How To Process LADCP Data With the LDEO Software

(last updated for version IX.5)

A.M. Thurnherr

July 9, 2008

1 Introductory Remarks

Processing data from Lowered Acoustic Doppler Current Profilers (LADCPs) is not trivial. There are currently (at least) three publicly available software packages that can be used for this purpose:

1. an implementation of the shear-method, primarily written by Eric Firing at University of Hawaii
2. an implementation of an inverse technique developed by Martin Visbeck, who wrote it while working at LDEO (Columbia University)
3. a re-implementation of the inverse method by Gerd Krahman at IfM/Geomar in Kiel

The purpose of this "how-to" document is to provide a cookbook-like introduction to using the LDEO software — it is assumed that the reader knows what LADCP processing is all about and knows how to acquire LADCP data.

Over the years many different versions of the LDEO LADCP processing software have been made available. This document was first written after I essentially re-implemented the "user interface" of the then-current LDEO software version (8b) during the CLIVAR P02 cruise in summer 2004. In order to avoid confusion this version with the new interface is called "Version IX", available from `ftp://ftp.ldeo.columbia.edu/pub/LADCP`. (Gerd Krahman's re-implementation of the inversion technique is newer than Version IX and is called V.10.) Version IX has many different sub-versions — see subtitle above to determine which sub-version this document corresponds to. The changes between the different sub-versions are listed in the `HISTORY` file, which is part of the software distribution.

It is assumed that the reader of this document is familiar with Matlab and the underlying operating system (usually a UNIX or Windows variant). Since I do not use Windows at all, there may be Windows-specific bugs — see appendix B for instructions on how to report bugs.

2 Software and Data Requirements

2.1 Software Requirements

Matlab ***(required).*** Both student and full versions should work. At the time of writing, no toolboxes are required except for calculating diffusivities (which is not covered in this document; section 3). I use Matlab versions as old as 6.5 (aka. Release 13), because this is easy to install under FreeBSD.

LDEO LADCP Processing Software Version IX ***(required).*** See document subtitle to find out which sub-version this "how-to" refers to.[1] The software consists of set of Matlab scripts and functions. The Matlab path must be set up to find the scripts and functions.

`geomag60` **Program** ***(semi-optional).*** Prior to Version IX.5, the magnetic declination[2] was calculated using the Matlab script (incorrectly[3]) called `magdev.m`. Since the International Geomagnetic Reference Field (IGRF) is updated every 5 years, `magdev.m` had to be updated as well. As of Version IX.5, magnetic declination is calculated using the external program `geomag60`, which can be downloaded from a NOAA server. (See comments in function `geomag` in file `loadnav.m` for details on how to obtain & install `geomag60`.) The `geomag60` program reads the IGRF from a file of coefficients, i.e. it is anticipated that only the coefficients file needs to be replaced when the IGRF is updated.

If `geomag60` is not installed correctly, the magnetic declination is calculated using the IGRF updated in 2000 and a corresponding warning is produced. In many regions, the value derived from the old IGRF will be an acceptable approximation to the true declination, but this is not the case everywhere. *Therefore, it is strongly recommended that `geomag60` be installed. Alternatively, the magnetic declination can be set manually for each cast (see section 3.2 below for details).*

`scan*` **Program** ***(optional).*** The Hawaii shear-method implementation, which is not covered by this "how-to", contains a utility that can be used to check the integrity of LADCP data files. The name of the utility depends on the type of instrument used but it always starts with `scan` — e.g. in case of RDI Workhorse data the program is called `scanbb`. See appendix D for an example.

Matlab `sw` Library ***(optional).*** If available, the `sw` routines are used to calculate buoyancy frequency and sound speed from CTD time-series data. As detailed in section 5 below there is little reason to do so in the current version. (Buoyancy frequency is not used and a sound-speed routine is included in the LDEO software.)

Matlab `netcdf` Interface ***(optional).*** If this interface is installed correctly `netcdf` files should automatically be created (one per cast). As I don't use this feature, I do not know whether this is the case in the current version.

2.2 Data Requirements

LADCP Raw Data ***(required).*** One file per cast per LADCP head.

Magnetic Declination ***(required).*** One value per station. See also comments in sections 2.1 and 3.2.

CTD Time Series ***(optional).*** For each cast there should be a file with CTD time-series data with a time resolution of order 1 s. It must contain time, pressure, *in-situ* temperature and salinity fields. While CTD time-series data are not required in order to process LADCP casts, the resulting profiles will not be of very high quality, because neither accurate depth- nor sound-speed data are available.

[1] This "how-to" does not apply to Gerd Krahman's V.10.
[2] Magnetic declination is the difference between magnetic and true bearings.
[3] Magnetic deviation is the difference between magnetic and compass bearings.

GPS Time Series *(optional).* For each cast there should be a file with GPS time-series data, containing time, latitude and longitude information. The time fields in the GPS and CTD time series must be synchronized. Therefore, it is recommended that the CTD and GPS time series of a given cast are in the same file. While GPS time-series data are not required in order to process LADCP casts, the resulting profiles will not be of very high quality, because GPS data provide by far the most accurate constraint for the barotropic velocities.

Shipboard ADCP (SADCP) Data *(optional).* There should be a single Matlab file containing a time series of SADCP profiles. The availability of SADCP data improves LADCP profiles, in particular for very deep casts or in regions with relatively few scatterers in the water column.

CTD Profile *(optional).* For each cast there may be a file containing the corresponding CTD profile. If these data are not provided, a profile is calculated from the CTD time-series data, which does not generally degrade the LADCP profile at all.

3 Single-Head LADCP Processing Without Auxiliary Data

In order to minimize complexity, LADCP data from a single head should be processed first, without using any auxiliary data. In case of a dual-head setup, this step should be carried out independently for the data from the two heads. In case of processing problems (or doubts) it is always a good idea to revert to this simplest possible case. Once processing of individual LADCP files works, the data from dual heads can be combined (section 4) and auxiliary data can be included in the processing (sections 5–8).

3.1 Creating the Cruise-Specific Directory Structure

The LDEO software does not have any specific requirements as to the directory structure. However, basic processing involves different types of files and it is recommended that each type resides in its own directory:

Processing Parameters. Suggested subdirectory name: `proc`;

LADCP Data Files. Suggested subdirectory name: `raw`;

Processed LADCP Files. Suggested subdirectory name: `processed`;

Checkpointing Files. The processing software can be set up for checkpointing to increase processing speed when processing has to be partially repeated, e.g. with different parameters (see appendix C for details). While checkpointing is by default disabled, one (small) checkpoint file is nevertheless written for every cast as a sentinel. Suggested directory name: `checkpoints`.

3.2 Setting Cruise- and Cast-Specific Parameters (`proc/set_cast_params.m`)

In order for processing to work correctly, a number of Matlab structures have to be initialized. This is done in the file `set_cast_params.m`, which usually resides in the directory from which processing is done. *Note that there is intentionally no file `set_cast_params.m` in the software distribution; a realistic example is shown in appendix A.* The script `set_cast_params.m` is called (twice!) from `process_cast.m`. The following structures are typically modified in `set_cast_params.m`:

`f` contains file- and path names;

`p` contains parameter used for reading and preparing the data files;

`ps` contains parameters used for calculating the absolute velocities;

`pk` contains parameters used for calculating diffusivities.

There are many different parameters that can be set. By design, all of them should be documented in file `default.m`, where their defaults are set. However, this is not necessarily the case, i.e. quite a few parameters are found in other source files. The only way of finding all parameters that can be changed, as well as their default values, is to search all Matlab source files for calls to the function `setdefv`, which is used to set the defaults. This should normally not be required as the commonly changed parameters are included in `default.m`. There are two things that are accomplished in `set_cast_params.m`:

1. Parameters without defaults (cruise id, path names, &c) are set.
2. Parameters with defaults are set to non-default values. This is optional but recommended for some parameters (see below).

`set_cast_params.m` can either be created from scratch or it can be adapted from another cruise. I recommend creating the file from scratch for each cruise, because this will prevent any parameters that were set for a particular cruise to be applied to another cruise for which they may not be appropriate. As an extreme case, it may be possible that the software has changed and the parameter does not even exist any more, in which case setting it will be a source of potentially endless confusion. (When creating a new `set_cast_params.m` from scratch, make sure that there are no old version lying around in the Matlab path.)

Within `set_cast_params.m` all variables from `process_cast.m` are available. (Care has to be taken not to overwrite the variables that are used internally by `process_cast.m`.) In particular, the variable `stn` contains the station number (1st parameter to `process_cast.m`).

Assigning values to the following parameters is *mandatory*:

`f.checkpoints = sprintf('checkpoints/%03d',stn);` This sets the basename of the checkpoint files (see appendix C for details). In the example, the checkpoint files will reside in the `checkpoints` subdirectory of the current directory (which must exist and be writable).

Important information for Windows users: Forward slashes (/), rather than backslashes (\) must be used as path separators for all parameters involving path names.

`f.res = sprintf('processed/%03d',stn);` This sets the basename of the output files (including log files and plots). In the example, the processed files will be written in the `processed` subdirectory of the current directory (which must exist and be writable). If `f.res` is not set, processing still works but the output is not saved, which not particularly useful.

`f.ladcpdo = sprintf('raw/%03dDN000.000',stn);` This sets the filename of the raw LADCP data file. In the example, the raw files are read from the subdirectory `raw`. Either a downlooker or an uplooker data file can be specified; if an uplooker data file is specified there is no information that can be used to determine the barotropic velocity constraint and only the baroclinic solutions are calculated.

`p.drot = 110;` This sets the magnetic declination, which must be obtained externally. One way of obtaining an estimate for the magnetic declination is by calling `magdev`, e.g. `p.drot = magdev(-72,170);` This uses the (approximate) cast location (72°S, 170°E in the example)

and an old version of the IGRF. Note that the declinations returned by `magdev` may be poor approximations for LADCP casts collected in some regions of the ocean after the year 2000 (section 2.2).

An alternative to setting `p.drot` consists in setting the cast start- and end positions `p.poss` and `p.pose` (see `default.m` for additional information). If these parameters are set, the processing software will use `magdev` to automatically calculate the value of `p.drot`. Setting the start and end positions will (by default) add a constraint for the barotropic flow and it is therefore important that accurate values are used.

For final processing it is always preferable to use GPS data (section 6), which obviates the need to set `p.poss`, `p.pose` or `p.drot` and calculates the correct magnetic declination, if `geomag60` is installed correctly (section 2.1).

At this stage it is possible to calculate absolute velocities by running `process_cast(<station number>)`. It is recommended, however, that the following optional statements are included in `set_cast_params.m` before processing:

`more off;` It is annoying when batch-processing stops because the screen is full.

`p.cruise_id = 'Anslope II'; p.ladcp_station = stn; p.whoami = 'A. Thurnherr';` This defines the cruise ID, station number and name of the person who processed the data. While these parameters are not used during LADCP processing they are stored in the output files and used in some auxiliary LDEO scripts that generate `html` tables and for archiving the data;

`p.name = sprintf('%s # %d downlooker',p.cruise_id,p.ladcp_station);` The station name is used as a plot title. In the example, the plots are marked with the fact that only the downlooker data are processed.

`clear pk;` This statement disables the calculation of diffusivity profiles. Since the current diffusivity code has not yet been tested, it is recommended that diffusivities are not being calculated. (Not calculating the diffusivities has the added advantage that no Matlab toolboxes are required.)

