Glider Processing

The following describes the toolbox version 2, GEOMAR PO subversion 930, 16 June 2022.

General

This wiki describes the GEOMAR version of processing Teledyne Webb Research Slocum Glider data from their raw state on the glider to a final calibrated state. The processing is nearly completely executed by the Matlab m-files of the toolbox. Teledyne Webb Research executable files are used to convert the raw data into Matlab readable files.

A large number of settings, paths, choices, and other infos are required for the processing. These are contained in a single Matlab m-file `processing_parameters.m`. This file needs to be properly edited for the processing to succeed. Apart from editing this file usually no user input is required.

The processing takes place in a single folder. Three subfolders (`raw_bd`, `raw_mat` and `plots`) will be created therein.

- First add the provided toolboxes to your Matlab path
- To set up the processing for a deployment, cd to a main processing folder and type step_00, the TAB key and ENTER.
- cd to that folder that was created by this step and edit processing_parameters.m.
- To run the processing type step_, the number of the step you want to execute (e.g. '01'), the TAB key and ENTER.

Requirements

- Steps 11 and 15 minimize an area between two curves in TS-space. This can be done either with Matlab's **Mapping Toolbox** or with a function developed at GEOMAR. The speed is comparable, but the results can differ. The optimization seems not to converge too well. When using GEOMAR's function the TEOS-10 toolbox is required.
- Steps 11 and 15 use nonlinear optimization. This can be done with either Matlab's **Optimization Toolbox** or with a toolbox by R. Oldenhuis https://de.mathworks.com/matlabcentral/fileexchange/24298-minimize . The Matlab version is a factor 3-6 faster than the other one.

References

- Garau, B., S. Ruiz, W.G. Zhang, A. Pascual, E. Heslop, J. Kerfoot, J. Tintoré, Thermal lag correction on Slocum CTD glider data. Journal of Atmospheric and Oceanic Technology 28 (9), 1065-1071, 2011. https://doi.org/10.1175/JTECH-D-10-05030.1
PDF
- Merckelbach, L., D. Smeed, G. Griffiths, Vertical Water Velocities from Underwater Gliders. J. Atmos. Oceanic Technol., 27, 547–563, 2010. https://doi.org/10.1175/2009JTECHO710.1.
PDF
- Merckelbach, Lucas, Berger, Anja, Krahmann, Gerd , Dengler, Marcus and Carpenter, Jeffrey R. (2019) A dynamic flight model for Slocum gliders and implications for turbulence microstructure measurements. Open Access Journal of Atmospheric and Oceanic Technology, 36 . pp. 281-296. https://doi.org/10.1175/JTECH-D-18-0168.1.
PDF

Known Problems

- The optimizers in steps 11 and 15 are not always well defined. Garau et al.'s relies on one of Matlab's toolboxes and gives wrong results when the TS curves do loops. And GEOMAR's function does not converge well for different starting points.
- The optode functions in step 17 are a bit convoluted and could be simplified.
- The best position estimate in step 05 should be checked again. While in most cases the result is ok, sometimes there appears to non correct conclusions on the positions.

step_00_prepare_processing.m

Description

The GEOMAR version of the processing of Slocum glider data is all executed in one folder. Intermediate and final data is stored in subfolders therein or in mat-files in this folder.

As a preparatory step you need to make copies of the contents of the two glider flash cards accessible.

Output

The main folder will be created and a parameter m-file template `processing_parameters.m` will be copied there.

processing_parameters.m

The whole file `processing_parameters.m` needs editing in this step. You have to go through all steps in this wiki for the relevant settings for each processing step.

`op.no_plot=0;` set this to 1 prevent most bandwidth consuming plotting commands. In particular when running over VPN.

`op.deplname='ifm14_depl06';` set the deployment name that will be used in filenames.

`op.suna_installed = 0;` set to 1 when a SUNA is installed. This will create an additional file to be used for SUNA processing.

