retrieval of the measurements originally provided by the IRU. If these are missing from the data file being used, they are added by this procedure.

6 Elaboration on the ancillary functions

6.1 Rate of climb

This is discussed fully in the Techical Note and the memo referenced there.

6.2 Retrieving IRU measurements

A workflow diagram for this section is presented as Fig. 3. The two parts shown in this figure are independent and are discussed separately below. The yellow box is an expanded description of the box leading into it, to show how that derivative is found. The R implementation of the code in the technical note ("KalmanFilter.R") include a branch that constructs these retrieved values when the IRU-provided measurements (BLATA, BLONGA, BNORMA, BPITCHR, BROLLR, BYAWR) are not present in the original data file.

Rotation rates

Item 1 in this subsection of the Technical Note prescribes starting with analytical expressions for the derivative of the transformation matrix. Those analytical expressions are developed in this

Workflow Document, as follows:

The rotation rates and accelerations measured by the IRU {BPITCHR, BROLLR, BYAWR, BLATA, BLONGA, BNORMA} are the rotation rates and accelerations of the a-frame relative to an inertial frame. Without Coriolis corrections, retrieving the original measurements would be straightforward: Differentiate the measured angles and components of the aircraft velocity and transform the results to the a-frame. However, the inertial contributions from motion and rotation of the a-frame, while minor, are not negligible. The needed corrections are developed in this section, to complement the outlined solutions presented in the technical note.

The derivatives of the attitude angles were expressed in terms of the transformation matrix $\mathbf{R}_a^l$ and its time derivative by (12)–(14). Here the reverse procedure, finding the derivative of the transformation matrix from the derivatives of the attitude angles, is needed. Equation (1) can be differentiated to express that derivative, which will have these components, where dots over symbols indicate time derivatives:

$$
\begin{aligned}
&[1,1]\text{: dR[1,1]dt = d}(\sin\psi\cos\theta)/\text{dt} \rightarrow \\
&\qquad \dot{\psi}\cos\psi\cos\theta - \dot{\theta}\sin\psi\sin\theta \\
&[1,2]\text{: } d(\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi)/dt \rightarrow \\
&\qquad \dot{\psi}(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi) + \dot{\theta}\sin\psi\cos\theta\sin\phi \\
&\qquad\quad +\dot{\phi}(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \\
&[1,3]\text{: } d(\cos\psi\sin\phi - \sin\psi\sin\theta\cos\phi)/dt \rightarrow \\
&\qquad \dot{\psi}(-\sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi) + \dot{\theta}(-\sin\psi\cos\theta\cos\phi) \\
&\qquad\quad +\dot{\phi}(\cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi) \\
&[2,1]\text{: } d(\cos\psi\cos\theta)/dt \rightarrow \\
&\qquad \dot{\psi}(-\sin\psi\cos\theta) + \dot{\theta}(-\cos\psi\sin\theta) \\
&[2.2]\text{: } d(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi)/dt \rightarrow \\
&\qquad \dot{\psi}(-\sin\psi\sin\theta\sin\phi - \cos\psi\cos\phi) + \dot{\theta}\cos\psi\cos\theta\sin\phi \\
&\qquad\quad +\dot{\phi}(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \\
&[2.3]\text{: } d(-\cos\psi\sin\theta\cos\phi - \sin\psi\sin\phi)/dt \rightarrow \\
&\qquad \dot{\psi}(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) + \dot{\theta}(-\cos\psi\cos\theta\cos\phi) \\
&\qquad\quad +\dot{\phi}(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi) \\
&[3,1]\text{: } d(-\sin\theta)/dt \rightarrow \\
&\qquad \dot{\theta}(-\cos\theta) \\
&[3,2]\text{: } d(\cos\theta\sin\phi)/dt \rightarrow \\
&\qquad \dot{\theta}(-\sin\theta\sin\phi) + \dot{\phi}\cos\theta\cos\phi \\
&[3,3]\text{: } d(-\cos\theta\cos\phi)/dt \rightarrow \\
&\qquad \dot{\theta}\sin\theta\cos\phi + \dot{\phi}\cos\theta\sin\phi
\end{aligned}
$$

where arrows indicate that the differentiation of the left side leads to the right side of the equations. The remainder of the retrieval of rotation rate is described in the Technical Note, preceding and via (6)–(8) in that document.

From the skew-symmetric representation of the rotation rates as measured in the a-frame, the individual components of the rotation-rate vector can be extracted. For example, the [3,1] component of the result is −BPITCHR.

Accelerations

The accelerations in the a-frame are retrieved as described in the Technical Note, but some additional detail is provided here. The detailed steps used are these:

1. Construct a vector representing the aircraft motion from the variables {VEW, VNS, VSPD} as produced by the INS.

2. Interpolate where there are missing values to avoid failure of the filter function in the next step. The routine used was that provided by the "na.approx ()" function in the "zoo" package for R.

3. Differentiate the time series, either using the R function "diff ()" or, to introduce some smoothing, Savitzgy-Golay polynomials. The "sgolayfilt ()" function in the "signal" package for R was used, and the derivative was determined over 11 1-Hz samples. With an appropriate argument, this function returns the derivative when called with the time series as an argument.

4. For the third (vertical) component, change the sign and subtract the acceleration of gravity.

5. Add the rotation correction (usually subtracted during translation from the a-frame to the l-frame) as calculated by the function "RotationCorrection ()". This function determines the same correction used in the derivative function "STMFV()" but is coded independently. (That duplication could be removed to avoid later introduction of an inconsistency between the two functions.) The correction can be explained by discussing the following code for that function. Argument ".data" is an R data.frame containing the latitude and altitude in variables .data$LAT and .data$GGALT. The vector ".V" is the l-frame vector representing aircraft velocity. The function treats the input arguments as arrays, so it will provide corrections for the entire flight on one call if provided a full-flight data.frame and velocity matrix. When used to correct just one vector, the dimensions of ".V" must be changed to 1x3 for consistency with the vectorized structure of the function.

```
RotationCorrection <- function (.data, .V) {
  omegaE <- StandardConstant ('Omega') ## Earth's angular velocity
  C <- vector ('numeric', 3*(DL <- nrow(.data))); dim(C) <- c(DL,3)
  if (DL == 1) {dim (.V) <- c(1,3)}    ## consistency w/ vectorization
  lat <- .data$LAT * pi / 180          ## convert to radians
  sinLat <- sin(lat); cosLat <- cos(lat); tanLat <- tan(lat)
  Ree <- 6378137; Ecc <- 0.08181919    ## constants for Earth radii
  Rn <- Ree / (1 - (Ecc * sinLat)^2)^0.5
  Rm <- Rn * (1 - Ecc^2) / (1 - (Ecc * sinLat)^2) + .data$GGALT
  Rn <- Rn + .data$GGALT
  omega <- as.vector (-.V[,2] / Rm, 2 * omegaE * cosLat + .V[,1] / Rn,
                      2 * omegaE * sinLat + .V[,1] * tanLat / Rn)
  dim(omega) <- c(DL, 3)
  zro <- rep(0, DL)
  ## skew-symmetric representation:
  M <- array (c(zro, -omega[3], omega[2], omega[3], zro, -omega[1],
                -omega[2], omega[1], zro), dim=c(DL, 3, 3))
  MP <- aperm(M)
  ## R uses column-major representation of matrices.
  ## Need matrix multiplication, each row:
  for (i in 1:DL) {C[i,] <- MP[,,i] %*% .V[i,]}
  return (C)
}
```

