HIRDLS

SC-HIR-2048

HIGH RESOLUTION DYNAMICS LIMB SOUNDER

Originator: Dan Packman Date: June 5, 2013

Subject/Title: **HIRDLS Level 3 Software Overview**

Program "latseq" and Preprocessor "h3ing"

The Kalman Filter Mapper is a program to convert asynoptic profiles of atmospheric quantities (typically temperature and mixing ratios) to synoptic fourier components in longitude. In addition, the mapper provides spike detection and gridding in latitude

Keywords: L3 software, latseq, h3ing

Purpose of this Document:

Reviewed by: Date (yy-mm-dd):			

Oxford University
Atmospheric, Oceanic & Planetary Physics, Clarendon Laboratory,
Parks Road,
OXFORD OXI 3PU,
United Kingdom.

University of Colorado at Boulder
Center for Limb Atmospheric Sounding,
3450 Mitchell Lane, Bldg. FL-0,
Boulder, Colorado,
80301-2296,
United States of America.

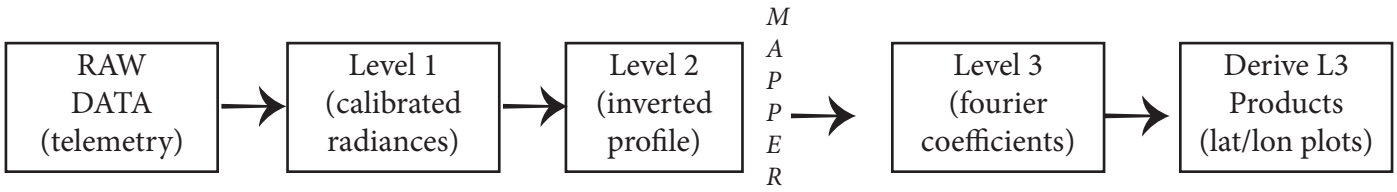
EOS

HIRDLS
L3 Software Documentation

Program latseq and preprocessor h3ing

Introduction

The Kalman Filter Mapper is a program to convert asynoptic profiles of atmospheric quantities (typically temperature and mixing ratios) to synoptic fourier components in longitude. In addition, the mapper provides spike detection and gridding in latitude. In the context of data analysis for satellite data (the original use of the mapper), the mapper is used within the overall data now like this:



Telemetry (counts, for example) is read into L1 program. The telemetry is scaled and processed into radiances. These radiances are then read in the L2 process. This is the logical center of the process where the physics involved in understanding the source of the radiance is used in inverting the process and in finding the physical quantity of interest (eg temperature or mixing ratio). This asynoptic physical quantity is then read into the mapper.

Actually, the L2 information must be rearranged before it is presented to the mapper. Typically, the L2 information contains a profile of the parameter of interest as a function of height for each data time. This profile is interpolated to standard pressures (or heights or layers or potential temperatures). Each standard pressure is mapped separately. Symbolically, then, the mapper takes a series of five elements(y_i=field of interest such as temperature, t_i=time of observation, x_{lon}=longitude, x_{lat}=latitude, Δy=error or repeatability of observation)and outputs a set of fourier coefficients **x** for each specified map time:

$$\left\{ \begin{matrix} y_i \\ t_i \\ x_{lon} \\ x_{lat} \\ \Delta y_i \end{matrix} \right\} \rightarrow \left\{ \textit{Kalman Filter} \right\} \rightarrow \mathbf{x}$$

If we define a vector k with elements k_1=1, k_{2n}=cos(nθ), k_{2n+1}=sin(nθ), n=1 to the number of waves, then the background or large scale field defined by these coefficients at a particular maptime and longitude θ can be found from the dot product:

$$\mathbf{k} \bullet \mathbf{x}$$

As we shall see, the difficulties in mapping are created by data voids in time and/or space.

Mapper Overview

Kalman filtering a given pressure proceeds in general like this:

1. Initialize the mapper: Set up an estimate **x**, a covariance **S**, and statistics ds/dt. These apriori (often shown with a subscript "a") values may be set up from a previous mapper run (perhaps an earlier data sequence) or from an initial guess. The guess may be based on a least squares estimate of the field or a zonal mean can be used and the mapper run backwards from a short distance into the data. This last option is how this program is used on HIRDLS data. This initialization proceeds from line 184 to 409 of *mapdat.f* . The call to initstat sets up the statistics ds/dt.

2. Run through the data forward in time. Update the estimate and covariance for each data point. This update proceeds in two steps. In the firs tstep, the expected time evolution of the estimate and covariance is applied to make the so-called *apriori* estimate **x**, and covariance $\mathbf{S}_i$. Then the data point(s) are used to update the estimate and the covariance. If a point is determined to be of improbable value (much too far from the apriori, that is, spike detection), it is given a very large uncertainty so it will not skew the estimate. Forward mapping proceeds in *mapdat.f* from line 410 to 644.