`p.saveplot = [1:4 11 13:14];` By default, only ProcFig. 1 is saved as a Postscript file in the directory specified in `f.res`. (In order to differentiate processing figure numbers from the figure numbers of this document, the processing figures are called ProcFigs. See appendix F for the list of ProcFig. numbers and their meaning.) In order to detect potential processing problems, other figures should be saved as well. In the example, all figures that make sense for a single-headed system are saved. (ProcFig. 12 is not generated because it currently generates an error when processing is carried out without auxiliary data.) Note: ProcFig. 4 will only be generated if CTD time-series data are included in processing (section 5).

`ps.smallfac = [1 0];` By default an inversion constraint is enabled that prefers low-mode solutions. While this may be a useful constraint in case of the runaway shear due to bad instrument range I recommend to disable this constraint under most circumstances.

`p.btrk_ts = <target strength>;` This parameter, which defaults to 10, is used during processing to differentiate true from false bottom returns. I had to increase this parameter to 30 during a cruise where 5 acoustic instruments were pinging without synchronization during CTD/LADCP casts, because instrument interference was misinterpreted as bottom returns. I recommend to play around with this parameter if the seabed is not detected correctly (as seen in ProcFig. 4; appendix F).

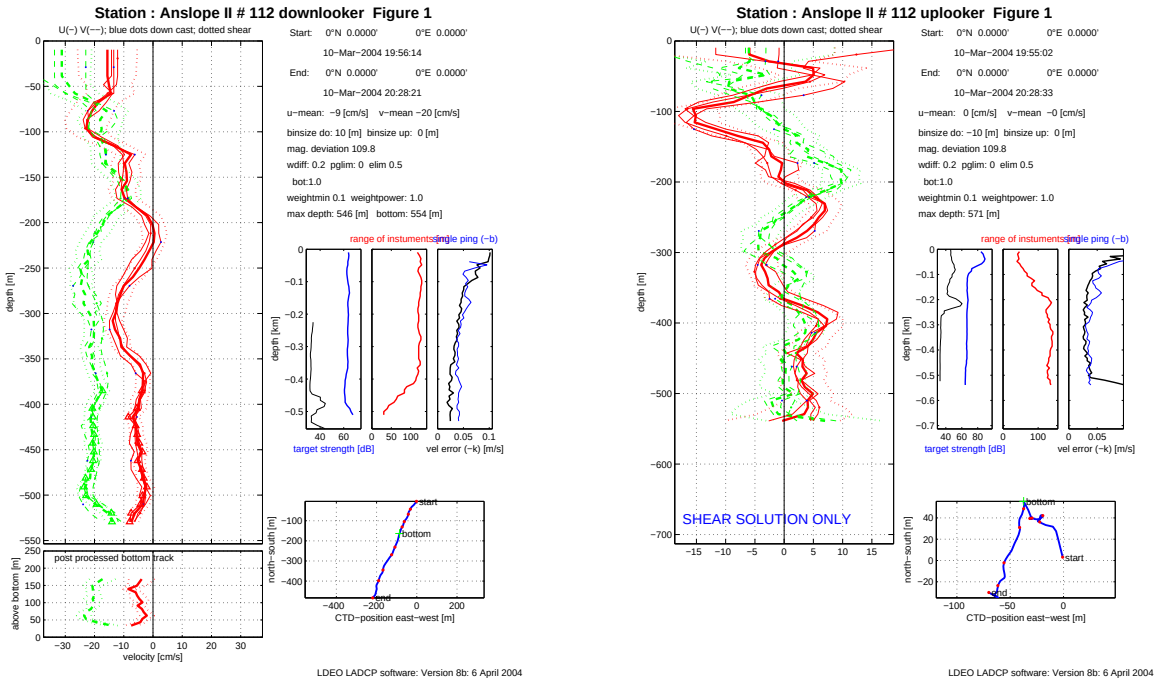


Figure 1: Separately processed velocity profiles (ProcFig. 1) from a dual-headed cast processed without any auxiliary data. The downlooker data (left panel) are bottom-track referenced. The uplooker data (right panel) are not referenced, i.e. only the baroclinic profile is shown.

`f.IGRF = '<IGRF path>';` This parameter, which defaults to `/usr/local/lib/IGRF10.cof`, sets the location of the current IGRF coefficients (section 2.2). If this is set incorrectly, an old IGRF model is used and the magnetic declinations may be in error.

Data-Editing Parameters. Depending on the instrument setup and configuration (including those of other acoustic instruments) it may be necessary to change the values of various data-editing parameters from their defaults; see appendix E for more details.

At this stage, the LADCP data can be processed with `process_cast(<station number>)`. On doing so, a lot of diagnostic output is produced. After processing, this output can be found in a `*.log` file in the `processed` subdirectory. An example of the result of processing the LADCP files from the up- and downlookers of one particular cast is shown in Figure 1. The barotropic component will be determined from bottom-track data if data from a downlooker are used and if the instrument was in range of the seabed for a sufficient amount of time. (It is possible that no bottom-track data are available because the bottom stop was too close to the sea bed and there were bottom-detection problems during the down- and the up-casts.) If bottom tracking failed the profiles shown in ProcFig. 1 are annotated with `SHEAR SOLUTION ONLY` (which is a lie — what is meant is `BAROCLINIC SOLUTION ONLY`).

At this stage the data quality can be assessed by checking the various processing figures for anomalies (appendix F), by intercomparing the solutions from the two heads (noting that the uplooker profiles are not referenced, i.e. only the shear can be compared) and by comparing the top part of the profiles to the SADCP data from the same cast.

3.3 Processing 3-Beam Data

With every ping ADCPs collect separate velocity components along each of the four beams ("beam coordinates"). For processing, these along-beam velocities are transformed into an Earth-based coordinate system (u, v, w). Since only 3 of the beam velocities are required for this transformation, the fourth provides redundant information that is used to assess the horizontal homogeneity of the flow ("error velocity"). When only 3 beam velocities are available, e.g. because of a beam with reduced range, it is still possible to calculate Earth-coordinate velocities. If the instrument is programmed to record the data in Earth coordinates the transformation is carried out automatically by the instrument.

It is recommended, however, that the instrument be programmed to record single-ping ensembles in beam coordinates.[4] With this choice, the coordinate transformation is handled by the LDEO processing software. As of Version IX_5, this transformation includes 3-beam solutions, unless this is explicitly disallowed by setting `p.allow_3beam_solutions = 0;`. Additionally, it is possible to force the software to disregard all velocity data from a particular beam by setting `p.ignore_beam` to the beam number to be ignored. This is useful, for example, in case of a beam returning biased velocities.

3.4 Processing Without Figure Windows

During processing, many figure windows are opened and updated. This can be annoying when the computer is used for other tasks during processing. In order to process LADCP data without figure windows on UNIX machines, Matlab can be started with the `-nodisplay` option, but the generation of PNG figure files must also be suppressed (`p.saveplot_png = [];`)

With some Matlab versions and operating systems this will not produce correct Postscript figures, however. As a workaround, the following code can be added to `set_cast_params.m`: `close all; figure(100);` Note, however, that even with this workaround wrong Postscript figures can be produced for the first cast that is processed after Matlab is started. Under these circumstances, the first cast should be processed twice to ensure good Postscript figures.

4 Combining LADCP Data From 2 Heads

When two LADCP heads are mounted on a single platform the data from the instruments can be combined. This requires the addition of a single statement in `set_cast_params.m`:

```
f.ladcpup = sprintf('raw/%03dUP000.000',stn);
```

Of course, it is also advisable to adjust `p.name` to reflect the fact that combined data are used. Furthermore, ProcFigs. 5, 6 and 10 can now be added to `p.saveplot` to assess the consistency between the data from the two heads. In the example shown in Figure 2 the up- and down-looker data from Figure 1 were combined for processing. Note in particular the horizontal banding in the two center panels, which indicates consistency between the data from the two instruments.

Merging of the data from the two instruments is implemented in a fairly simple fashion in `loadrdi.m`. In order to function properly there are a few assumptions that have to be (approximately) satisfied — failure to do so will generate warnings (which can be found after processing in the `*.log` files). Otherwise, it is not clear to me what the consequences are. The assumptions are:

Constant Pinging Rate.

[4] Because of instrument motion, beam coordinates must not be used for multi-ping ensembles!

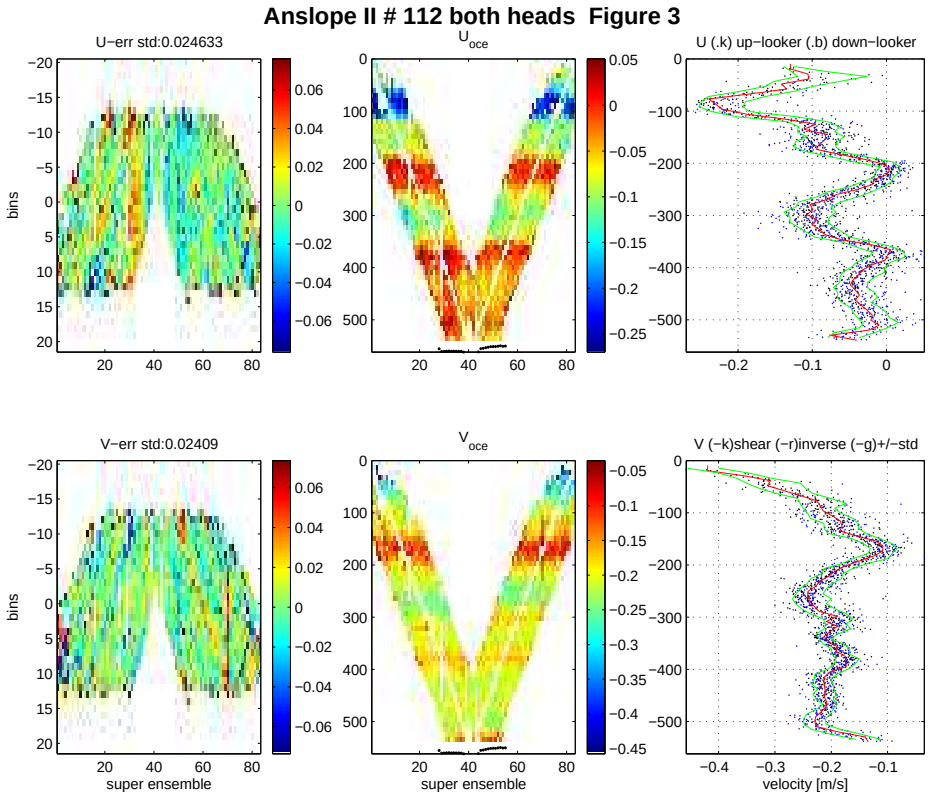


Figure 2: ProcFig. 3 from a dual-headed cast processed without any auxiliary data.

Equal Pinging Rate in Both Instruments.

Synchronized Clocks. The merging code is able to cope with a modest clock offset (currently hardcoded to a maximum of 20 ensembles).

5 Adding CTD Time-Series Data

Perhaps the main problem with processing LADCP profiles without auxiliary data (sections 3 and 4) is the lack of direct measurements of depth, which must therefore be estimated by integrating the measured vertical velocities. The next logical step therefore consists in adding CTD time-series data. Since CTD time-series data are usually acquired specifically for LADCP processing the LADCP processor can usually chose the data and format to be recorded. Recommendations:

File Format. There are, of course, an infinite number of possible CTD file formats. Earlier versions of the LDEO software included several different CTD loading routines (e.g. `loadctd_whoi.m`). Unfortunately, `loadctd.m` does not only load the CTD data but also merges them with the LADCP time series. The merging code is rather sensitive and has been modified a lot over the years, implying that most of the `loadctd.m` functions were constantly out of date and either did not work at all or worked poorly. Therefore, it it was decided to provide only a single

`loadctd.m` that is generic enough to load any CTD ASCII file. The file format of the CTD file is supplied by a number of parameters that should be set in `set_cast_params.m` (see below).