`op.microrider_installed = 0;` set to 1 when a microrider is installed. This will create an additional file to used for microrider processing.

step_01_copy_bd_to_processing_folder.m

Description

The GEOMAR version of the processing of Slocum glider data starts from full copies of the two flash cards on the gliders. These copies should be stored somewhere on the file system. You can however simply copy all `*.*bd` files to the folder `raw_bd` in case you do have only these files.

Output

Files will be copied to `raw_bd`. Please note that this folder is not purged before copying. Should there be old files, these will remain there or be overwritten.

processing_parameters.m

The paths to the copies need to entered in `processing_parameters.m` similar to the following examples (you will have to enter the correct path for your data):

```
op.path_to_main_state_copy =
'/data2/states/ifm12/main/ifm12_main_2014_04_18/';

op.path_to_science_state_copy =
'/data2/states/ifm12/science/ifm12_science_2014_04_18/';.
```

This processing step will copy **all** data files (`*.dbd *.mbd *.sbd *.ebd *.nbd *.tbd` with both lower and upper case endings (copies of the flash cards sometimes result in lower or in upper case file names, depending on the operating system in use) to the subfolder `raw_bd` and then apply the

executable supplied by Teledyne Webb Research that renames the file names from 8-digit.*bd file names to expanded file names which contain the glider name and other information.

Sometimes data from old deployments is still present on the flash cards. To limit the missions used in the subsequent analysis two variables in `processing_parameters.m` can be modified from their default values:

E.g. set `op.minimum_mission_number = 10; op.maximum_mission_number = 20;` . to use include data only from missions 10 to 20. The default values are 0 to 9999. The mission number of the data file are the first 4 digits of the data files. E.g. `00110000.ebd` would be the ebd file from the first segment (the `0000`) of mission `11`. Should you have only renamed files, this numbering scheme is not available from the file names. It can however be found in the files themselves. Even though the data files are binary you can open them in a regular text editor and in the first few lines regular ASCII characters show some information including the 8-digit compressed file name.

step_02_binary_bd_to_ascii.m

Description

Three substeps are executed in this processing step:

- determination of the files containing the most complete data
- merging of main and science data
- conversion to ASCII data and a Matlab file that sets the variable names.

Slocum glider data can take a lot of storage space. In cases of full flash cards Slocum gliders start to delete older data to be able to store new data. Since the full data sets contains a lot of information that is not scientifically required, files with less than the complete data can be stored. In case of full flash cards the full data set files are delete while the reduced data sets are retained thereby still allowing a full scientific analysis. In the first substep, it is examined which of the files is available and the most complete is used for the further processing.

Slocum glider data is stored partially on the mainboard flash card (system and navigational data) and partially on the science bay flash card (all scientific data). These files need to be merged for the further processing. Teledyne Webb Research has supplied executable programs which perform this.

Slocum glider data is stored in a specifically developed binary format. Another Teledyne Webb Research executable converts these binary files into pairs of ASCII and Matlab m-files both of which have file names similar to the original expanded file names.

Output

`*.*bd` files found in `raw_bd` will be converted and the results be stored in `raw_mat`.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_03_load_ascii_save_mat.m

Description

The ASCII and m-files written by the previous step are quite slow to load into Matlab. In particular, if the glider has been deployed for several weeks in which case there are a few hundred of these files. Another problem is the naming and data storage convention employed within these files. All data is stored in a single variable `data` and the information in which column which variable is stored is contained within the variable itself. In this step the files are loaded into Matlab, the data is reorganized to contain the data of each variable under its own name.

The data is stored again as a fast-loading Matlab mat-files with names similar to the original ones.

Output

Matlab mat-files will be stored in the folder `raw_mat`.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_04_merge_mat_files.m

Description

During the previous step the conversion kept the order of the original files created on the glider. As there can be several hundred of these files, the loading is complicated. Here all relevant variables (set in the file `slocum_variable_list.txt` in the Matlab toolbox) are loaded from each of the file and concatenated.