3. Run through the data in reverse time. Again, update estimate and covariance. Reverse mapping proceeds in *mapdat.f* from line 645 to 796.

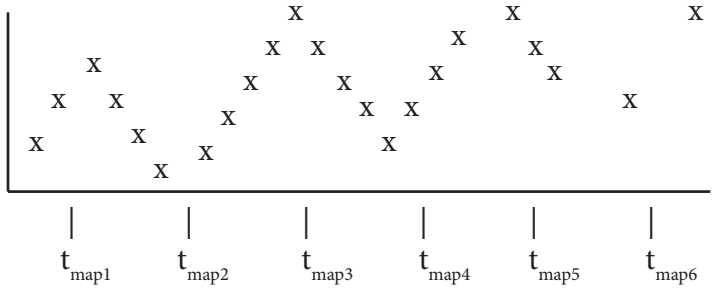
4. Combine forward and reverse estimates (routine *fnlmap.f*). Calculate new statistics ds/dt (line 986 of *mapdat.f*)

5. Use a convergence criterion of ds/dt or number of iterations (line 1003 of *mapdat.f*) to determine if iterations are needed.

6. If we must iterate, use the new calculated ds/dt and go to step 2.

7. If we are done, output the combined estimate and covariance (routine *mkmat.f*)

The guts of the procedure is how the estimate and covariance are updated. For this version of the Kalman filter, we use a simple model of a zonal mean and n_{wave} (often 6 or 7) waves in longitude yielding a total of $1+ 2*n_{wave}$ components to fit. A more complex model might explicitly include time dependent wave components (moving and/or stationary waves) and might take into account more complex time dependence of the covariance. Schematically:



»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»map forward»»»»»»»»»»»»»»»»»

[save forward]

«««««««««««««««««««««««««««««« map reverse«««««««««««««««««

[combine forward and reverse]

The Update Equations

The update equations have been given by Rogers (1976):

$$\hat{\mathbf{S}} = \mathbf{S}_a - \mathbf{S}_a\mathbf{K}^T(\mathbf{K}\mathbf{S}_a\mathbf{K}^T + \mathbf{S}_\varepsilon)^{-1}\mathbf{K}\mathbf{S}_a$$

$$\hat{\mathbf{x}} = \mathbf{x}_a + \mathbf{S}_a\mathbf{K}^T(\mathbf{K}\mathbf{S}_a\mathbf{K}^T + \mathbf{S}_\varepsilon)^{-1}(\mathbf{y} - \mathbf{K}\mathbf{x}_a)$$

In this vector form we can represent an update of an arbitrary number of latitudes at an arbitrary number of pressures (or heights, or layers, etc.) simultaneously, that is, with the same *apriori* estimate $\mathbf{x}_a$ and covariance $\mathbf{S}_a$ With multiple computational elements, this might be the optimum approach, but for this program we take the opposite approach and use the scalar form of the update equations with y becoming a set of scalar quantities y_i which span the latitudes, pressures, and times of interest. The matrix inverse then becomes a scalar reciprocal:

$$\hat{\mathbf{S}} = S_a - S_a k_i k_i^T S_a \,(k_i^T S_a k_i + \sigma_i^2)^{-1} \qquad \hat{x} = x_a + S_a k_i (y_i - k_i^T x_a)\,(k_i^T S_a k_i + \sigma_i^2)^{-1}$$

In the trivial limit of a scalar model (that is, n_{wave}=0 and we fit only a zonal mean) the fourier vector $\mathbf{k}_i$ becomes unity and in the same way we subsituted the scalar quantity σ_i^2 for $\mathbf{S}_\varepsilon$ we substitute σ_a^2 for $\mathbf{S}_a$ Applying these simplifications to the above equations yields:

$$\hat{S} = \sigma_a^2\sigma_i^2/(\sigma_a^2 + \sigma_i^2) \qquad \hat{x} = (x_a\sigma_i^2 + y_i\sigma_a^2)\,/\,(\sigma_a^2 + \sigma_i^2)$$

In this limiting case, it is clear that the update procedure merely combines the apriori and the measurement according to their statistical weights. If off-diagonal terms in the covariance matrix are sufficiently small, this conclusion is true for $n_{wave} > 0$. We can calculate the ratio

$$k_i^T S_a k_i \,/\, \sigma_i^2$$

If this ratio is large , that is, the uncertainty in the field is much larger than the uncertainty associated with the data point, then the updated estimate will pass through the data point and the *apriori* estimate will have vanishing import. If this ratio is small, that is, the uncertainty in the field is much smaller than the uncertainty associated with the data point, then the updated estimate and covariance will not change appreciably. In most situations where the data point represents valuable information with some real uncertainty , this ratio should be on the order of unity. In most situations, we desire the ratio to be some fraction of unity so that the updated estimate is somewhat more dependent on the data point than on the *apriori* estimate.