Time Base. The current `loadctd.m` routine allows the time of CTD time series to be specified in three different ways:

Elapsed. (Number of seconds since the CTD was turned on.) This is the most convenient time base since it does not require the CTD clock to be set correctly. Matching of the CTD and LADCP data is accomplished by correlating the CTD pressure time series with the LADCP depth (from integrated w) time series.

Year Day. (Decimal days since the beginning of the current year, starting with 1.0 at midnight of January 1st.) This is perhaps the most oceanographically common way of specifying time.

Gregorian. (Decimal days.) This is the time base used in Martin Visbeck's earlier versions of this software. While it provides an absolute, unambiguous standard it is numerically suboptimal because subsequent times consist of large numbers differing by a small fraction. I am aware of at least one case where the precision used during CTD processing was inadequate which led to significant jitter in the CTD time series.

In my view there is confusion as to what is meant by Gregorian and Julian times. In oceanography year days are often called Julian, which is pretty bad as it has nothing to do with the shift from the Julian to Gregorian calendars. To make matters worse, the LDEO software uses a non-standard definition of Gregorian time with days starting at midnight rather than at noon (see `gregoria.m`).

Temporal Resolution. The temporal resolution of the CTD time series should be at least as high as the LADCP ensemble rate. A higher rate will improve the accuracy but increase the file sizes and processing time. I generally use 1 s averaged CTD data.

Clock Synchronization. `loadctd.m` is quite good at merging CTD to LADCP data. (The results are reported in the `*.log` files and shown on ProcFig. 4 — section F — and should be checked for each cast.) The merging works best if the LADCP and CTD clocks are approximately synchronized (see `p.ctdmaxlag` in `default.m` for current default limits).

While the CTD/LADCP clock synchronization is not particularly sensitive it is vitally important that the CTD and GPS times are synchronized accurately. It is therefore recommended that GPS time series are merged with CTD time series during acquisition; the easiest way to ensure this is by combining the CTD and GPS time series into a single file. For standard SeaBird systems this can usually be accomplished by providing the deckbox with a serial GPS feed.

In order to merge the CTD time series with the LADCP data, the CTD file name and -layout are defined in `set_cast_params.m` as follows:

`f.ctd = sprintf('CTD/%03d.CTD+GPS',stn);` In this example, the CTD and GPS time series are merged in the same file (in the `CTD` subdirectory) to ensure accurate clock synchronization.

`f.ctd_header_lines = 115;` Number of header lines (before the CTD data start) in the CTD files.

`f.ctd_fields_per_line = 20;` Number of data fields per CTD scan (record).

`f.ctd_pressure_field = 2;` Field number of pressure field (left-most field has number 1).

`f.ctd_temperature_field = 3;` Field number of *in-situ* temperature field.

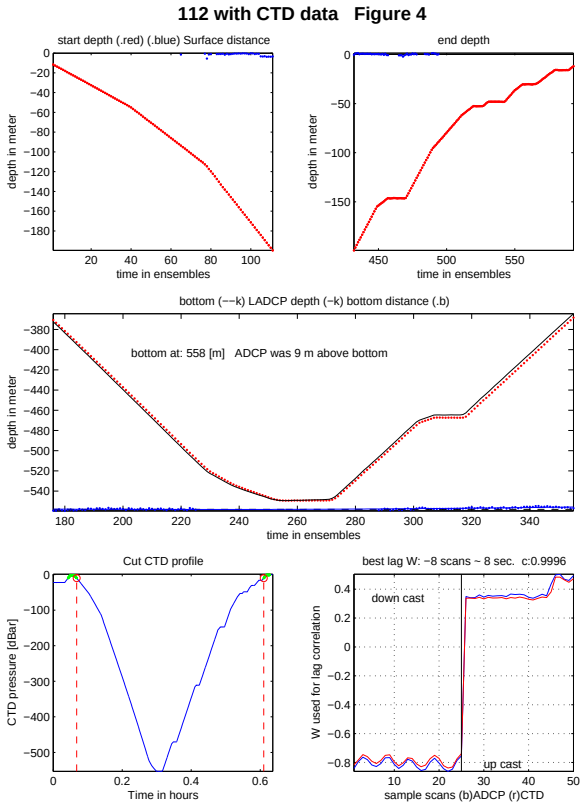


Figure 3: ProcFig. 4 from a dual-headed cast processed with CTD time-series data.

`f.ctd_salinity_field = 16;` Field number of salinity field.

`f.ctd_time_field = 14;` Field number of time field.

`f.ctd_time_base = 0;` Time base to use. 0 is elapsed; 1 is year-day; 2 is Gregorian. Strictly speaking this statement is superfluous, because `f.ctd_time_base` defaults to 0.

In order to calculate depth from pressure accurately the latitude of the cast must be known. The cast latitude can be specified either by setting `p.poss` and `p.pose` in `set_cast_params.m` (see comments in section 3) or by including GPS data (section 6). Figure 3 shows an example of ProcFig. 4, which is only available if CTD data are included. The quality of the data merging can be assessed from the center and the bottom-right panels — if there are apparent horizontal offsets between the CTD and LADCP curves in either panel there is a problem that has to be solved before proceeding.

If the Matlab `sw` routines are available at this stage (check using `which sw_svel`) sound speed and buoyancy frequency are calculated in `loadctd.m`. Since buoyancy frequency is only used to calculate diffusivities (currently not recommended; section 3) and because the LDEO software contains a routine to calculate sound speed the `sw` routines are not really required.

6 Adding GPS Data

Traditionally, GPS information at the start and end of a LADCP cast is used to estimate the barotropic component of the flow. As seen above, the geographical location of a station is also used to calculate magnetic declination (section 3) and to calculate depth from pressure (section 5). In the LDEO software, a full time series of GPS data can furthermore be used to constrain the CTD platform motion using a drag model (currently experimental code). Therefore, it is recommended that a GPS time series at the same temporal resolution of the CTD time series is collected for each cast. Merging of the GPS data is accomplished in `loadnav.m`. The recommendations regarding file format &c are analogous to those of the CTD time series (section 5):

File Format. The function `loadnav.m` provided with the current version of the software handles ASCII files; adaptation to a particular file format requires setting 6 variables: number of header lines in each file (`f.nav_header_lines`), number of fields per scan (`f.nav_fields_per_line`), field numbers (starting at 1) for time (`f.nav_time_field`), latitude (`f.nav_lat_field`) and longitude (`f.nav_lon_field`), as well as time base (`f.nav_time_base`).

Time Base. The current `loadnav.m` routine allows the time of GPS time series to be specified as year days, as Gregorian days, or as elapsed time (section 5). *Do not use elapsed time as a time base for GPS data unless the GPS data are co-recorded with the CTD data, in which case it is recommended.*

Temporal Resolution. For future use of the drag model it is recommended that the temporal resolution of the GPS time series should be at least as high as the LADCP ensemble rate. A higher rate will improve the accuracy but increase the file sizes and processing time. If the drag model is not used, it may be OK to use a lower-resolution of the GPS time series but I have not tested this.

Clock Synchronization. The only way the GPS time series can be merged with the LADCP data is to synchronize the time fields. If CTD data are included in the processing (highly recommended; section 5) the CTD clock is used as the absolute standard (the LADCP times are adjusted in `loadctd.m`) and the CTD and GPS times must agree. If no CTD time series is available, the LADCP and GPS clocks must agree. *By far the best way to ensure correct time referencing for the GPS data is to include GPS data in the CTD data stream.*

Merging of the GPS time series with the LADCP data is accomplished in `loadnav.m`. In order to enable CTD time-series merging something like the following statement should be added to `set_cast_params.m`:

`f.nav = f.ctd;` In this example, the CTD and GPS time series are merged in the same file to ensure accurate clock synchronization (section 5). If the GPS data reside in a different file, the pathname must be specified in `f.nav`.

Once GPS time-series data are included in processing it is important that the magnetic declination (`p.drot`, or `p.poss` and `p.pose`) are not set manually (section 3). (GPS data overrides the manually set values of `p.poss` and `p.pose` for calculation of the barotropic flow component, but magnetic declination is calculated before the GPS data are read!)

Once GPS information is used for processing the shiptrack is added to the bottom right panel of ProcFig. 1 (Figure 4). Note that even without GPS information (left panel) the processing software is quite capable of estimating the ship drift while on station. This implies that referencing the LADCP profiles using bottom-track data can work quite well.

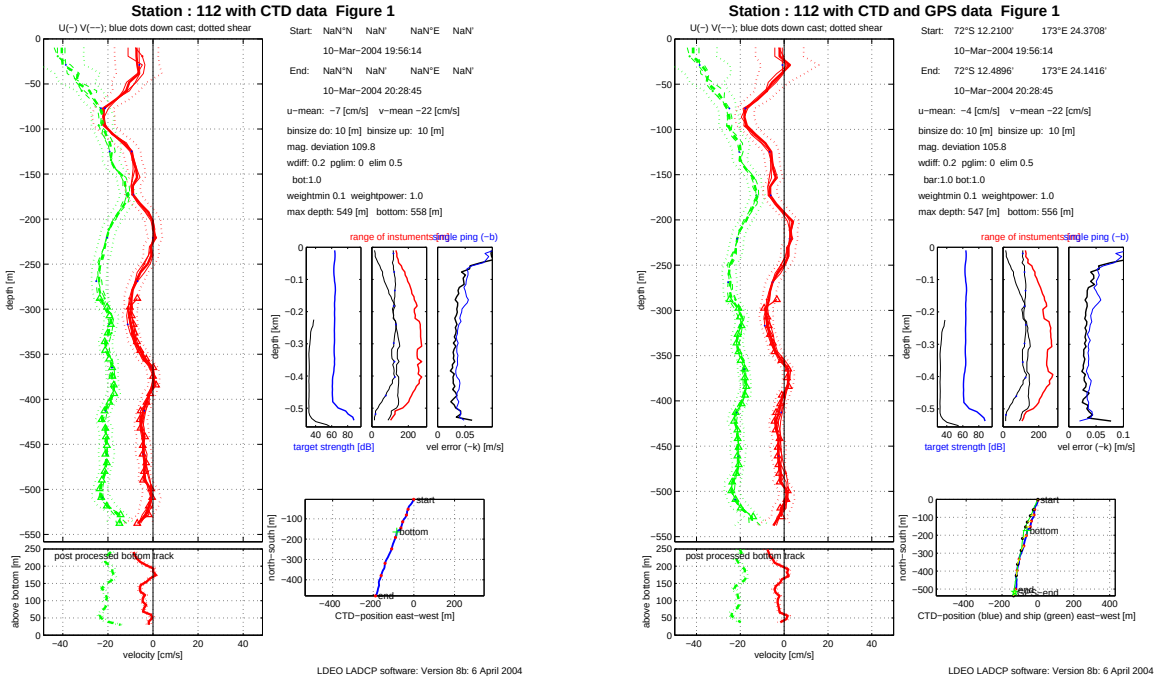


Figure 4: ProcFigs. 1 from a cast processed (left) without and (right) with GPS data.

7 Adding Shipboard ADCP Data

If the shear in a LADCP profile were known perfectly, a single constraint for the barotropic flow would be sufficient. Either bottom-track or GPS data would do in this case. (In the LDEO software the shear is not calculated explicitly. Nevertheless, the system of linear equation resulting from using water-track LADCP data alone is singular and an arbitrary constant can be added to the solution, i.e. water-track LADCP data only constrain the baroclinic part of the flow field.) However, since the shear is associated with uncertainties it is advisable to use measurements that constrain the velocity profiles at different depths. The bottom-tracking data provides an estimate for the velocity near the sea bed. The GPS data provides an estimate for the depth-integrated velocity which, in the presence of shear bias, constrains the velocity near the middle of the cast. Near the sea surface, shipboard ADCP (SADCP) data can be used to constrain the velocities.