6. Apply the transformation matrix that transforms from the l-frame to the a-frame. This is the transpose of (1).

7. Subtract the acceleration of gravity from the third component.

6.3 The "simpler" functions

Pitch correction

The technical note describes this in detail, and complementary descriptions are available as referenced there (with some plots of the effect of the correction) and also in the workflow document for the previous technical note, available at this link. Some additional detail regarding the function (Ranadu::CorrectPitch()) that calculates the correction to pitch is included here.

A workflow diagram for the function is shown in Fig. 4. The steps in the algorithm are these:

1. Prepare the data.frame: ["preprocessing" box in the diagram]

   (a) Test to see if all required variables are present in the input data.frame; return 0 otherwise.

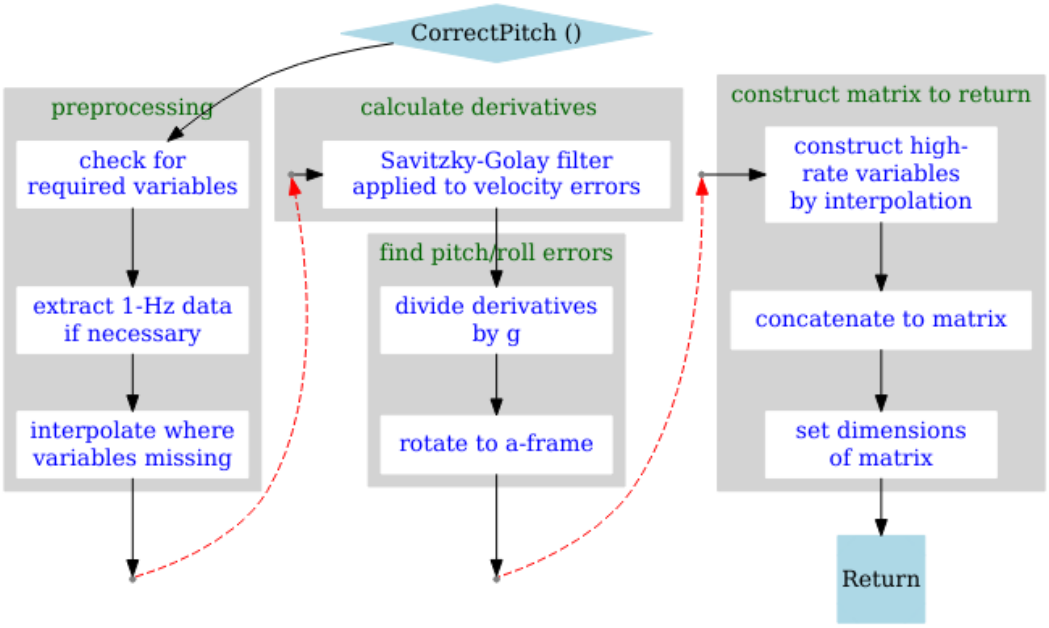


Figure 4: Workflow diagram for the Ranadu function CorrectPitch(), which returns estimates of errors in measurements of pitch and roll. The returned values, with units of degrees, should be subtracted from the measurements to obtain corrected values.

(b) Determine the data rate of the input data.frame from the time difference between samples. If it is higher than 1 Hz, extract a working 1-Hz data frame to use for the calculations. This greatly execution time for high-rate files.

(c) Allow for either LAT or LATC to provide the latitude required for the gravity routine.

(d) Interpolate to fill in missing values for VNS, VEW, GGVNS, GGVEW, using the zoo::na.approx() function. If there are remaining missing-value elements in the time series (which can occur if the gaps exceed 1000 s), set the values to zero for those regions to avoid failure at the next step. The variable MaxGap in the function determines the maximum gap for interpolation; it can be changed only by changing the code.

2. Calculate the derivatives: ["calculate derivatives" box in the diagram]

   Determine the derivatives of the error terms (VNS-GGVNS) and (VEW-GGVEW) using third-order Savitzky-Golay filters, specifically the R function signal::sgolayfilt(). Use a span of 1013 points (changeable via an argument to CorrectPitch()) and calculate the first derivative by supplying an appropriate argument (m=1) to the filter routine.

3. Find the errors in pitch and roll: ["find pitch/roll errors" box in the diagram]

   (a) Calculate the acceleration of gravity as a function of latitude and altitude using the Ranadu function Gravity() and divide the derivatives (with appropriate sign) by this value to get the *l*-frame errors in pitch and roll.

   (b) Transform the result to the *a*-frame using manuscript equation (27). (This should eventually be changed to (16) in this workflow document to get better results in and near turns.)

4. Construct the matrix to return, which has rows matching the input data.frame and two columns, the first representing the pitch error and the second the roll error. ["construct matrix to return" box in the diagram]

   (a) If the data rate of the original data.frame is higher than 1 Hz, interpolate the working 1-Hz data.frame to the original rate, again using zoo::na.approx() for linear interpolation.

   (b) Concatenate the two vectors representing the pitch and roll errors to one vector.

   (c) Convert to an appropriate matrix with dimensions [DL,2] where DL is the length of the original data.frame. This conversion uses "dim()" with the concatenated vector.

5. Return a two-dimensional array with the two components being the *a*-frame errors in pitch and roll, so that the result can be used to obtain corrected values of the pitch and roll via

```
PITCHC <- D$PITCH - CorrectPitch (D)[,1]
ROLLC  <- D$ROLL - CorrectPitch (D)[,2]
```

All steps in this function are vectorized, to operate on the entire time series in vectorized function calls. Even for high-rate data.frames, the processing involves little delay: On a modest linux desktop system, processing a 7-h 25-Hz file takes about 2 s.