Of course, the scalar ratio discussed above only gives an indication of how much relative weight is assigned to a given point. The elements of the *apriori* covariance matrix parcel this information out amongst the various wave coefficients. Again, a special limiting case can help illustrate how the procedure works.

If each wave component evolves independently, then the covariance matrix will be diagonal. While in practice this ideal is not achieved, typically off-diagonal elements of the covariance matrix are an order of magnitude less than diagonal terms. If we let S_{ai} be the ith diagonal element of the *apriori* covariance matrix (i=0 to $2n_{wave}$) then the relative weighting of a given wave component (i>0) to the zonal mean is:

$$\tfrac{1}{2}\, S_{ai} \,/\, S_{a0}$$

Again, for a typical case, the wave spectrum will be red and we expect this weighting ratio to decrease for larger

wave numbers.

The *Apriori* Solution

The other half to sequential estimation is the calculation of the *apriori* solution. That is, given the estimate calculated for a previous data point y_i, find the *apriori* solution to update the estimate for data y_{i+1}. In general, any dependence on the entire available set of data points (field values, times, longitudes, latitudes, and errors) could be used to arrive at an *apriori* estimate at time *i+1* . The better this estimate is, the less sensitive the update equations are to errors in the covariance matrix since proportionately less modification to the estimate is being made. Another advantage to an excellent *apriori* solution is that the estimate is more accurate for times *between* data points. Typically, the filtered solution is sampled at regular time spacings. If there happens to be a significant data void at a given time of interest, the solution reported at that time will be more accurate if the *apriori* calculation is accurate.

In practice, it is most convenient to restrict consideration of data points to the previous and current ones. In this incarnation of the mapper (Ver 3.2), we use a very simple scheme to arrive at an apriori estimate. We use a stored time derivative of the covariance and use it to degrade (make more uncertain) the diagonal elements of the covariance matrix at time t_{i+1} from the estimate at t_i :

$$S_a = \hat{S}_{i-1} + (t_i - t_{i-1})\, dS/dt$$

$$x_a = \hat{x}_{i-1} f_{damping}$$

where the damping factor $f_{damping}$ is a cubic that varies from unity for $\Delta t \leq .5$ days to zero for $\Delta t \geq 8$ days. This damping of wave power is implemented in routine *holdest.f* starting on lines 52 & 97 and in routine *update.f* starting on lines 46.

For dense data streams, there will never be data voids of a half day or more and the *apriori* estimate merely reduces to the previous estimate. This limiting case was used by previous incarnations of the mapper. In this version we conservatively assume that planetary waves are uncorrelated after 8 days (Khori estimated 4 days) so if we encounter a data void of that size or greater we set the wave coefficients to zero (of course, we retain the zonal mean). Partial damping of waves occurs for data voids between half a day and eight days. It has been found that without this damping procedure, the mapper can become unstable in the presence of data voids. An update after a large data void can, in the absence of wave damping, produce arbitrarily large wave amplitudes.

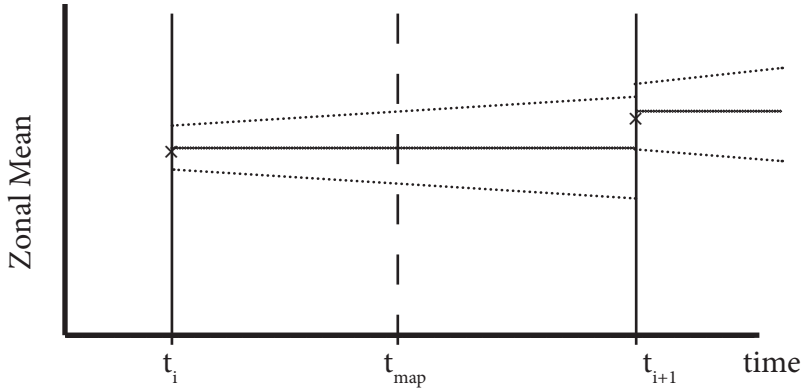
The covariance is increased by an amount proportional to the time difference (Δt) from the previous datum. We simplify this procedure by maintaining a diagonal dS/dt (we only degrade diagonal elements of the covariance matrix).

The code had been set up to allow a different Δt for each wave component and zonal mean. The logic behind this is that one must have more complete coverage in longitude to adequately define higher planetary wave numbers. Preliminary tests of this refinement indicate it is of limited utility (the estimate did not adequately follow the data points when larger values of Δt were used) but more testing is needed.