The present version of the software LDEO requires time series of SADCP profiles, which are merged with the LADCP data in `loadsadcp.m`:

File Format. In the current version, `loadsadcp.m` reads a Matlab file containing several matrices (see comments in the source for a description). Time, latitude and longitude meta-data are required for each SADCP profile, which consists of depth, u and v values.

On many US ships, SADCP acquisition and on-board processing is carried out by UH software, which outputs SADCP profiles in Matlab files called `contour_xy.mat` and `contour_uv.mat` in a directory named `contour`. If output from the UH software is available the pre-processing script `mkSADCP.m`, which is part of the LDEO software, can be called. `mkSADCP` takes two arguments: the pathname where `contour_xy.mat` and `contour_uv.mat` are found and the filename of the output file.

During shipboard LADCP processing `mkSADCP` should be called from `set_cast_params.m` in order to update the SADCP matrix with the most up-to-date data. After the cruise, this call is no longer necessary because the SADCP data don't change any more.

Latitude/Longitude Data. As far as I am aware, latitude and longitude data are used only for consistency checking with the GPS time series.

Temporal Resolution. Typically, SADCP profiles are averaged over several minutes by the acquisition system; these averaged profiles are perfectly adequate for use in LADCP processing.

Clock Synchronization and Time Base. Currently, Gregorian time must be used. Time is only used to extract the SADCP data between the beginning and the end of the cast. Therefore, the clock must be synchronized with the CTD clock (or with the LADCP clock, if CTD data are not used; section 5). However, the synchronization does not have to be particularly accurate — differences of a couple of minutes or so are generally fine.

SADCP Calibration. Underway SADCP data have to be very carefully calibrated in order to remove the ship's velocity which is typically much greater than the ocean velocity. Since only on-station SADCP data are used during LADCP processing, even somewhat suboptimal SADCP calibrations are OK.

Temporal Variability. Currently, the ocean velocity is treated as steady over the time scale of the cast, i.e. no attempt is made to fit different SADCP profiles to the beginning of the downcast and the end of the upcast.

In order to include SADCP data the following additions to `set_cast_params.m` are required:

`f.sadcp = 'SADCP/Anslope.mat';` This parameter describes the path of the SADCP Matlab file.

`p.sadcp_dtok = 0;` This parameter, which extends the SADCP time window to an interval before the beginning of the cast and after the end of the cast, is by default set to 5 minutes. On a 'snappy' cruise, this may mean that underway SADCP data are used, which requires very high SADCP calibration quality. Except for very short casts, where not enough SADCP data available it is recommended that this parameter be set to zero.

When, additionally, ProcFig. 9 is added to `p.saveplot` SADCP information is plotted as shown in Figure 5.

8 Adding CTD Profile Data

CTD time-series (section 5) are usually generated from raw data, i.e. before spikes and other glitches are removed. The LDEO software allows inclusion of fully processed CTD profiles to replace the profiles constructed (during LADCP processing) from the time series. Since the CTD time series are still needed for calculation of the depth of the instrument package and since I never had a problem using profiles constructed from the time series, I do not use this feature and `loadctdprof.m` is not well integrated into version IX. If you want to use it, the following recommendations apply:

File Format. `loadctdprof.m` reads ASCII files. In the current version (which is severely suboptimal because of the way the data files are parsed) no headers are allowed (although header-handling code can be found commented out in the `readctd()` subroutine). `loadctdprof.m` must be adapted to the file format of the CTD profile files by editing the line where the default of `ipos` is set. `ipos` needs to be set to the field numbers (1-relative) of the pressure, *in-situ* temperature, and salinity fields.

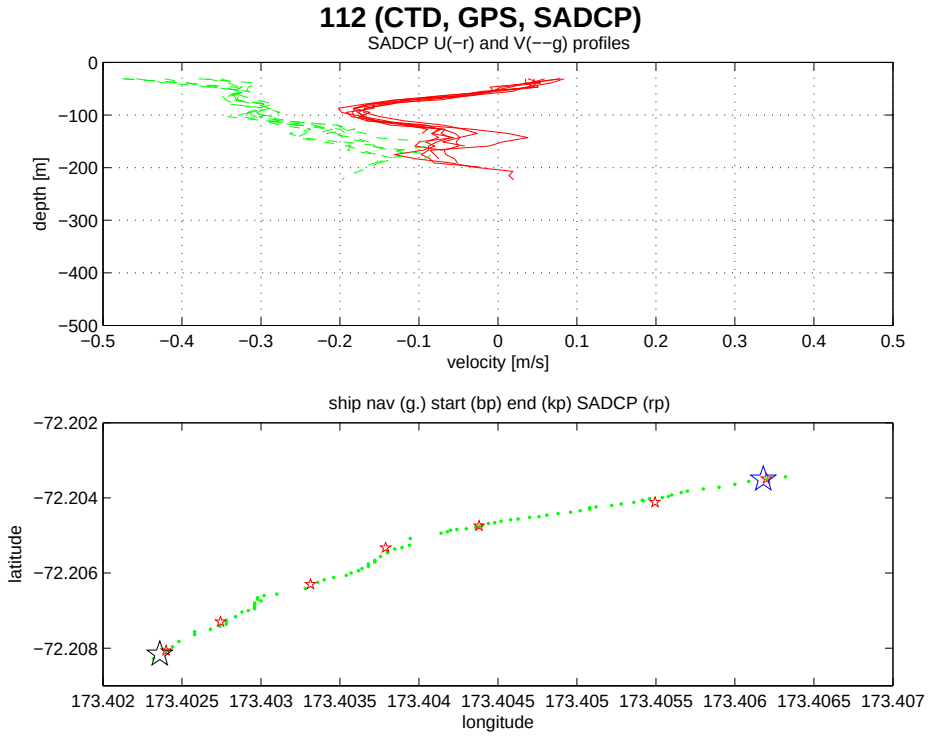


Figure 5: ProcFig. 9 showing SADCP data. Upper panel: Ensemble of SADCP velocity profiles in time-range of cast. Lower panel: Shiptrack from GPS (blue stars and green dots) and location of SADCP profiles (red stars).

Depth Resolution. Typical CTD profile resolutions of 1 or 2 dbar are adequate.

In order to enable inclusion of CTD Profiles, something like the following statement must be added to `set_cast_params.m`:

```
f.ctdprof = sprintf('CTD/%03d.prof',stn);
```

9 Post-Processing LADCP Profiles

LADCP profiles can be post-processed by providing a `post_process_cast.m` Matlab script, usually in the same directory where `set_cast_params.m` is found. For example, it is possible to copy all data to a backup directory on a different disk drive after processing. If `post_process_cast.m` is accessible in the current Matlab path, it will be called from `process_cast.m` after processing has finished. If it is not available nothing happens (i.e. there is no warning or error message).

A Example of a Real set_cast_params.m

```
more off;
clear pk;
mkSADCP('/Data/ADCP/LADDER-1','../data/SADCP/SADCP.mat');

%--------------------------------------------------------------------
% on stn 4, switched heads but not serial numbers in DEFAULTS.expect
% on stn 9, uplooker bombed; after stn 17, no more uplooker
%--------------------------------------------------------------------
if stn == 4
  f.ladcpup = sprintf('../data/raw/%03d/%03ddn000.000',stn,stn);
  f.ladcpdo = sprintf('../data/raw/%03d/%03dup000.000',stn,stn);
elseif stn == 9 | stn >= 18
  f.ladcpdo = sprintf('../data/raw/%03d/%03ddn000.000',stn,stn);
  f.ladcpup = ' ';
else
  f.ladcpdo = sprintf('../data/raw/%03d/%03ddn000.000',stn,stn);
  f.ladcpup = sprintf('../data/raw/%03d/%03dup000.000',stn,stn);
end

f.res         = sprintf('../data/processed/%03d',stn);
f.checkpoints = sprintf('../data/checkpoints/%03d',stn);

f.ctd = sprintf('../data/CTD/at1512_911_%03d_1s.cnv',stn);
    f.ctd_header_lines      = 115;      % file layout
    f.ctd_fields_per_line   = 20;
    f.ctd_pressure_field    = 2;
    f.ctd_temperature_field = 3;
    f.ctd_salinity_field    = 16;
    f.ctd_time_field        = 14;
    f.ctd_time_base         = 0;        % elapsed

f.nav = f.ctd;
    f.nav_header_lines      = 115;      % file layout
    f.nav_fields_per_line   = 20;
    f.nav_time_field        = 14;
    f.nav_lat_field         = 12;
    f.nav_lon_field         = 13;
    f.nav_time_base         = 0;        % elapsed

f.sadcp = '../data/SADCP/SADCP.mat';

%--------------------------------------------------
% before stn 10, computer time was set to local time
%--------------------------------------------------

if stn <= 9
```

```
    p.timoff = 6/24;
end

p.cruise_id = 'LADDER-1';
p.whoami    = 'A.M. Thurnherr';
p.name      = sprintf('%s cast #%d',p.cruise_id,p.ladcp_station);

p.saveplot      = [1:6 9:11 13:14];
p.saveplot_png  = [];

p.ladcp_station = stn;

p.btrk_ts       = 10;
p.sadcp_dtok    = 0;

p.edit_mask_dn_bins = [1];
p.edit_mask_up_bins = [1];

p.checkpoints = [1];

ps.smallfac = [1 0];
```

B Reporting Bugs

I do not receive any salary support for maintaining and documenting the LDEO LADCP processing software or for user support. It is important that I can reproduce bugs reported to me as quickly and easily as possible. I therefore request that any bug report is accompanied by data files and figures illustrating the problem, according to the following instructions:

1. make sure that you are using the newest version of the software available at `ftp://ftp.ldeo.columbia.edu/pub/LADCP/Version_IX`
2. chose a single cast exhibiting the problem
3. create a single directory containing:
   (a) a `README` file describing the problem
   (b) Matlab output `<stn>.mat`, unless none is produced because of a processing error
   (c) log file `<stn>.log`
   (d) ProcFigs. 1, 2, 3, 4, 9, 11 & 14
   (e) all data files required for processing the cast
   (f) specifically adapted `set_cast_param.m` that does not depend on anything except what is provided in this directory
   (g) nothing else
4. email me a `tar` or `zip` file containing this directory

Please note that I want to be able to unpack your file file in a particular directory, start Matlab in the same directory, and use `process_cast.m` to process the data without having to edit anything at all or create subdirectories.

C Debugging and Checkpointing

The LDEO LADCP processing software has debugging and checkpointing capabilities. Currently, these are only documented in the CLIVAR P02 LADCP Cruise Report.

D Checking LADCP Data File Integrity

The following shows the output from verifying the integrity of an RDI Workhorse data file called `111DN000.000` using the `scanbb` program of the Hawaii software (section 2).

```
$ scanbb 111DN000.000

Raw BB-SC-ADCP data file: 111DN000.000
Output will go to file 111DN000.scn
instrument coords


Last ensemble was 769; 185 were bad
zmin:    -32.8  at    666
zmax:    499.7  at    407
zend:    -32.8  at    666

number_with_velocity:     583
first_good_ensemble:       81
last_good_ensemble:       666

downcast:  2004/03/10  17:51:56.81  to  2004/03/10  18:09:43.25
upcast:    2004/03/10  18:09:46.50  to  2004/03/10  18:23:51.30
```

As a side effect the `scan*` utilities write their output into a file with the same basename as the input data file and a `.scn` extension. The `scan*` utility, which works for both downlooker and uplooker data, is optional because it is not actually needed to process LADCP data with the LDEO software. However, it is extremely useful for making sure that i) the available data files are valid and ii) that the instrument was set up correctly.