The uncertainty in the pitch correction expected from this function is not discussed in the technical note, so some comments regarding uncertainty are included here. Cooper et al. (2016) estimated that the uncertainty in the pitch correction arising from the uncertainty in determining the derivatives is smaller than 0.0001°. Some elaboration leading to that estimate is presented here. Modern GPS receivers, especially if augmented by special signals or special processing, produce 1-Hz measurements with uncertainty of around $0.03\,\mathrm{m\,s^{-1}}$(Cooper et al. (2016)). The consistency of the Schuler oscillation, as illustrated by Figs. 6 and 7 of the technical note, suggests that derivatives in velocity can be determined by averaging over periods of at least 10 min or more, so if the ground-speed measurements from the INS have uncertainty of about $0.03\,\mathrm{m\,s^{-1}}$ (where variance spectra for the 1-Hz measurements begin to show noise), the uncertainty in differences between these two signals might be expected to be $0.03\sqrt{2} \approx 0.04\,\mathrm{m\,s^{-1}}$, and averaged over 10 min or perhaps 60 autocorrelation times the resulting difference could be resolved to $0.04/\sqrt{60} \approx 0.005\,\mathrm{m\,s^{-1}}$. Over intervals separated by 10 min, the derivative then might be determined with an uncertainty of $0.005 \times 1/600 \approx 10^{-5}\,\mathrm{m\,s^{-2}}$, leading to an uncertainty in the pitch correction from technical note Eqn. (10) of about 10^{-6} or 0.00005°. Third-order Savitzky-Golay polynomials of order m reduce noise in an average by a factor of $\sqrt{(3(3m^2-7)/4m(m^2-4)} \simeq 0.05$ for m=1013 or 0.06 for m=601. Then for $0.04 * 0.06 = 0.0024$m/s, 600-s separation gives acceleration uncertainty of about $0.0024/600 = 4e-6$ or pitch uncertainty of 2e-5°. Therefore this contribution to uncertainty in the correction has been neglected. Differences between the estimate determined from the "CorrectPitch()" function and from the Kalman filter, shown in Fig. 9 of the technical note, are mostly smaller than 0.01°, which suggests about this level of confidence in the results. Much of the discrepancy in that figure arises from measurements in turns, where the "CorrectPitch()" function applies an approximate correction for the angle transformation from the *l*-frame to the *a*-frame that has some error in turns. That function could be improved by the exact transformation.

Heading correction

The technical note itself contains a complete description of this algorithm, so no additional information is included here except for the following flow chart (Fig. 5) that indicates the structure of the associated function Ranadu::CorrectHeading().

6.4 Angle of attack

The discussion in the Technical Note, supplemented by the memo referenced there, should be adequate for this section.

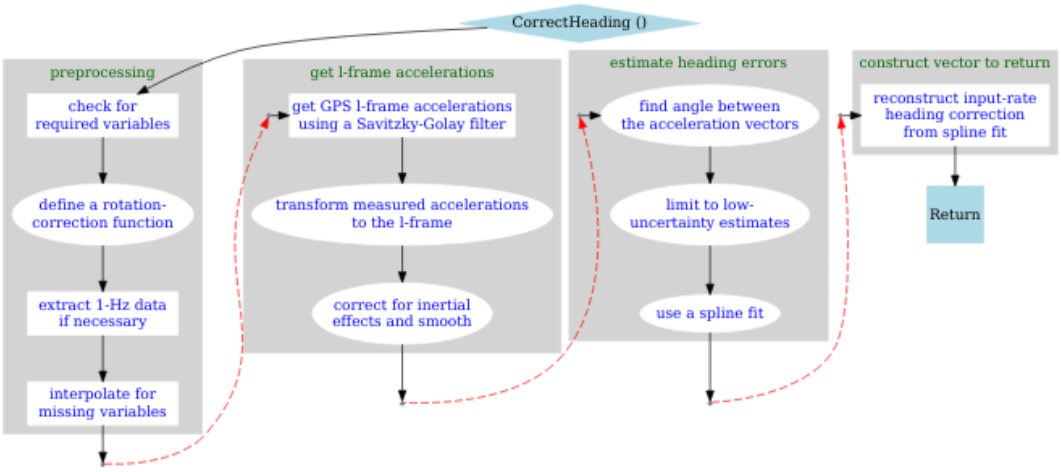


Figure 5: Workflow diagram for the Ranadu function CorrectHeading(), which returns estimates of errors in the measurement of heading. The returned values, with units of degrees, should be subtracted from the measurements to obtain corrected values.

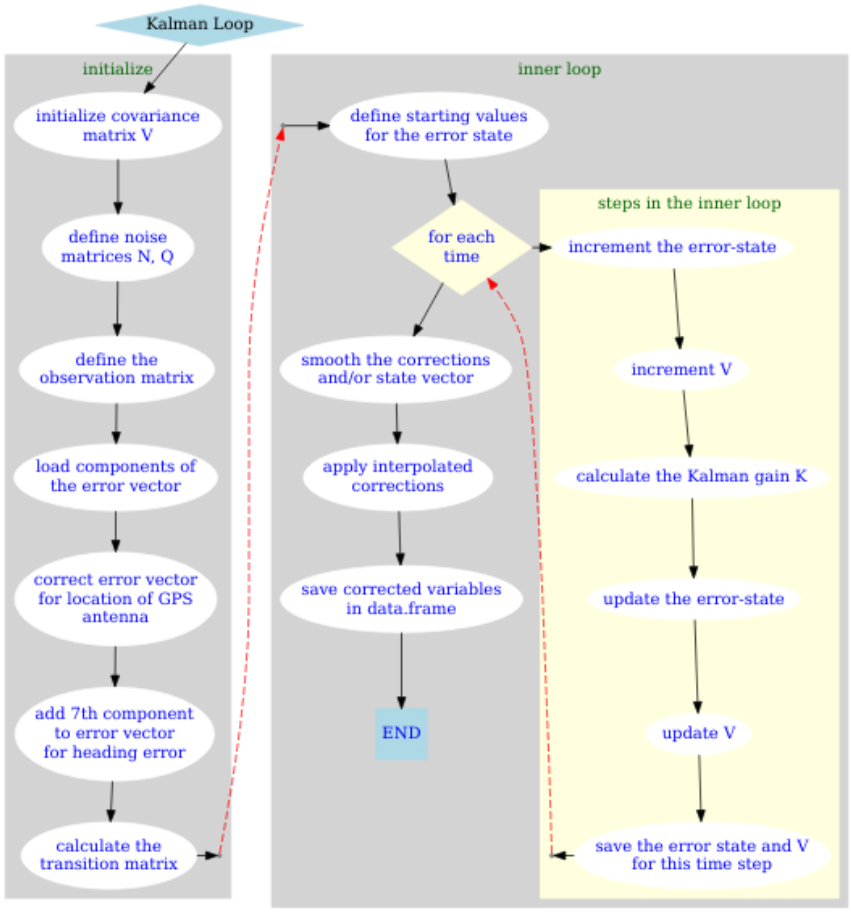


Figure 6: Workflow diagram for the main loop of the Kalman filter. The yellow box contains the steps that are repeated for each time step, using the equations listed in the algorithm summary in the technical note.