The Update Process

We now cover the update process, that is, once we obtain the five basic field quantities (field value y_i, time t_i, longitude x_{lon} latitude x_{lat} and field repeatability σ_i^2) we describe the process needed to obtain actual field values and total variances at a gridded latitude of interest. In addition, we will describe the spike detection scheme.

Gridding in time proceeds as a selection process. When we obtain the next data point, we examine its time and see if we have passed a time of interest (maptime). Ifwe have, we form the *apriori* solution at the maptime. This solution is stored for a forward mapping. If it is a reverse mapping , we combine the reverse *apriori* with the previously stored forward solution at this maptime and store the combined estimate. In a sparse dataset, it is possible that several maptimes will have to be updated at one time. In a well-covered dataset, there will at most one maptime to update. Schematically:



For simplicity, the diagram shows the zero wave case so the vertical axis is the zonal mean. The x characters depict data points at t_i and t_{i+1}. The size of these x characters depict the confidence in the data. The estimate remains within this uncertainty of the data points and slightly lags the variability of the data. After each update, the covariance (depicted by the lines on either side of the estimate) decreases representing the increased confidence associated with processed data. Between data points (for instance, at the maptime depicted, t_{map}) the covariance increases linearly.

Gridding in latitude is not quite as straightforward. In earlier versions of the mapper, the input data was constrained to already be on a standard latitude grid. In the current scheme, the input point is not constrained to be on the latitude grid. Instead, a configurable number (*nlatcon*) of grid latitudes may be updated using the input point. This number defaults to two. The gradient is found from a four-point difference formula. If we number the grid latitudes consecutively from one to four with the data point lying between 2 and 3, then the gradient is:

$$grad = \frac{(f_3 - f_1) + (f_4 - f_2)}{4\Delta x_{lat}} \qquad errlat = \frac{\Sigma_1^4 \mathbf{Tr}(\hat{\mathbf{S}})}{4(\Delta x_{lat})^2}$$

The process of applying a latitude gradient correction to the input data depends only indirectly on the latitude variance of the field. More properly, this operation depends on the uncertainty associated with the estimate since it is the estimate that is used to generate the gradient. Further, since the gradient is independent of an offset, we use an estimate of the average uncertainty of the estimate. It may be argued, then that a more correct estimate of the latitude uncertainty is:

$$errlat = \frac{\Sigma_1^4 \mathbf{k} \bullet \hat{\mathbf{S}} \bullet \mathbf{k}}{4(\Delta x_{lat})^2}$$

In numerical experiments it was found that the former method gave mapping results indistiguish able from the more exact formulation. The main virtue of the method of averaging traces is its computational simplicity. Similarly, it may be argued that higher order difference equations such as the gradient of a lagrangian interpolation are better than the scheme used. In practice, the present scheme seems quite adequate of the latitude grid is chosen to adequately describe the latitude variation of the background field . If the data coverage does not allow such a fine latitude grid, then higher order gradients cannot be calculated anyway. Schematically:

In a perfect world, adding to the input point's uncertainty due to latitude gridding would be sufficient [longitude/time gridding uncertainty is part of the calculation of the covariance *apriori*]. For real atmospheric data, fine spatial and temporal details are present in input data that are not properly part of the background field that we desire the kalman filter to reveaL This unresolved (in space and/or time) signal is effectively noise. We expect some part of this signal to be due to gravity waves and we expect strong vertical and spatial correlation over short scales. (Fetzer & Gille, 1994; Wright & Gille 2013) Despite these expectations, the noise model used in all versions of the mapper so far is gaussian.

Previous versions of the mapper either used an educated internal guess (dependent on pressure, constituent, zonal mean, and wave one amplitude), or depended on an external calculation of ds/dt, normally from a previous mapping. In the latter case, the variance of ds/dt of all waves was summed, implicitly multiplied by one day, and used as an unresolved variance. One cannot properly calculate this value from mapped quantities. In the present scheme, the unresolved component is specified as an input quantity. The designated variance associated with an input point is then composed of three values:

$$\sigma_d^2 = \sigma_i^2 + \sigma_{\Delta lat}^2 + \sigma_{unres}^2$$

the intrinsic repeatability, the error due to gridding in latitude, and the unresolved value, respectively. In previous versions of the mapper this was:

$$\sigma_d^2 = (\sigma_i^2 + \sigma_{unres}^2) e^{-\alpha(\Delta lat)^2}$$

For the LIMS dataset, we found that the component due to latitude gridding is always the smallest of the three. At low altitudes (pressures greater than 10mb), the intrinsic repeatablity of the measurements dominated. At higher altitudes, the unresolved variance (primarily gravity waves) dominated.