Output

The result is stored in a single mat-file named `deployment_name_1vector.mat`.

processing_parameters.m

One setting can be entered in `processing_parameters.m`.

`op.gps_overrun = 0;` Set this to 1 to fix wrong GPS dates stemming from the 1023 week limit in GPS receivers. If not set a default of `0` will be used.

step_05_best_position_vector.m

Description

The glider can determine accurate position fixes only at the surface through GPS. Under water Slocum gliders estimate their position via dead-reckoning. I.e. they know their pitch angle and heading and from changes in the pressure readings they can calculate the distances they should have moved through water. This does however not take into account ocean currents.

GPS systems have a warm-up of up to a few minutes period during which they establish reception of the satellites. During this time the determined position is not optimal and can be off by a significant distance. Slocum gliders thus gather positions for a few moments until they finally determine a trusted position. All positions during the waiting time are recorded and stored in a separate variable. Positions which differ from previous positions by too much are deemed not trustworthy and stored in another variable.

Sometimes Slocum gliders deem a bad position trustworthy. This leads to all subsequent good positions being flagged as not trustworthy. The original data is however still stored in the system. In this step `all` available position variables are examined for such false trustworthy positions and it is attempted to fix the problems.

Once the set of really trustworthy GPS positions has been determined the difference between the dead-reckoned surfacing positions and the first GPS position is determined. This is a measure of integrated ocean currents during the subsequent dive. As no better information is available, the difference is distributed linearly over the underwater positions so that the GPS positions at the beginning and the end of the dive agree with the new underwater positions.

Output

The resulting complete set of positions is stored as new variables `n_lat` and `n_lon` in `deployment_name_1vector.mat`. This data is now deemed to be the optimal position estimate of the glider.

processing_parameters.m

Two settings can be entered in `processing_parameters.m`.

`op.use_median = 0;` is a special treatment for deployment 9 of glider ifm02. In that case set this flag to 1, otherwise leave at 0.

`op.min_dive_length = 250;` can be used to recognize very short dives and prevent the application of the position correction routine. The number 250 is about equal to 10 minutes. If there is a gap in GPS positions longer than this, we assume that we have a dive and apply the correction routine. There should be no need to change this.

step_06_cut_up_down_parts.m

Description

In step 4 the data from all files was concatenated. For the scientific analysis of the data it is however better to have separate single down and up parts of the glider's movement. In Teledyne-Webb-Research-Speak these are **yos**. In this step the navigational pressure record is examined to find upper and lower inflection points.

Output

The result is stored in a single mat-file named `deployment_name_yos.mat` retaining Slocum variable names.

processing_parameters.m

Two settings can be entered in `processing_parameters.m`.

`op.top_turnaround_limit = 1/3;`. In Slocum gliders one can command the glider to do multiple dives between surfacings. Imagine the glider doing a **W** shape dive pattern. Starting at the surface, diving, coming up, diving again, and coming up again. In the mission commands one can set the depth of the upper inflection point (the center of the **W**). In this routine we assume these points to be shallower than 1/3 of the lower points of the **W** (the maximum dive depth commanded). Usually 1/3 works well, but for e.g. a maximum dive depth of 200m and an upper inflection point of 100m, this would not work. In this case set this settings to 3/5 which is a bit larger than the upper inflection depth (100m) divided by the lower inflection depth (200m). We have to do this since the mission settings of the inflection depths are not available from the data itself.

`op.minimum_deepest_depth_of_dive = 16;` This setting allows us to discard data from dives with very shallow dive depths. Usually these are only one or two test dives at the very beginning of the deployment. 16m is a good value for deep gliders. For a shallow glider in shallow waters this has

to be set to a smaller number. E.g. 2m or so.

step_07_cleanup_science.m

Description

Slocum glider science data sometimes has bad values. Usually this is the case when a sensor has not delivered a value right after the start of the mission, or comparable minor problems. In this routine all yos are inspected and the bad data is replaced with NaN.