7 Discussion of Section 4 (the Kalman filter)

7.1 General comments on the structure and the algorithm

The structure and algorithm are discussed in detail in the technical note. A flowchart describing the algorithm and the processing related to the Kalman filter is shown in Fig. 6. This diagram may be helpful when reading the description of the algorithm in the technical note. These elaborating comments may also be worth recording here:

1. The process of smoothing the results after the processing loop finishes is described in places distributed through several subsections of the technical note where various plots of the results are shown, so a summary here may be useful:

(a) When extrapolating from the Kalman-filter time step to the data time step, the extrapolation is performed by the R routine "stats::approx()", and after extrapolation the result is smoothed by a Savitzky-Golay polynomial of 4th order that spans 75 s in the output data, and the start and end of the resulting time series are padded so that the final series matches the original series in time and in length.

(b) For position and velocity components, a low-pass Butterworth filter is used (with cutoff frequency corresponding to a period of 600 s) to smooth the error-state components.

(c) For latitude and longitude, the variables after correction are filtered again using a low-pass Butterworth filter with cutoff frequency corresponding to a period of 10 s, to eliminate resolution noise in the original measurements.

(d) For pitch and roll, the corrections after extrapolation are transformed to the *l*-frame, filtered using a low-pass Butterworth filter with 900 s cutoff period, and then transformed back to the *a*-frame where they are used to correct the original measurements.

(e) Heading is treated in a special way by using a weighted spline applied to the corrections from the Kalman filter, where the weight factor is the inverse of the corresponding variance from the Kalman filter. That spline is then used to determine the time history of the heading correction, and that correction from the spline fit is used to correct the measurement of heading.

2. The calculation of the transition matrix in principle can be done outside the processing loop. This matrix is not required for every time step in the state vector, but only for every time used in the inner loop, so only a subset needs to be calculated and saved. After completion of the calculations in the inner loop, the corrections that are obtained can then be extrapolated to times between the larger time step used in the inner loop. In this way high-rate data files, with measurements more frequent than 1 Hz, can be processed almost as fast as standard-rate files.

7.2 The vectors and matrices

The main report defines some vectors and matrices used in the error-state Kalman filter. Here some of the associated R-code and variables are included to make these definitions specific.

SVE — At initialization, the *error-state vector* is set to SVE <- c(DZ[1, 1:6], rep(0, 9)) where DZ[1, 1:6] is a vector containing the initial values of the differences between the first six INS-provided state variables and the corresponding GPS-provided variables. That is, DZ=c(LAT-GGLAT, LON-GGLON, ZROC-GGALT, VEW-GGVEW, VNS-GGVNS, ROC-GGVSPD) where all values are read from the data archive at the initial time. All angular quantities at initialization and during the integration are used in units of radians, including latitude and longitude.

dcm — The *error-state transition matrix* $T_{k,k-1}$ is calculated, for a specified time during the integration, as follows:

- The 15 components of the state vector sv from the INS mechanization are read from the data archive.
- dcm <- jacobian (STMFV, sv, .aaframe='l') * dt * NSTEP + diag(15)

The argument to the jacobian function is the derivative function STMFV discussed above, the state vector, and .aaframe which in this case specifies that the attitude angles should be the *a*-frame values, as is discussed below. The variable dt contains the time interval between samples, but for increased speed the integration is performed in larger time steps specified by NSTEP, here set to 5 s.

CV — The *covariance matrix* is updated during the integration, but it is set initially using values specified in the technical note. The squares of these values are entered into the diagonal elements of CV, and off-diagonal elements are left zero at the start of the integration. These values were selected to be reasonably consistent with the specifications of the INS where known.

K — The *Kalman-gain matrix* will be discussed in subsequent sections. It is calculated independently at each step during the integration, so it does not need to be initialized.

DZ — The *measurement-error vector* as determined from comparison between the INS-provided variables and the corresponding GPS-provided values. This is initialized in a data.frame at the start of the integration, where that data.frame contains all the measurements from the flight. For the error-state Kalman filter, 1-Hz measurements were sufficient, so a standard-rate data file was used. The values for each time during the integration are then extracted from that data.frame. Some preliminary steps were included:
- interpolation to provide any missing-value measurements
- optionally, smoothing with a third order Savitzky-Golay polynomial spanning 11 s.

H — The *observation matrix* for the initial implementation is a 15×6 matrix that indicates that the six components of the measurement-error vector DZ correspond to the first 6 components of the error-state vector. In R convention ordering (which stores matrices in column-major order), H has 6 rows and 15 columns.

Q and R — The *measurement-noise matrices* that apply to the error-state vector and the measurement-error vector, respectively. These are not adjusted during the integration, so they need to be specified using reasonable values. Those values can depend on functions of the aircraft-state vector, so the matrix does not necessarily remain the same for each step in the integration, but they can be specified prior to the integration and saved in data.frames so that appropriate values can be read during the integration. They are best specified iteratively, by finding adjustments that produce results consistent with the error estimates, so the process here will be to start with a simple specification for each matrix and then later consider how this specification should be changed. For Q, the initial choice was a diagonal matrix as specified in the text. There are important off-diagonal elements that should be included, e.g., to represent strong coupling between the pitch error and the latitude error, but this will be reconsidered later.

7.3 Detecting the error in heading

The discussion in the technical note is extensive so no expansion is provided here. The addition of a seventh component to the measurement-error vector is not conventional and it may be redundant in the sense that the Kalman filter already makes appropriate adjustment to the errors to minimize errors in velocity so using the direction of the acceleration provided by differentiating the GPS-based measurements does not add information not already present. However, because heading is treated specially because of the large uncertainty involved in its estimated error, the inclusion of this term appears to improve performance of the correction scheme as tuned in this application. There is a controlling factor, "MH", that if set to zero will suppress the effect of this added term.