In case of an invalid data file there is an error message (e.g. `%% ERROR: 1, Header ID or DataSource not found.`). In case of a valid data file from a bad instrument or from a good instrument programmed with bad set-up parameters the values of `zmax` (approximate maximum cast depth) and `zend` (approximate depth at the end of the cast, i.e. zero) are bogus.

E Data Editing

LADCP data processing is hard because there are generally a lot of velocity outliers that have to removed before reliable averages can be calculated. In addition to random outliers, every raw LADCP data files contains portions that are known to be bad. Examples include the data recorded before and after each cast when the transducer is out of the water, data contaminated by side-lobe echo return near the seabed, as well as data contaminated by the echo return from a previous ping. Prior to Version IX the LDEO software relied on random-outlier detection to remove known bad data. In Version IX there are explicit data-editing routines for removing known bad data.

E.1 Random-Outlier Removal

Random-outlier removal takes place during loading of the ADCP data files (`loadrdi.m`), as well as when the super-ensembles are constructed (`prepinv.m`). Furthermore, the inversion is carried out twice: after the first run the top 1% of outlier velocities are removed in `lanarrow.m` before the final inversion is carried out. There are a number of parameters that control the removal of random outliers described in `default.m`. I have not played around with any of these parameters.

E.2 Explicit Data Editing

The currently available data-editing parameters are described in some detail in `edit_data.m`. The following describes the parameters that are most likely to be changed:

`p.edit_mask_dn_bins = [1];` With this parameter, entire bins from the downlooker can be excluded from processing. Our observations indicate that the first bin should almost certainly be excluded from the data of 300 kHz RDI Workhorse instruments. (Because of this, the blanking distance can be set to zero during data acquisition.)

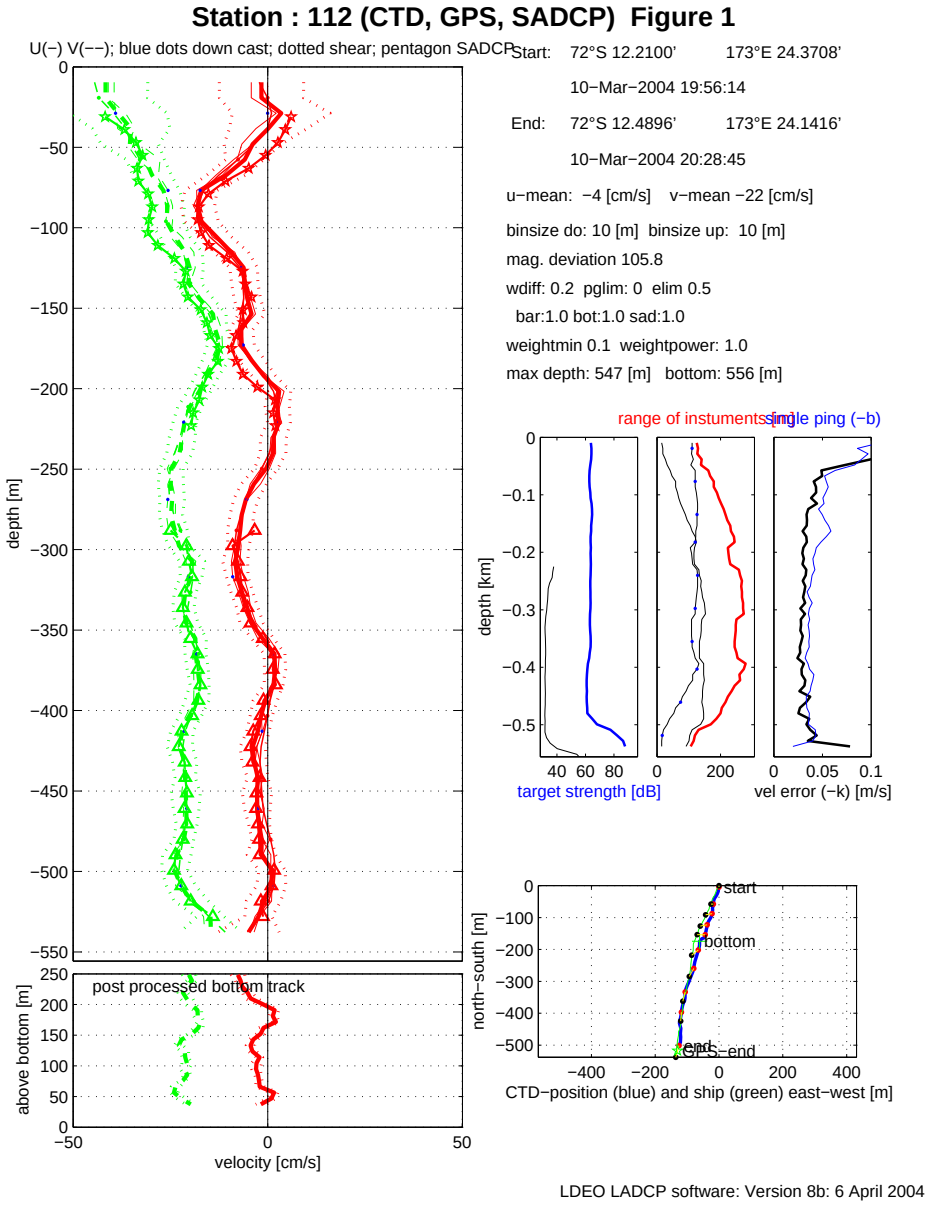
`p.edit_mask_up_bins = [1];` Same for the uplooker.

`p.edit_spike_filter, p.edit_spike_filter_max_curv;` Prior to version IX.3 a time-domain speckle filter was enabled by default (e.g. ProcFig. 14 on p. 30). With RDI broadband 150 kHz instruments set up using staggered pings, this filter appears to be able to effectively remove previous-ping interference, as well as interference from other acoustic instruments. However, it was later found that this filter can cause serious (but difficult to detect) shear bias. *Use with caution.*

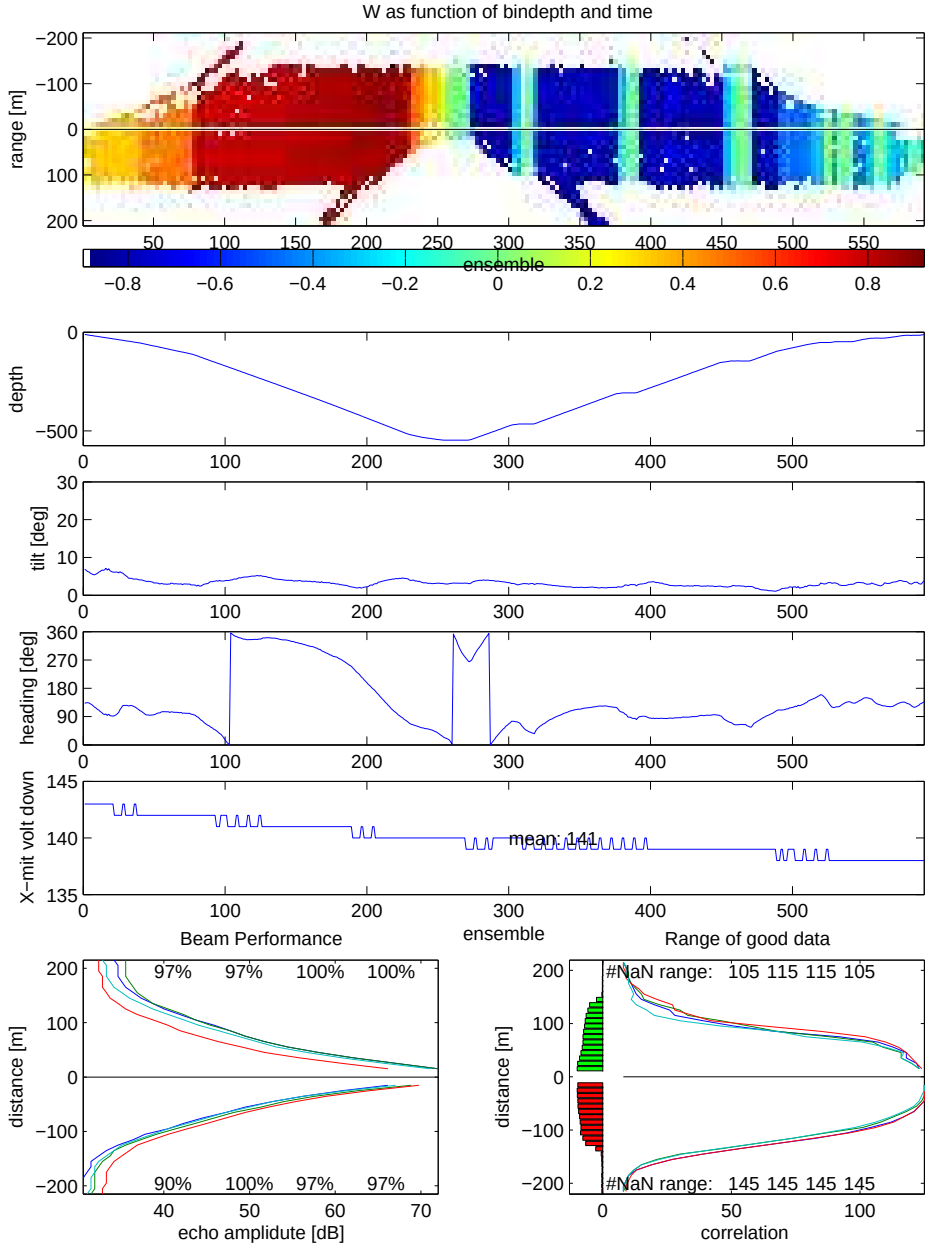
`edit_PPI;` In low-scattering regions it is common for bottom echoes from previous pings to contaminate the velocity data at some height above the sea bed. (This is usually — but not in case of the example shown on p. 30 — quite apparent in ProcFig. 14, which typically shows a trace that is parallel to the seabed both during the down- and upcasts.) If a constant pinging rate is used, the previous-ping interference (PPI) causes a velocity gap a constant distance above the seabed, given by $\Delta z = c\Delta t/2$, where c is the sound speed and Δt is the interval between pings.

Enabling the PPI filter removes the affected velocity data, causing a gap that has to be filled somehow. The best solution is to use a non-constant pinging rate (aka. "staggered pings"); otherwise the gap has to be filled by the smoothing constraint. **Notes:** 1. Prior to version IX.3, the PPI filter did not work correctly for multi-ping ensembles. 2. The current implementation of the PPI filter does not work correctly for multi-ping ensembles with non-constant pinging rates (e.g. for 10 s 3-ping ensembles with a 1 s interval between pings).