Occasionally, input points are totally erroneous and should be filtered out of the solution. Spike detection consists of determining if the point is beyond σ_{rej} of the estimate. That is if

$$|x_i - y| > \sigma_{rej} \quad \text{then} \quad \sigma_d \rightarrow (\sigma_d^2 + \sigma_{rej}^2)^{.5}$$

By setting the designated uncertainty of this wild point to be large, its effects on the solution are minimized. In old versions of the mapper, the cutoff for wild points was a guess of a 6 sigma event:

$$\sigma_{rej} = \max(\sigma_{min}, 6\,(\sigma_i^2 + \sigma_{unres}^2)^{.5})$$

where σ_{min} was chosen to be a constituent dependent fraction of the zonal mean (for instance, for temperature, the fraction was .05). The problem with this expression is that it only represents the uncertainty of the input

data while the comparison for spike detection is with the estimate. The current version of the mapper uses the data quality check criterion discussed by Rogers:

$$\sigma_{rej}^2 = \chi_0^2 \, (\mathbf{k} \bullet \hat{\mathbf{S}} \bullet \mathbf{k} + \sigma_d^2) \quad \text{with} \quad \chi_0^2 = 36$$

This expression correctly incorporates the estimate uncertainty. The choice of χ_o (parameter *sigrejf* in the program) as 6 is a rough analogy to the earlier expression. This really corresponds to percentages in a chi-squared distribution and the percentages are dependent on the degrees of freedom in the model. In the case of six planetary waves (cosine and sine) and the zonal mean, we have 13 degrees of freedom and the choice of 6 corresponds to less than a .5% event. Points that exceed this large error are marked as "wild", counted separately in forward and reverse mappings, and are part of the optional text diagnostic output.

This data quality check is used to detect data of unexpected value and to increase the uncertainty associated with these points. The subroutine *update* implements this and attempts to deal with the situation when the estimate itself is apparently far from reality but the data point might be better. To do this, maximum and minimum expected values of the estimate (f_{max} and f_{min}) are maintained for each latitude. These are formed as running estimates updated as the data is processed:

$$y > f_{max} \quad \rightarrow \quad f_{max} = \min(\, (\, 5 * f_{max} + y)/6 \, , \, \text{fmaxcons})$$

$$y < f_{min} \quad \rightarrow \quad f_{min} = \max(\, (\, 5 * f_{min} + y)/6 \, , \, \text{fmincons})$$

The updates of f_{max} and f_{min} are done only for the latitude closest to the input point no matter how large *nlatcon* is set. This is controlled by the 3rd to the last argument in *update*, the logical *newmnmx*. Setting $z^2 = (x_i - y)^2$, then set a measurement uncertainty, σ_{meas} as follows:

$$z^2 < \sigma_{rej}^2 \quad \rightarrow \quad \sigma_{meas}^2 = \sigma_d^2$$

$$z^2 > \sigma_{rej}^2 \begin{cases} x \bullet k > f_{max} \text{ or } x \bullet k < f_{min} \rightarrow \sigma_{meas}^2 = \sigma_d^2 + \sigma_{rej}^2 \\ f_{max} > x \bullet k > f_{min} \rightarrow \sigma_{meas}^2 = \sigma_d^2 + z^2 \end{cases}$$

The first case corresponds to the normal input point that is within expected parameters. The last case corresponds to a point outside of expected values, so we add an uncertainty equal to that of the distance of the point from the estimate. The middle case corresponds to a potentially bad point, but the estimate itself is outside of expected values, so we decrease the uncertainty of input points in this special case.

In an effort to achieve better behaved solutions in the presence of data voids, we apply a smoothing in latitude after each maptime. Since this smoothing is done after the estimate is sampled, it will have minimal effect on the solution so extracted unless there is little or no data. The smoothing is a 1, 2, 1 smoothing in adjacent latitudes for all interior latitudes and a -1, 2, 1 smoothing for latitude extremes:

$$x_i = \begin{cases} \tfrac{1}{2} x_1 + x_2 - \tfrac{1}{2} x_3 & i=1 \\ \tfrac{1}{4} x_{i-1} + \tfrac{1}{2} x_i - \tfrac{1}{4} x_{i+1} & \text{interior} \\ \tfrac{1}{2} x_N + x_{N-1} - \tfrac{1}{2} x_{N-2} & i=N_{lat} \end{cases}$$

In older versions of the mapper, we smoothed the consituents H_2O andNO_2 twice in succession. All other consituents were smoothed only once. In the present mapper, the smoothing number (*nsmooth*) is an input parameter that can be chosen at will. In older versions of the mapper, each coefficient of the estimate was smoothed

independently. In the present version of the mapper, we smooth the zonal mean separately, as before, but we smooth the amplitudes and phases separately. The amplitudes are smoothed using the same prescription as above for each coefficient. Since phase is ill-defined for weak amplitude waves, we form an average weighted by the respective wave amplitudes:

$$\theta_i = \frac{¼\theta_i\text{-}1r_{i-1} + ½\theta_i r_i + ¼\theta_i{+}1r_{i+1}}{r_{i-1} + r_i + r_{i+1}}$$

By explicitly smoothing wave phase this way, we ensure that phase is a smoothly varying quantity from latitude to latitude and is not unduly affected by regions of small wave amplitude. The routine *smthest* does all the smoothing in the program.