7.4 Smoothing the errors in pitch and roll.

Section 4.3 and Eq. (1) defined and justified the transformation from an a-frame vector to an l-frame vector. Here that transformation is adapted to the errors in pitch and roll. Approximate equations (technical note Eqn. 27) are discussed in Sect. 4.4.3 of the technical note, but exact equations are used in the code. Those equations are developed here.

In the a-frame where pitch and roll are measured, the roll angle is defined as the reverse of the rotation about the a-frame forward axis required to level the wings, and the pitch angle is the reverse of the subsequent rotation about the starboard axis required to level the fuselage. The l-frame error in pitch is defined to be the platform misalignment with tilt to the south after the roll error is removed,, while the l-frame error in roll is the component of platform misalignment toward the east. If the l-frame errors are known, one can find the a-frame errors by considering a unit vector $u^{(l)}$ normal to the sensed platform alignment in the approximately upward l-frame direction, with components that would result from rotating a unit vector that points in the l-frame z direction by the pitch error ($\delta\theta^{(l)}$) about the x-axis and then by the roll error ($\delta\phi^{(l)}$) about the y-axis. The result of these two rotations is

$$\begin{aligned} u^{(l)} &= \begin{bmatrix} \cos(\delta\phi^{(l)}) & 0 & \sin(\delta\phi^{(l)}) \\ 0 & 1 & 0 \\ -\sin(\delta\phi^{(l)}) & 0 & \cos(\delta\phi^{(l)}) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta\theta^{(l)}) & -\sin(\delta\theta^{(l)}) \\ 0 & \sin(\delta\theta^{(l)}) & \cos(\delta\theta^{(l)}) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \sin(\delta\phi^{(l)})\cos(\delta\theta^{(l)}) \\ -\sin(\delta\theta^{(l)}) \\ \cos(\delta\theta^{(l)})\cos(\delta\phi^{(l)}) \end{bmatrix} \simeq \begin{bmatrix} \delta\phi^{(l)} \\ -\delta\theta^{(l)} \\ 1 \end{bmatrix} \end{aligned}$$

Transformed to the a-frame via the inverse of (1), this unit vector becomes

$$\begin{aligned} u^{(a)} &= (R_l^a)^{\mathsf{T}} u^{(l)} \\ &= \begin{bmatrix} \sin\psi\cos\theta\,\delta\phi^{(l)} - \cos\psi\cos\theta\,\delta\theta^{(l)} - \sin\theta \\ (\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi)\,\delta\phi^{(l)} - (\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi)\,\delta\theta^{(l)} + \cos\theta\sin\phi \\ (\cos\psi\sin\phi - \sin\psi\sin\theta\cos\phi)\,\delta\phi^{(l)} + (\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi)\,\delta\theta^{(l)} - \cos\theta\cos\phi \end{bmatrix} \end{aligned}$$

The difference between this unit vector and the corresponding unit vector obtained without any errors in pitch or roll is then

$$\delta u^{(a)} = \begin{bmatrix} -\cos\psi\cos\theta\,\delta\theta^{(l)} + \sin\psi\cos\theta\,\delta\phi^{(l)} \\ -(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi)\,\delta\theta^{(l)} + (\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi)\,\delta\phi^{(l)} \\ (\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi)\,\delta\theta^{(l)} + (\cos\psi\sin\phi - \sin\psi\sin\theta\cos\phi)\,\delta\phi^{(l)} \end{bmatrix}$$

In the a-frame, the error in roll is $\delta\phi^{(a)} = \delta u_y^{(a)}$ and the error in pitch is

$$\delta\theta^{(a)} = -\cos(\delta\phi^{(a)})\delta u_x^{(a)} \simeq -\delta u_x^{(a)} \ ,$$

so the transformation of (small) errors from the l-frame to the a-frame is

$$\begin{aligned} \delta\theta^{(a)} &= \cos\psi\cos\theta\,\delta\theta^{(l)} - \sin\psi\cos\theta\,\delta\phi^{(l)} \approx \cos\psi\,\delta\theta^{(l)} - \sin\psi\,\delta\phi^{(l)} \\ \delta\phi^{(a)} &= -(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi)\,\delta\theta^{(l)} + (\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi)\,\delta\phi^{(l)} \quad (16) \\ &\approx \cos\phi(\sin\psi\,\delta\theta^{(l)} + \cos\psi\,\delta\phi^{(l)}) \end{aligned}$$

where the approximations are that $\cos\theta \approx 1$ and $\sin\theta \approx 0$. These are reasonable because the pitch angle seldom exceeds a few degrees, but a similar approximation cannot be made for roll because the roll becomes significant in turns through the factor $\cos\phi$ in the last equation.

Equations (16), without approximation, were coded into the function XPitch(). That function also has an argument .inverse that, when true, calculates to reverse transformation from the a-frame to the l-frame, obtained from the inverse of (16). Because the transformation is not orthonormal, the result is not simply the transpose; it is given by

$$\delta\theta^{(l)} = \frac{1}{\cos\theta\cos\phi}\left[(\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi)\,\delta\theta^{(a)} + \sin\psi\cos\theta\,\delta\phi^{(a)}\right]$$

$$\delta\phi^{(l)} = \frac{1}{\cos\theta\cos\phi}\left[(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi)\,\delta\theta^{(a)} + \cos\psi\cos\theta\,\delta\phi^{(a)}\right]$$

7.5 Generation of Figs. 6–11 in the technical note

These figures mostly follow the procedure discussed above (cf. Sect. 5.5) for Figs. 1–3. The faceting capability of the R package "ggplot2" was used to produce these multi-panel plots. One new feature in Figs. 8 and 9 is the addition of a shaded ribbon to show the uncertainty estimate about the correction from the Kalman filter. This is constructed by defining a special data.frame that contains the upper and lower limits of the desired ribbon, "melting" that data.frame as for the basic plot, and using the function "geom_ribbon()" from ggplot2 to add the ribbon. Figure 11 also is different; it uses the "position" argument to "Ranadu::ggplotWAC()" to define the stacked panels, and it forces the ordinate scales to be the same to keep the size of the panels the same.

8 Discussion of Section 5 (new variables for wind)

8.1 Calculating the wind

The function "Ranadu::WindProcessor()" calculates new values of the wind by duplicating the standard calculations, with the exception that it includes a correction for the motion of the GPS antenna where relevant. The procedures are documented in Lenschow and Spyers-Duran (1989), and the code is available in the GitHub archive for Ranadu.