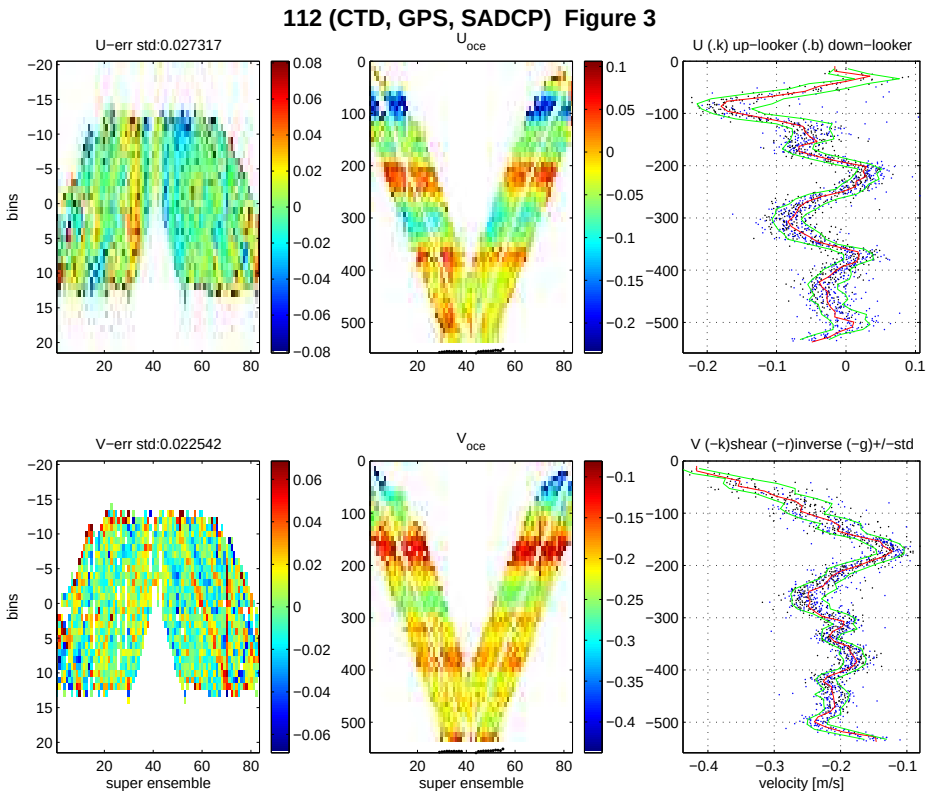
F Processing Figures



ProcFig. 1: Constraint- and solution overview. Main panel: eastward (red) and northward (green) velocities: full solution with error bars, down- & up-cast solutions, shear solution, bottom-track, and SADCP. Bottom-left panel: bottom-track velocities. Bottom-right panel: Ship- and instrument-drift during cast. Center-right panels: target-strength, range and error profiles. Top-right text: meta-data and constraints used.



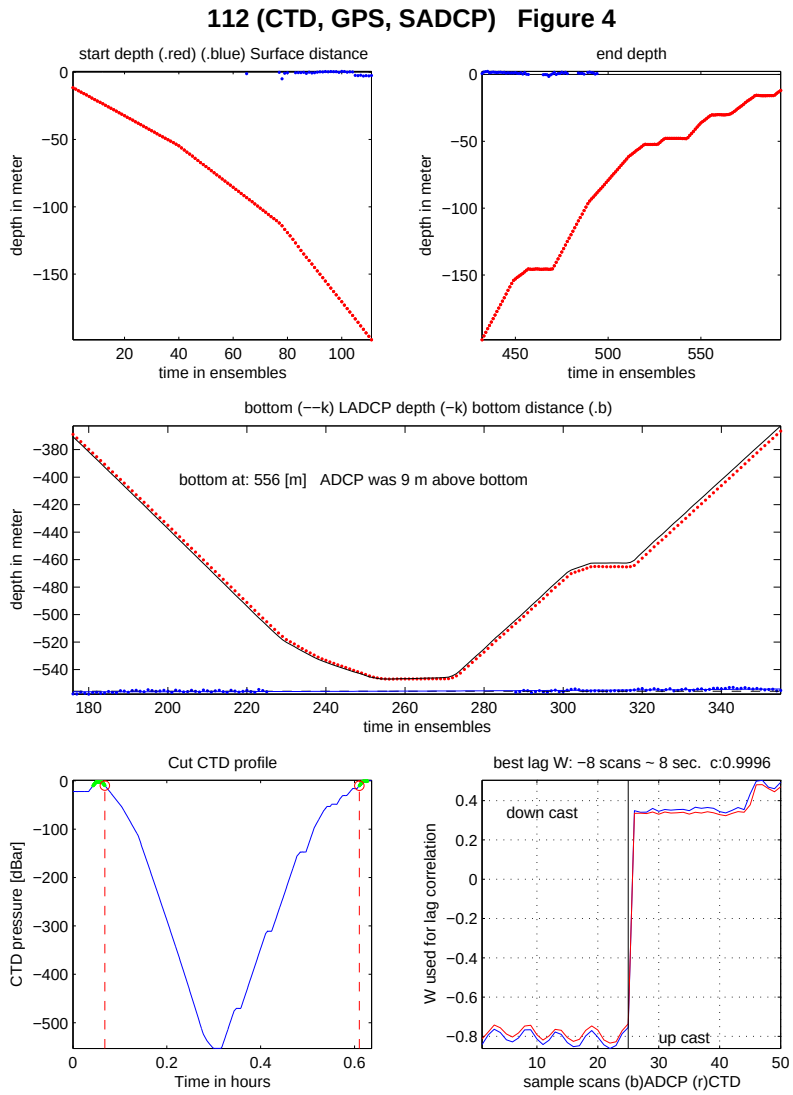
ProcFig. 2: Engineering data. 1st panel: Vertical-velocity-profile time series. 2nd panel: Depth vs. time. 3rd panel: $\sqrt{(\text{tilt}^2 + \text{roll}^2)}$. 4th panel: Heading. 5th panel: Transmit voltage. Bottom-left panel: echo amplitude vs. distance from head for each beam. Bottom-right panel: correlation vs. distance from head for each beam.



ProcFig. 3: Residuals and ocean velocities. Upper panels: Eastward velocities. Lower panels: Northward velocities. Left panels: Residuals. Center panels: Ocean-velocity time-series. Right Panels: Ocean-velocity scatter. Notes:

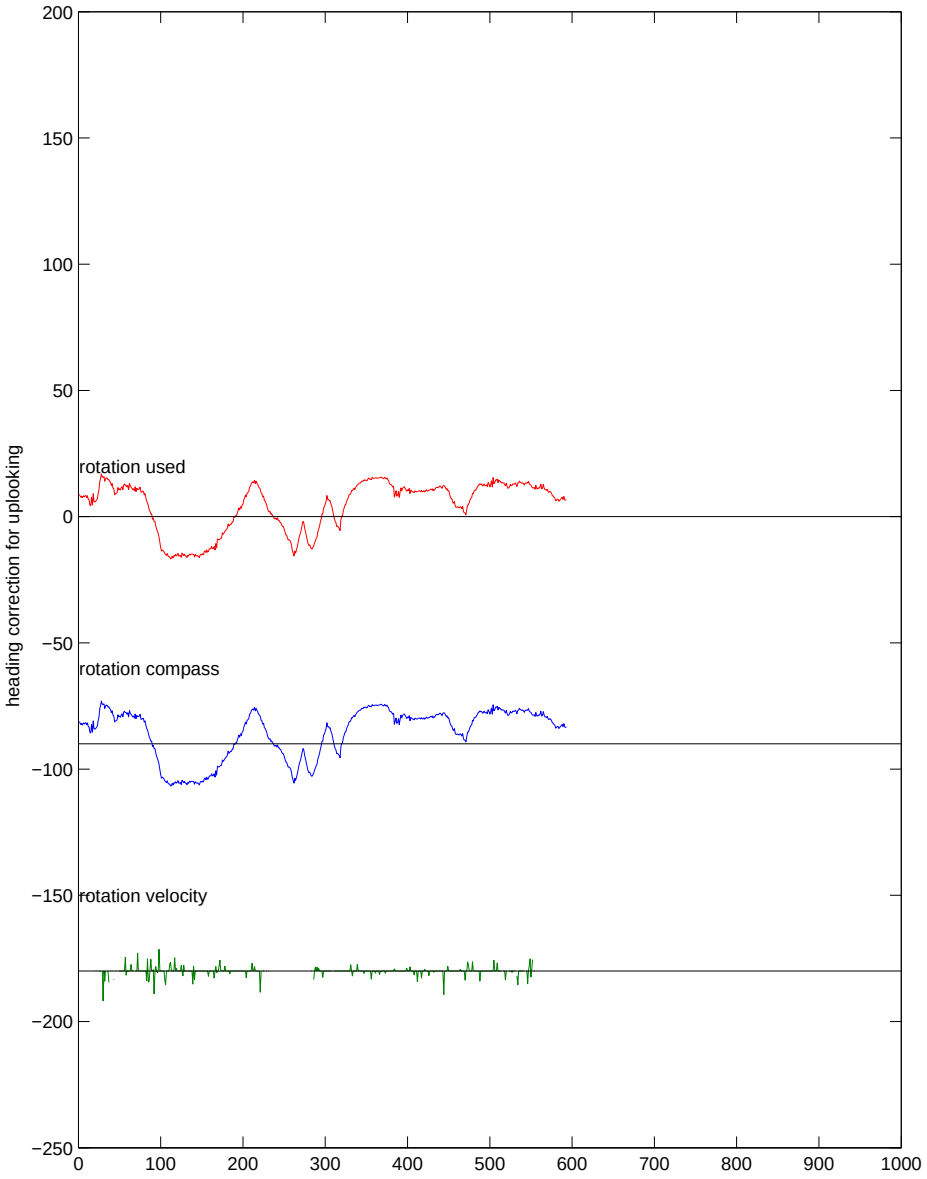
Residuals. The left two panels should be more-or-less random with few structures such as coherent patches or lines. The example shown is acceptable.

Ocean Velocities. For LADCP profiles where there is significant shear (as in the example) the ocean-velocity time series should take on a horizontally banded structure. Vertical or slanted banding indicates data problems.