A method was used in older versions of the mapper to stabilize the solution in the presence of data gaps in longitude. If incomplete coverage was detected at each maptime, a set of so-called pseudopoints were generated from data at neighboring latitudes. The incompleteness of coverage was judged on the basis of counting data from the previous maptime in 12 30°longitude bins.If the maximum gap was only one bin (one consecutive empty bin), then waves 5 and 6 were damped (by a factor of .5 applied to the coefficient). If two consecutive bins were empty, then waves 3, 4, 5 and 6 were damped (with respective factors of .6, .5, .5, and .5). If 3, 4 or 5 adjacent bins were empty, then wave two was also damped (with a factor of .8). Finally, with more than 5 adjacent bins empty, wave one was also damped (with a factor of .9). In addition to damping the waves, the covariance was increased. The diagonal terms were increased by multiplying ds/dt by one day (a typical time difference between maptimes) and adding to the diagonal terms. The off-diagonal terms (again, only for waves that were damped) were multiplied by 1.2.

In addition to selectively damping the waves and adding to the covariance, pseudo points were added at five times the variance (σ^2) of the point at the central latitude. The value of this point was found from the estimate at neighboring latitudes evaluated at the central longitude of the empty bin. The neighboring latitudes were chosen such that their corresponding longitude bin was not empty.

The motivation of this process is clear. If longitude coverage is incomplete, planetary waves are ill-defined. The larger the gap in longitude, the more the data sparcity affects lower wave numbers. In practice, the procedure has proven unable to limit wave growth in data voids in the SBUV dataset. The process is keyed on the maptimes as the time to check the longitude grid and possibly apply damping and covariance adjustment. The remnants of this approach can be seen in the *lonobs* array of counts in the *mapdat* routine. A generalization of this technique has been investigated as part of the *apriori* update procedure discussed previously.

In the present method, the longitude grid is generalized to contain not data counts, but times of last data. We then can calculate (at either a new data point to update or a maptime for estimate extraction) a time of last data corresponding to longitude gap of a given size. We can then apply the time difference for any gap to the zonal mean, as before, and use larger gaps for the higher waves. We then use a particular Δt for a given wave both to determine damping and as a factor for ds/dt in increasing covariance. Tests of this procedure have shown little advantage over the adoption of a single Δt. For the HIRDLS mission in particular, there are few data gaps and no longitude-specific limitations to the coverage.

Code Overview

This L3 code delivery consists of two programs: an ingestion program (*h3ingw*) and the kalman filter program itself (*latseq*). The ingestion program reads through a series of HIRDLS L2 files, extracts the variable of interest (eg, Temperature) and writes out a data file in netcdf version 4 format. The kalman filter program reads in this netcf version 4 ingested data and eventually writes out an L3 coefficient file, also in netcdf version 4 format. Both programs are controlled by input data files. These inputs are described in the runtime options.

The ingest directive file contains the output file name, the input swath name (HIRDLS for this dataset), the variable of interest (eg, Temperature), and then a series of L2 file names. The program loops over these L2 filenames exiting when all have been processed. For each L2 file, it reads each day within the timeLoop extracting the appropriate constituent and writing it out. After all input files are read, the output file is put into redefinition mode and several global variables are updated such as start and stop times and ranges of the data.

The kalman filter main program is *latseq*. This file contains the overall loop from line 118 to 538 where successive namelists are read from the input file. Between line 118 and 196 where the namelist read occurs, several variables are set to their default values. The inner loop in this main program is the pressure loop running from line 492 to 517. For each pressure to map, the program *mapdat* is called (line 505).

The subroutine *mapdat* handles the mapping at a given pressure. The forward mapping occurs from line 410 to 644 with the forward time loop labeled *looptf* extends from line 450 to line 585. Similarly, the reverse mapping occurs from line 646 to line 799 with the reverse time loop labeled *looptr* extends from line 686 to line 796. Each of these time loops start by call *getnextf* (or respectively *getnextr*) to get the next data point and its associated time, latitude, longitude, uncertainty, solar zenith angle, and local time. The forward mapping routine *getnextf* calls the routine *rding*. It is here that data is selected on the basis of time window, node, scan, range, and other criteria. This subset of data is written into a direct access file (unit 10) on the first pass through the data in the forward direction, and then is read back in as needed.