This section includes plots of variance spectra showing the character of the new variables after correction by the Kalman filter. The generation of these plots is a departure from the effort to make this study reproducible, because the plots were generated using copyrighted code (from Press et al. (1992)). The plots therefore were generated outside the R program and were inserted as figures. The specific algorithm used is that described on pp. 568–575 of that reference. The C code that generates these figures, without the copyrighted routines, is saved with this project archive as "otto.c".

8.2 Adding new variables to the netCDF file

The template included in the technical note is the model used for adding variables. The code actually used in is R chunk "modify-new-netcdf.R," which is shared with the processing code "KalmanFilter.R" that is designed to process existing NCAR/EOL/RAF data files in netCDF format and add variables after correction by this Kalman Filter. The netCDF interface is based on the R package "ncdf4" (Pierce (2015)).

9 The associated processor ("KalmanFilter.R")

9.1 Getting started

The processing discussed in the Technical Note has been incorporated into a processor that can be used outside the structure of the file that generates that file. When the GitHub repository is retrieved, the appropriate code for the processor will also be downloaded along with the R "chunks" needed to run it. RStudio is strongly recommended as the initial way to run this, but the repository can also be downloaded outside RStudio and used with R. The following are detailed instructions for downloading using RStudio. These should work with minor exceptions in linux, Windows, or Mac computers. A few exceptions are listed in an Appendix, and some of the following instructions are written for linux and may need some adaptation for Windows or Mac systems.

1. In your home directory, make a directory named RStudio. The R code will then be downloaded into the "KalmanFilter" subdirectory, and some program components expect this structure, so installation will be much smoother if you follow this structure.

2. Start RStudio. The manual for Ranadu provides some instructions for obtaining RStudio and R if they are not present on your computer. Follow the instructions in that manual to install the package ‘‘Ranadu’’ and also add these packages to your installation if they are not there: numDeriv, ggplot2, reshape2, grid, knitr, graphics, ggthemes, ncdf4, shiny, signal and zoo.

3. Use `File -> New Project -> Version Control -> Git` and provide this repository address: `https://github.com/WilliamCooper/KalmanFilter.git` . Use the Project directory name ‘‘KalmanFilter’’ (without quotes) and add it as a subdirectory of ‘‘~/RStudio’’. Click the ‘‘Create Project’’ button and the project files will be downloaded and installed as a new project in RStudio.

Note that this is not necessary if you use the web interface to the EOL computer “barolo.eol.ucar.edu”. The easiest way to use the Kalman-filter processor on barolo is to follow the instructions in the Ranadu manual as the link given above. Then copy my directory on barolo to your workspace (“cp -r ~cooperw/RStudio/KalmanFilter ~/RStudio/.”) You can then use the web interface to RStudio on barolo: (a) If you are not “inside” VPN in; use this URL: “barolo.eol.ucar.edu:8787” and a web version of RStudio will start; (b) use `File -> Open Project` to open the KalmanFilter project. You may need to use `File -> New Project -> Existing directory` and select the KalmanFilter subdirectory to have RStudio recognize this as a project.

9.2 Key components of “KalmanFilter.R”

The processor uses many components from the Technical Note via the use of “source()” statements in the R code of each, to guarantee that the identical code is used for key components. The following is a list of such “chunks”:

RotationCorrection.R A function that corrects measured accelerations during transformation between the *l*-frame and *a*-frame.

STMFV The function providing the derivative of the state vector.

AcquireData.R Code for reading the data file to an R-language data.frame and making some initial adjustments. This chunk also includes another, called **AddIRUVariables.R**, to add surrogate measurements from the IRU to the data.frame if the original measurements of acceleration and rotation rate are not present in the data file.

ROC.R Adds the new rate-of-climb variable “ROC” to the data file.

AdjustCal.R Applies calibration adjustments as determined in the Technical Note.

Kalman-setup Defines the covariance and other matrices and vectors needed for the Kalman filter.

Kalman-loop Processes the data file sequentially using the Kalman-filter algorithm, then defines corrected variables.

wind-calculation Uses the corrected values from the Kalman filter to calculate new values of the wind.

create-new-netcdf.R Copies the original data file and prepares it for addition of new variables.

modify-new-netcdf.R Adds the new variables with appropriate attributes to the new netCDF file, then writes it to storage.

The structure of "KalmanFilter.R" follows that described in the workflow diagrams shown as Figs. 1 and 3-–6, except that the "mechanization" and "compare results" steps in the left-side box of Fig. 1 are omitted. Otherwise, the program flow duplicates that in the program constructing the Technical Note except for new specifications of run parameters like the data-archive file to be used, as described in the next subsection.

9.3 Instructions for using "KalmanFilter.R"

The Kalman processor does not duplicate the early part of the Technical Note where the derivative is justified by performing a trial mechanization. It also differs from the document generating the Technical Note by including features to tailor it for particular runs. Because the key mathematic structure is identical to that in the Technical Note, the algorithms used will not be discussed here. Instead, this section of the workflow document will emphasize the user interface.

There are three ways that the processor can be run. (The third is recommended, but the second will support batch processing best.)

1. From an R console window or the console window in RStudio, after installation as described above. If you type "source ('KalmanFilter.R') when in the working directory ~/RStudio/KalmanFilter, the script will ask interactively for the run arguments like project name, flight number, and other processing options. These are described in detail for the third option listed below.

2. With working directory set to "~/RStudio/KalmanFilter", type "Rscript KalmanFilter.R [arguments]". If you omit the arguments, the script will print a brief usage statement and exit, so you can use the model it provides to repeat with appropriate run-time arguments. The usage statement is as follows:

```
"Usage:  Rscript KalmanFilter.R Project Flight UpdateAKRD[y/N] UpdateSSRD[y,N]
                Simple[y/N] Interval''
 "Example:  Rscript KalmanFilter.R CSET 1 y y n 15"
```

   The arguments are explained in the list following item 3.(d) below. This second option may be useful for batch processing.