Some bad data can not be detected automatically. E.g. when something occupied the conductivity cell. In such cases the bad data will lead to bad optimizations in steps 11 and 15. In this step data can be set to NaN by creating a file `bad_data_list.txt`. In this file you can enter lines similar to: `sci_water_cond 1819 1820 1 1000000`. This line would set all values of the variable `sci_water_cond` for yos 1819 to 1820 to NaN. `sci_oxy3835_wphase_oxygen 1819 1819 15 20`. This line would just set the 15th to the 20th value of yo 1819 of the variable `sci_oxy3835_wphase_oxygen` to NaN.

Output

The result overwrites the single mat-file named `deployment_name_yos.mat`.

processing_parameters.m

No information needs to be entered `processing_parameters.m`.

step_08_correct_pressure.m

Description

Slocum gliders typically have two independent pressure sensors. One is always installed and is used for the glider's navigation and one is part of the CTD package. These two sensors are treated in different ways which are particular to their purpose and result in distinctive properties.

The navigational pressure sensor:

- is always on,
- is of a lower quality and lower energy consumption with significant noise,

- is reset to 0bar at the beginning of a dive,
- can be calibrated in lab with the proper equipment,
- sometimes deviates signifcantly from the higher quality CTD pressure sensor,
- sometimes shows a distinctive 2-3dbar jump at about 5dbar.

The CTD pressure sensor:

- is only on when the CTD is turned on,
- is of higher quality with very little drift over many years,
- is `not` reset to 0bar at the beginning of a dive,
- can only be calibrated at the manufacturer,
- shows no jumps.

To obtain a complete set of navigational information, even in those times when the glider did not record scientific data, the problems of the navigational pressure sensor (noise, deviations from the CTD pressure sensor, jumps) need to be overcome. In this step first a routine determines whether the sensor suffers from the jumps. If necessary, these are removed. Then the usually very small (<1dbar) surface offset of the CTD pressure sensor is determined and removed. The final step is the calculation of a constant factor which, when applied to the navigational pressure data, brings it into best agreement with the CTD pressure data.

See also:

slocum_pressure_sensors_v1.doc

Output

The result creates or overwrites the single mat-file named `deployment_name_yos.mat`.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_09_interp_to_1sec.m

Description

Slocum glider's main processors usually run with an internal cycle of 4 seconds. Meaning every 4 seconds system status sensors are read and e.g. navigational computations are performed and decisions made. The science processor runs independently of that with another cycle (1 second?) and reads the scientific sensors. Most Slocum scientific sensors work in a boardcast mode. I.e. they run on their on pre-set cycle time and send the data to the science processor to be read. These cycle times are typically 1 or 2 seconds. A very small number of sensors work in a polled mode, where the science processor requests data from the sensor and only then a measurement is performed. This means that all science data comes in at various times. If programmed properly, the glider records the time when

each sensor sends its data (timestamps). In this processing step we linearly interpolate `all` data (both system and science) to full seconds of the main processor clock. For most system variables this means an oversampling by a factor of 4, while for science sensors it either means an oversampling by a factor of 2 or just a resampling at the sensors cycle. The drawback of the linear interpolation is a likely small additional noise. The advantage is a much simplified processing afterwards. As the glider moves by no more than 0.2m in the vertical over 1 second, the advantages outweight the disadvantages.

During this step the original Slocum variable names are replaced by simpler, more generic ones. E.g. the different Aanderaa optode types each result in different variable names. Here `oxygen` is used for whichever of `sci_oxy3835_oxygen`, `sci_oxy3835_wphase_oxygen`, and `sci_oxy4_oxygen` actually contains the data.

Output

The result is stored in a single mat-file named `deployment_name_1sec.mat`.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_10_prepare_flow_speed.m

Description

In this step the speed of the glider through the water is estimated from its pitch angle and the changes in observed pressure.