The estimate is updated for *nlatcon* latitudes from a given input point. The forward latitude update loop is line 377 to 407 and the reverse latitude update loop is line 511 to 584. The fourier coefficients for the field are updated in the routine *update* and the scalar quantities (solar zenith angle and local time, if applicable) are updated in the routine *updates*.

The routine *fnlmap* (called on lines 704 and 826 of *mapdat*) combines the forward and reverse estimates at a given maptime, *kurmap*. The first 100 lines of *fnlmap* are devoted to wild point counting and optional plotting. Simple weighting by covariance ratio is used to combine the forward and reverse estimate (*xfwd* and *xrev*) and covariance (*sfwd* and *srev*). Notice that the forward quanties (*xfwd* and *sfwd*) are dimensioned ncoef by nlat by number of maptimes; however, the reverse quantities (*xrev* and *srev*) are only dimensioned ncoef by nlat. This is because after we finish a reverse maptime, the combined forward and reverse quantities are stored in the forward arrays. We then move on to the next maptime and can reuse the reverse variables. This consideration is also true for the scalar quantities that are optionally mapped at well (solar zenith angle, lines 118-127; and local solar time, lines 128-137). The rest of the routine *fnlmap* is devoted to more plotting.

It is also probably helpful to point out why *fnlmap* is called in two different locations within *mapdat*. The update process is data driven. As the program moves backwards through the data, a test is made to see if there are any maptimes that are later than the time of this data. The while loop at line 695-711 updates all the maptimes that have not already been updated since the previous test of this sort. With dense data, this will almost always be a single maptime. Once the data has been reviewed backwards in time, then a test is made for any maptimes

before the data. The loop between lines 821-827 updates these maptimes, all of which are outside of the data coverage.

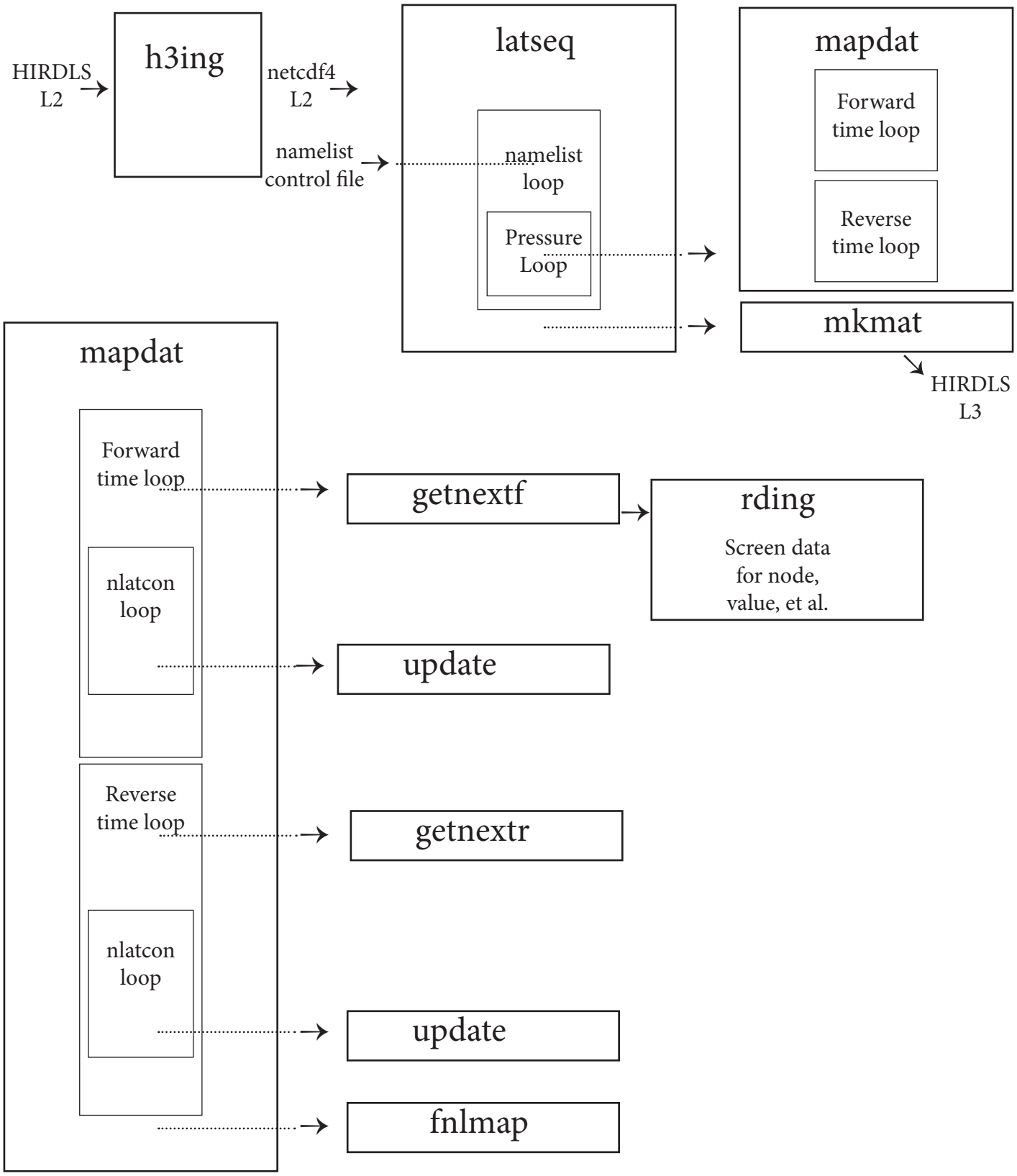
Going back to the top level program *latseq*, we see that the *mapdat* routine returns a logical true if the mapping is successful at that pressure. In that case, the routine *tpwrtc* saves the combined estimate for that pressure in a direct access file. Once all the pressures are processed, then *mkmat* is called (line 531) to create the final output file. The global attributes are used to store the runtime options such as the max/min data values allowed.