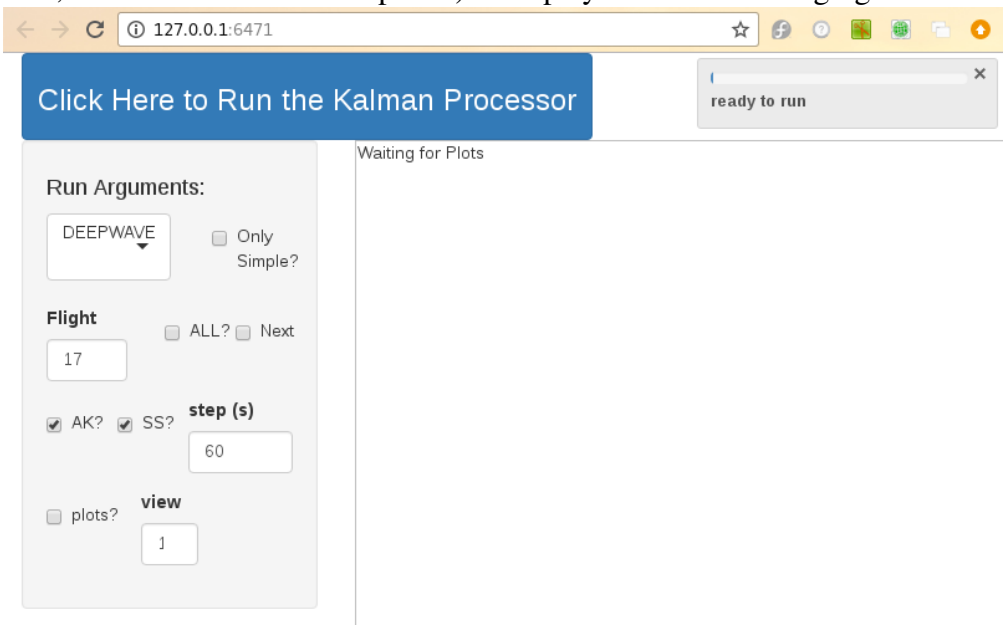
3. The most convenient option is probably the "Shiny App" version that provides an interactive graphic interface for running the program and displaying some key results. From RStudio, select the "KalmanFilter" project, and then run the ShinyApp, as follows:

(a) Select the "Files" tab at the top of the bottom-right RStudio panel (*not* the "File" menu at the top-left).

(b) In the list of files and directories that appears, click on the directory "KFapp". Then the list of files will include one named "app.R". Click on that and it will be displayed in the top-left RStudio panel.

(c) There will now be a "Run App" button just right of a green arrow in the panel at the top of the source file. Before clicking that, use the small down arrow just right of the "Run App" button to display a menu of choices. There are different ways of displaying the ShinyApp. I suggest "Run External" but you may want to experiment with the other choices later. "Run External" wil display in a new browser window.

(d) Once you have made a display selection, click the "Run App" button. (After selecting the display mode, you can also enter "runApp('KFapp')" in the RStudio console window, which is the bottom left panel.) A display like the following figure should appear:



This then provides ways of selecting the data-archive file to process, and it provides control over some options. The components in this window and the options that can be selected are described here:

i. **Click here:** The large blue button labeled "Click Here..." starts the processing run, once the choices of "Run Arguments" are made.

ii. **(progress meter):** Because the processing runs are sometimes long (depending especially on the entry for "step (s)"), a progress meter is included at the top right. In this figure it says "ready to run", and once the run starts it will indicate progress of each step in processing including reading and initial processing of the data file, retrieval of the IRU measurements if they are missing, the main Kalman loop (which consumes most of the time), the generation of plots, and production of the final modified netCDF file. A run with step size 60 (which is too large to be useful

except for testing) will take about 3 min, while a practical run with step size of 10 will take about 11 min and the recommended step size of 5 s will run for typically about 21 min on barolo.

iii. **Project:** The item labeled "DEEPWAVE" in the figure is a drop-down menu listing all GV projects, from which you can select a file for processing. For this to work, the data structure should match a particular pattern, as described in the RanaduShinyManual. When new projects occur, they need to be added to the code in "KFapp/app.R" in the character vector "PJ" near the top of the routine. The first step in defining the processing run desired is to select the project name from this drop-down menu.

iv. **Flight:** The entry box below the label "Flight" is used to specify the desired flight number. The number in this box can be changed in several ways: by selecting the displayed number with a left-button mouse sweep over it and then entering the desired number, by left-clicking in the entry window and then using the up-arrow and down-arrow keys on the keyboard, or by using the small incrementing/decrementing arrows that appear to the right of the displayed number when the mouse hovers there. Other options for specifying the flight number are controlled by the "ALL?" and "Next" check-boxes to the right of this entry window, as described below.

v. **ALL?:** The "ALL?" check-box needs to be used with caution because it can result in long runs that are not interrupted easily. When checked, this will cause all flights in a project *that do not already have a processed file with "KF" at the end of the name* to be processed. If such files are present and you want to reprocess, you need to delete or move the previously processed files from the project directory. A project directory containing 20 flights, processed with a time step of 5 s, can take almost 8 h. (On the EOL machine barolo, be sure to use the "renice" feature to change the priority to avoid interference with other users, because the run makes intensive use of the CPU.) For full-project runs, it will usually be better to use method-2 processing above with a script that processes each file, because that is easier to interrupt. To interrupt the ShinyApp processor, dismiss the special window and click the red "STOP" button at the top-right of the bottom-right panel of the RStudio display. Processing of the current file will still continue, as well as possibly the next one if that has been queued by the controlling script. To stop everything, you have to terminate the R session.

vi. **Next:** The "Next" checkbox processes the file numbered one beyond the highest-numbered file that has already been processed. For example, if the data directory for the selected project contains files DEEPWAVErf06KF.nc, DEEPWAVErf16KF.nc, the next run will used flight 17.

vii. **AK** and **SS:** The check-boxes "AK" and "SS" control, respectively, if the new algorithm for angle-of-attack should be used to calculate a new value for the AKRD/ATTACK variables and if the sideslip calibration recommended in Cooper et al. (2016) should be used. If these are selected, the new angle-of-attack and

sideslip will be used in the new calculation of wind. The "AK" check-box should not be used (and will not be used even if checked) for projects flown on the GV before 2012 (i.e., TORERO and earlier) because an earlier radome with characteristics not described well by the new algorithm was used in those projects.