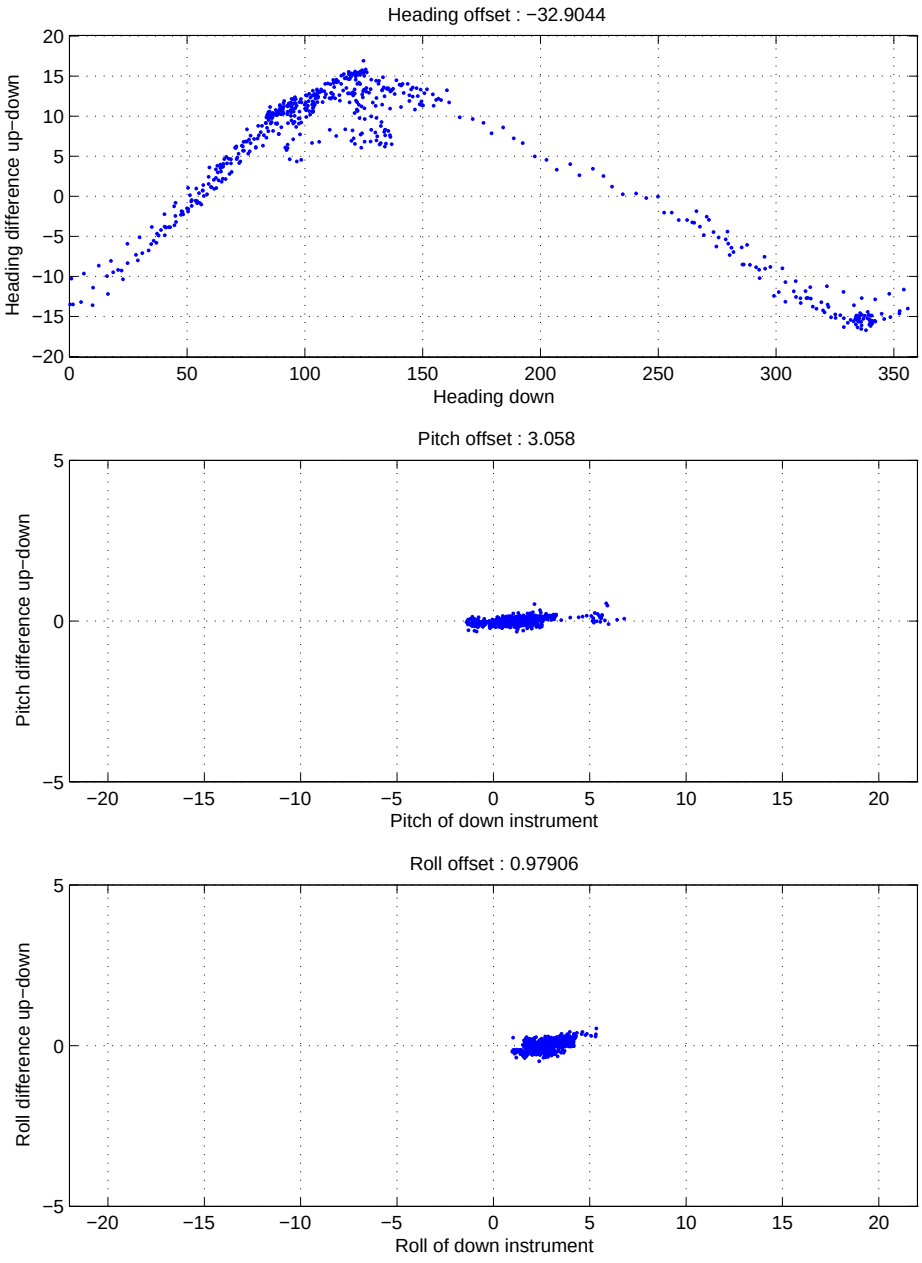


ProcFig. 4: Surface/seabed detection and CTD/LADCP time-series match. Upper panels: beginning and end of cast (red) with detected sea-surface (blue). Center panel: bottom of cast (black: from CTD depth, red: from LADCP w) with detected sea-bed (blue). Bottom-left panel: Time-range of cast (dashed red lines). Bottom-right panel: details (left: downcast, right: upcast) from CTD/LADCP time-series match. Notes:

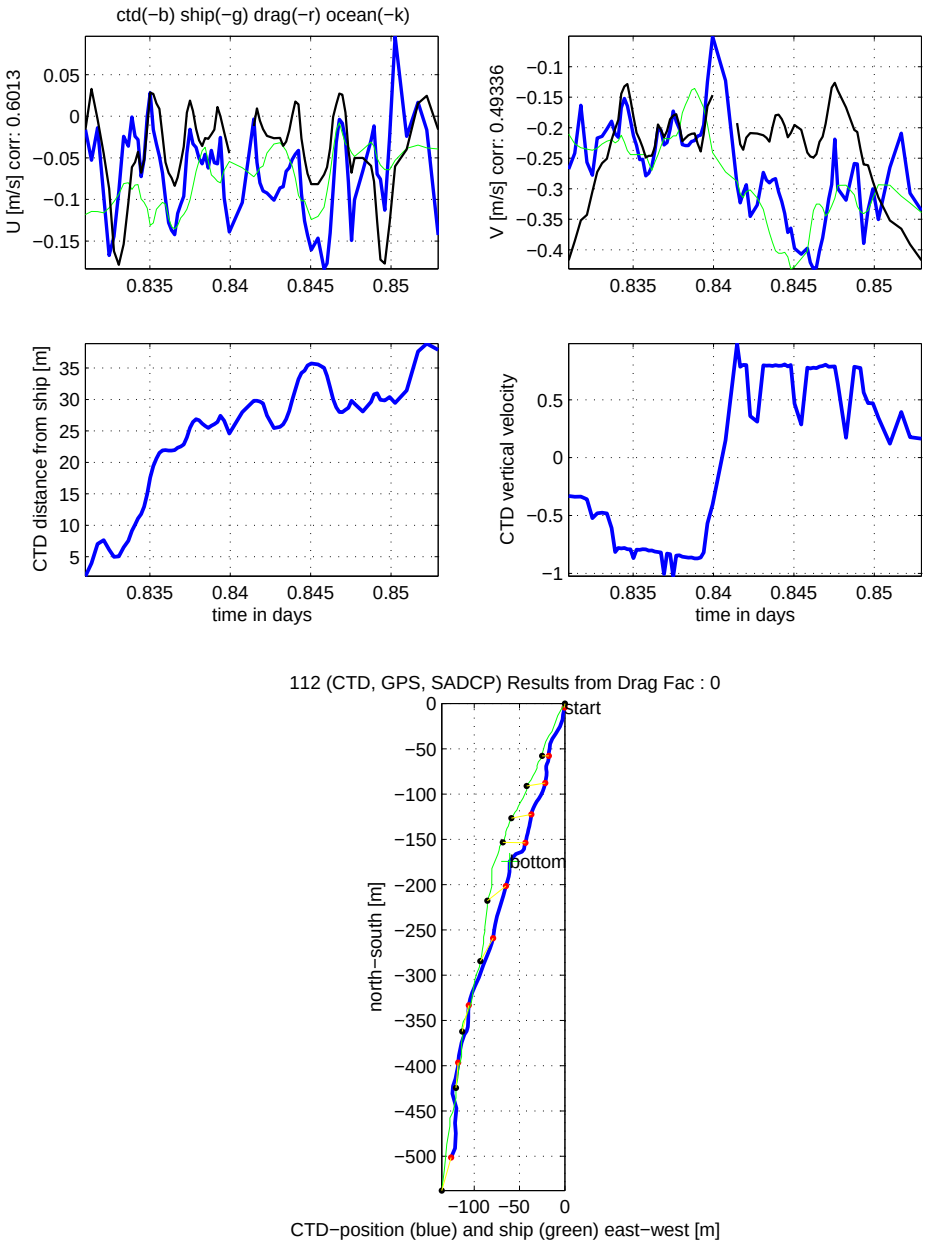
Time-Series Match. Vertical offsets between the CTD- and LADCP-derived depth vs. time profiles are due to biases in the LADCP w measurements and do not indicate time-series mismatches, which manifest themselves as horizontal offsets of vertical accelerations (bottle stops and/or waves).



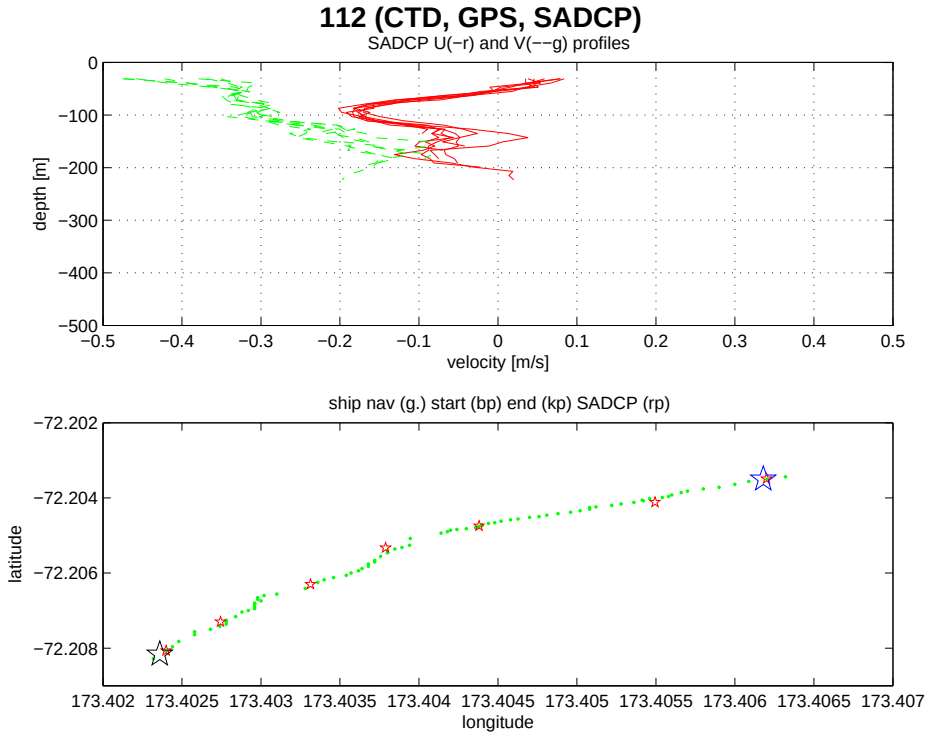
ProcFig. 5: Compass comparison. Rotation-offset time series.



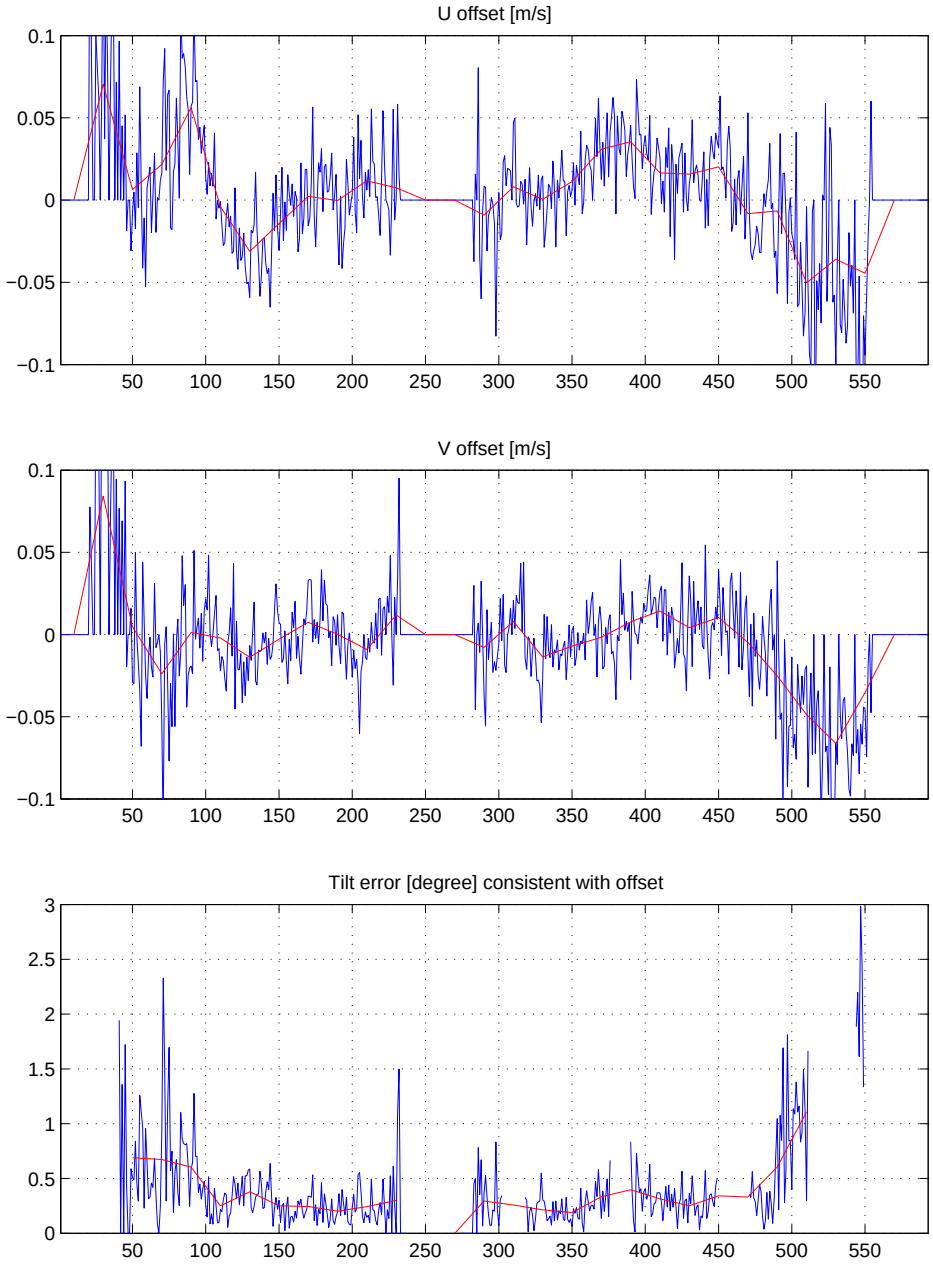
ProcFig. 6: Heading-, pitch- and roll offsets.