Runtime Options

The program is run via the command line thus: ./latseq inputfile.txt where the argument inputfile.txt is a namelist input file. The arguments are as follows:

nonegrms=true by default to exclude input marked with negative uncertainty (dominated by apriori)
xvol=filename for input, default value “ingout”
tmbeg,tdbeg,strtyr=month, day, year for start of map; default is beginning of data
tmend,tdend,stopyr=month, day, year for end of map; default is end of data
nodename=”Combined”(default), “Ascending”, “Descending”, “Day, “Night”
uarslat=.false.(default), set to .true. for -32→80 or -80→32 latitude range
numlat=number of latitudes for mapping grid, default is input grid
latmin,latmax=latitude grid to map, default is the input data grid
iprs=vector of length maxprs, if true, map that pressure. default is all true
nsymt=number of maps per day, defaults to one
synoptm=time offset of first map per day in hours, default is 12.
nwave=only set via compile time and equal to MAXWAVE
kplot=.true. for additional plots, default is .false.
plgrid=.true. for more data coverage plots, default is .false.
plthrm=.true. for additional plots, default is .false.
npltrms=number of map times to sum for rms diff plots, default is 6
meta=filename for output metacode, default is gmeta.cgm, set to /dev/null for no plots
diagout=filename for diagnostic output, default is diags.txt, set to /dev/null for no output
verbose=.true. for more diagnostic output
cofout=file for coefficient out, default is cofout.dat
comment=freeform text to write into the output, default is none
maxiter=number of iterations of forward/reverse mapping, default is 0, most mapping done with 1 or 3
nlatcon=number of nearest latitudes to update for each data point, default is 2
scan=HIRDLS specific screening for “up”, “down”, or “both”; default is “both”
fmincons=minimum value to accept for input data, default is no minimum
fmaxcons=maximum value to accept for input data, default is no maximum
sig2noise=minimum value of input value divided by noise, default is zero
DAYWIND=mapping window for the ds/dt calculation, default is 3
tau=constant vector (dimensioned nwaves+1) such that <S>/tau is ds/dt estimate, default is one.
sigadd=constant vector (dimensioned npress) to add to input noise, default is zero
sigfactor=constant vector (dimensioned npress) to multiply input noise, default is one
fieldfac=constant to scale field and noise internally to avoid numerical inaccuracy, default one
nsmooth=number of times to smooth in latitude after maptimes, default is one
szamin=for “night” node, exclude solar zenith angles less than this, default is 100
szamax=for “day” node, exclude solar zenith angles greater than this, default is 90

Other namelist entries have only been used for testing, not operational use of the mapper.


HIRDLS
L2
h3ing
netcdf4
L2
namelist
control file
latseq
namelist
loop
Pressure
Loop
mapdat
Forward
time loop
Reverse
time loop
mkmat
HIRDLS
L3
mapdat
Forward
time loop
nlatcon
loop
Reverse
time loop
nlatcon
loop
getnextf
rding
Screen data
for node,
value, et al.
update
getnextr
update
fnlmap

References

Clive D. Rodgers (1976). “Retrieval of Atmospheric Temperature and Composition From Remote Measurements of Thermal Radiation”. Reviews of Geophysics and Space Physics 14 (4). p. 609.

Fetzer, E. and J. Gille (1994), Gravity wave variance in LIMS temperatures. Part I: Variability and comparison with background winds. J. Atmos. Sci. 51, 2461-2483.

Wright, C.J., and J.C. Gille (2013), Detecting overlapping gravity waves using the S-Transform, Geophys. Res. Lett., 40, doi:10.1002/grl.50378