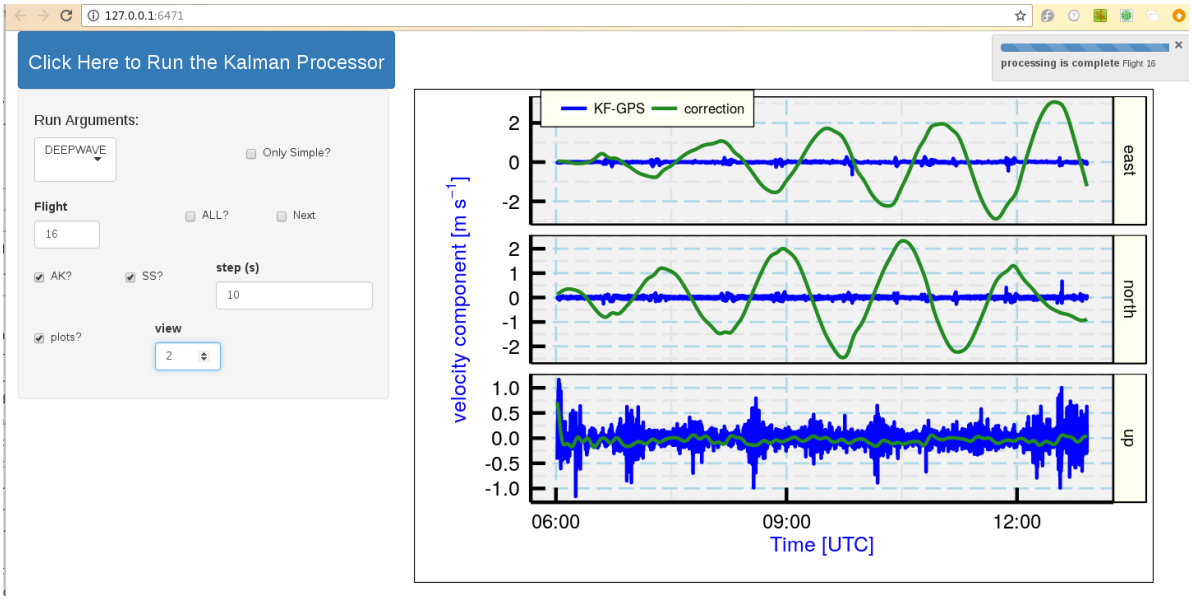
viii. **step (s):** The entry box labeled "Step (s)" specifies the interval between updates in the Kalman-filter loop. This should be set to some value between 5 and 15 for normal processing, although higher values may be useful for testing. Lower values increase processing time, which is dominated by the Kalman loop and so is almost proportional to the number of steps taken. (A rough approximation on barolo is that the processing time will be about 100 min/(step_size) for a 7 h flight.) A value of 5 is recommended for normal processing, although 10 will usually produce satisfactory results also.

ix. **plots?:** The "plots?" check-box controls the generation of plots showing the results of processing. If checked, eight plots are generated for the specified flight, and those plots can be displayed when the run finishes by using the "view" numeric entry to cycle through the plots. These plots are similar to those shown in the Technical Note to characterize the performance of the Kalman filter, and they have been generated using the same code.

x. **view:** The "view" entry window selects the plot to be displayed, and will only be functional after the run completes. The plots show the Kalman-filter results for: (1) position; (2) aircraft velocity; (3) pitch angle in the *l*-frame; (4) pitch angle in the *a*-frame; (5) heading; (6) vertical wind; (7) horizontal wind; and (8) heading correction from the simplified "CorrectHeading()" function (not used in normal processing).

xi. **Only Simple?** This check-box should not be checked in most cases. If it is checked, the Kalman processing is skipped and the changes made to the data file are made by the simplified correction functions "CorrectPitch()" and "CorrectHeading()" that are discussed in the Technical Note., as well as the revisions to angle of attack and sideslip if those check-boxes are selected. In that case, the final output variables for the pitch, roll, heading, vertical wind, horizontal wind direction and horizontal wind speed are respectively PITCHC, ROLLC, THDGC, WISC, WDSC, and WSSC.

When the run finishes, if the "plots?" check-box is selected, the display window should look like this:



The plots are saved to a tar file in the subdirectory KFplots, named ProjectPlotsFN.tgz where 'Project' is the project name and FN the flight number. The processor is then ready to run again. When it does, the plots are cleared from that directory but the tar file is left, so it can be examined later. This is particularly useful for all-project runs. The last plot in the sequence of plots is that generated by the "CorrectHeading()" function. If there are too few qualifying turns to produce a valid estimate, that plot will not be present and the plotted line for "HC" in the plot of heading will be a steady default value.

The result of the processing run is a new netCDF file with variables added as described at the end of the Technical Note.

References

Cooper, W. A., Friesen, R. B., Hayman, M., Jensen, J. B., Lenschow, D. H., Romashkin, P. A., Schanot, A. J., Spuler, S. M., Stith, J. L., and Wolff, C.: Characterization of uncertainty in measurements of wind from the NSF/NCAR Gulfstream V research aircraft, proposed NCAR technical note, undergoing review NCAR/TN-XXX+STR, Earth Observing Laboratory, NCAR, Boulder, CO, USA, URL https://drive.google.com/open?id=0B1kIUH45ca5AT2NwVFpqRFNOZ0U, 2016.

Lenschow, D. H. and Spyers-Duran, P.: Measurement techniques: air motion sensing, Tech. rep., National Center for Atmospheric Research, URL https://www.eol.ucar.edu/raf/Bulletins/bulletin23.html, 1989.

Noureldin, A., Karamat, T., and Georgy, J.: Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration, SpringerLink : Bücher, Springer Berlin Heidelberg, doi: 10.1007/978-3-642-30466-8, URL https://books.google.com/books?id=qGbsW6sDr7YC, 2013.

Pierce, D.: ncdf4: Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files, URL https://CRAN.R-project.org/package=ncdf4, r package version 1.15, 2015.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P.: Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing, Cambridge University Press, New York, NY, USA, 1992.

R Core Team: R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, URL http://www.R-project.org, 2016.

RStudio: RStudio: Integrated development environment for R (Version 0.98.879), URL http://www.rstudio.org, 2009.

Xie, Y.: Dynamic Documents with R and knitr, Chapman and Hall/CRC, Boca Raton, Florida, URL http://yihui.name/knitr/, iSBN 978-1482203530, 2013.

Xie, Y.: knitr: A general-purpose package for dynamic report generation in R, URL http://yihui.name/knitr/, r package version 1.6, 2014.

– End of Memo –

Reproducibility information for this workflow document:

PROJECT:	WorkflowKalmanFilter
ARCHIVE PACKAGE:	WorkflowKalmanFilter.zip
CONTAINS:	attachment list below
PROGRAM:	WorkflowKalmanFilter.Rnw
DIAGRAMS:	dot/*, FlowDiagrams/*
GIT:	https://github.com/WilliamCooper/KalmanFilter.git

Attachments:
WorkflowKalmanFilter.Rnw
WorkflowKalmanFilter.pdf
WAC.bib
dot/*
FlowDiagrams/*
otto.c
SessionInfo