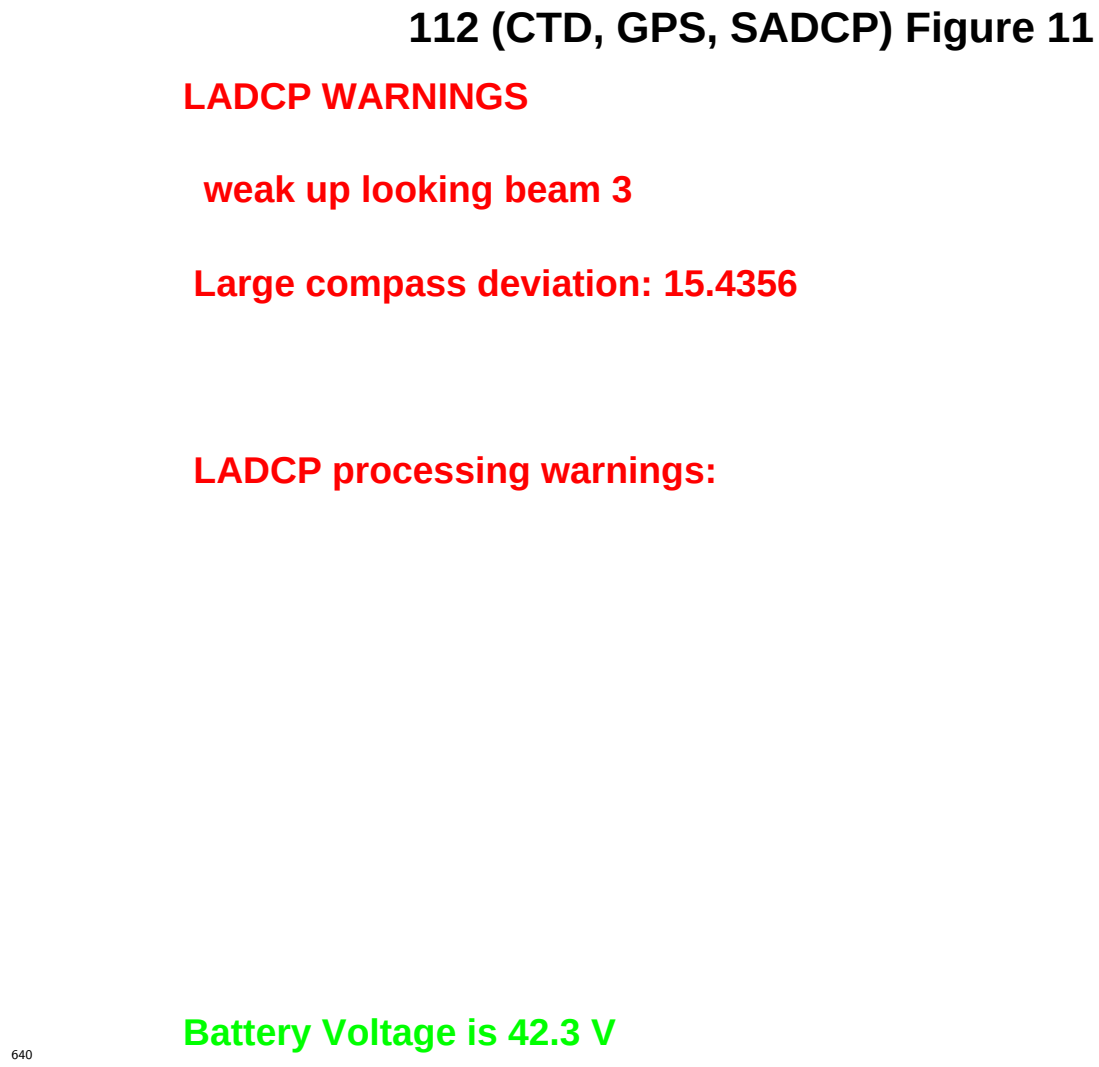
ProcFig. 7: Drag-model diagnostics. (By default, drag-model constraints are disabled `Drag Fac : 0`.)



635 ProcFig. 9: SADCP data. Upper panel: Ensemble of ADCP velocity profiles in time-range of cast. Lower panel: Shiptrack from GPS (blue stars and green dots) and location of SADCP profiles (red stars).



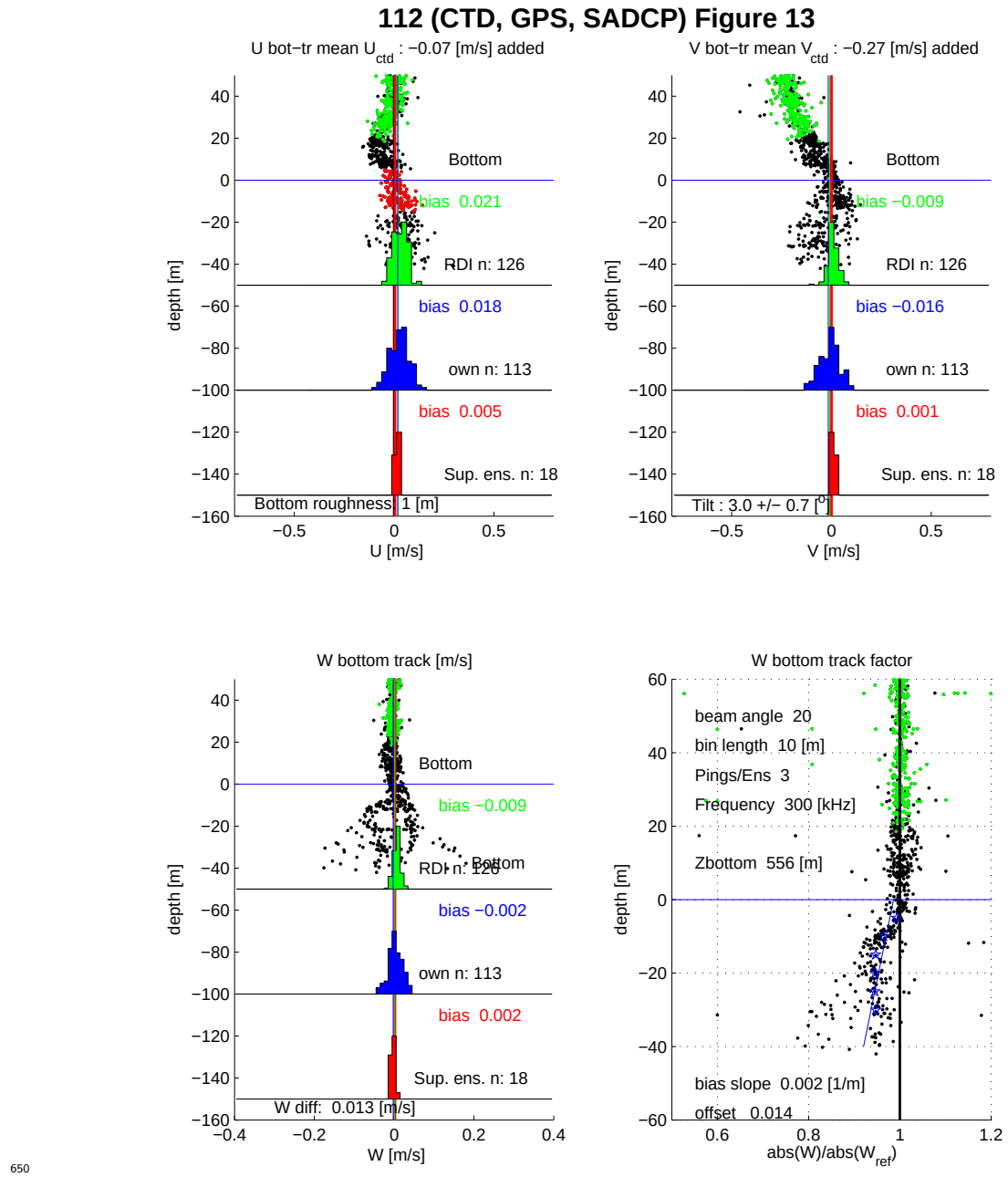
ProcFig. 10: Tilt-error estimate from velocity offsets.



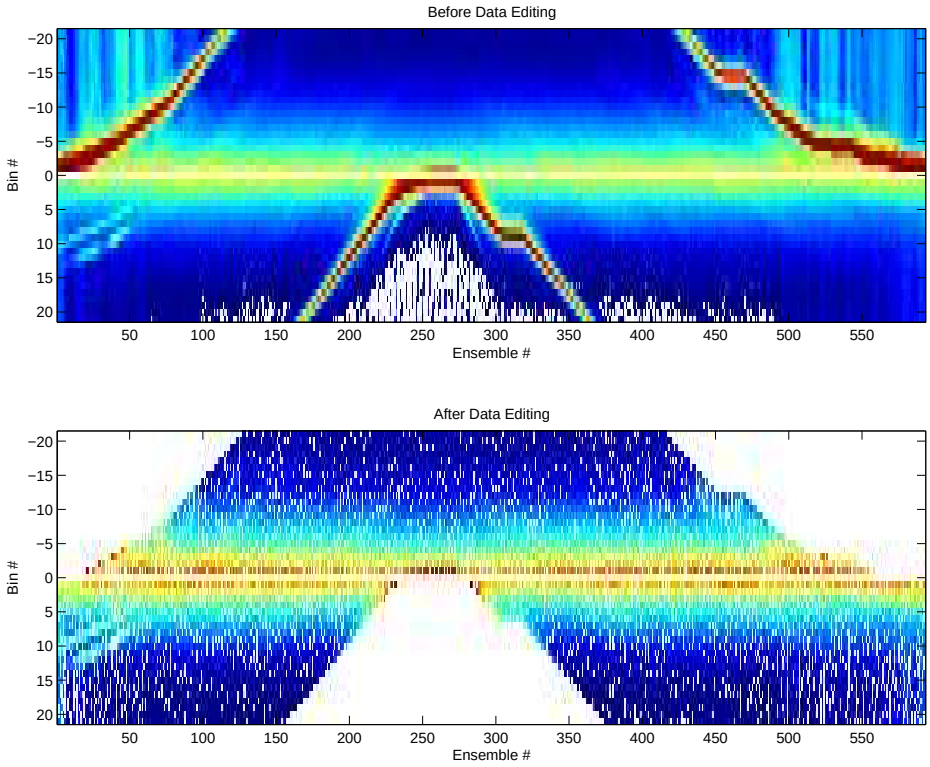
ProcFig. 11: Processing warnings. Notes:

Seriousness of Warnings. The warnings that are shown in this ProcFig. are a subset of the warnings that are written into the log files (`*.log`). They are considered potentially serious and should be investigated. The warnings that made it onto this figure are known to have been early-warning signs of serious instrument problems in the past. (In case of the examples, both warnings can be ignored: the first one turns out to be true but not fatal; the second one is only serious if it occurs in many casts.)

Instrument Voltage. This is not to be trusted unless the software has been calibrated for a particular instrument.



ProcFig. 13: Detailed bottom-track diagnostics. This ProcFig. is difficult to interpret in detail but it can be used to assess potential bottom-track bias due to sidelobe contamination.



ProcFig. 14: Data-editing diagnostics. Upper panel: raw signal-strength time series. Lower panel: signal strength after editing (side lobes, acoustic interference, &c).