Output

The result is stored in a single mat-file named `deployment_name_1sec_derived.mat`.

processing_parameters.m

One setting can be entered in `processing_parameters.m`.

`op.pitch_for_flow_speed = [];`. This should be set to `[]`. Only for testing purposes a fixed and constant pitch angle in degrees can be entered.

step_11_optimize_temp_for_cond_params_dpdt.m

Description

Seabird CTD systems installed in Slocum gliders come in two flavours. The older one has an open conductivity cell on the left side of the glider hull. Seawater is not pumped through this cell but the replacement of the measured water relies on the movement of the glider through the water. The newer version of Seabird CTDs has the conductivity cell inside of the glider hull and relies on a small pump to get the water into the cell. For the calculation of the salinity of the seawater sample the temperature of the sample during the measurement needs to be known and used. In the older systems this temperature is not exactly the same as the seawater outside of the cell but it is influenced by the thermal mass of the cell itself (Garau et al., 2011). The measured temperature used in the calculation is however not influenced by the thermal mass of the conductivity cell. One thus needs to modify the measured temperature in a way to mimic the unknown temperature inside of the cell. This can easily be seen when the glider passes from a warm surface layer into colder ones. The non-zero thermal mass of the conductivity cell leads to wrong salinities which appear like lagging sensors (salinities from the surface are *carried* downwards on the descent and salinities from the depth are *carried* upwards during the ascent. In reality the salinities of the water were the same. The strength of the *carrying* depends on the speed of the glider. Here a set of parameters is estimated through a non-linear optimization that creates a modified temperature with which a better salinity estimate can be calculated.

In theory one would like to minimize the salinity difference between subsequent up and down profiles and that best in density space. To do such a comparison one needs however to grid the data onto a regular depth, pressure or density grid. This is computationally very expensive. Instead Garau et al. (2011) optimized the area between the two curves in the TS diagram which avoids the gridding. Unfortunately there seems to be no free Matlab code for such an area calculation (the problem is that the curves can intersect). Here we developed code for another optimization goal: A simplified integration of the area between the two curves in Density-Spiciness-Space. This should significantly reduce problems with curved curves in the TS diagrams. To keep the calculations really simple and quick we use pratical salinity, in situ temperature and lat,lon=0 instead of the correct values. For the optimization this is likely not a problem. Unfortunately both Garau et al.'s and our version are somewhat erratic in the result of the optimization. Thus we usually try optimizations on several different reduced data sets and hope that one attempt finds the best one. The best one in this case is the result with as few as possible density inversions, as small as possible difference between subsequent up and down salinity profiles, and as small as possible an area in the TS diagram when calculated for the first half of the deployment.

For the newer pumped CTD systems the resulting parameters are always very similar and amount to basically no correction.

Output

All results from the optimizations are stored in a mat-file named `deployment_name_dpdt.mat`. Should this step be rerun with different of the settings below this will be stored as additional results. Salinities from the best result will be calculated and stored in `deployment_name_1sec_derived.mat`.

processing_parameters.m

A number of settings can be entered in `processing_parameters.m`.

`op.yo_numbers = [];` During some deployments the conductivity measurements became progressively worse. This was usually caused by bio-fouling on the conductivity cell. In these cases only the yos during which the data still appears to be ok should be used for the derivation of the parameters. Here a vector of yos to be used can be entered. Typically `[1:half_of_the_maximum_yo_number]` is ok.

`op.subsample_profile = [];` The non-linear optimization is computationally very costly. Here a reduction of the available down/up pairs to be used can be set. The default is to do three calculations with every 3rd, every 6th, and every 10th pair only. The default is selected with an empty vector. Should you have a very short deployment or only very few down/up measurement pairs you can set this to 1} in which case all pairs will be used.

`op.ts_data_reduction = 5;` This again is a parameter to limit the computation costs. As the CTD measures every one or two seconds, not all values are always required for a good optimization result. Here one can enter a reduction factor. In cases when you have few down/up pairs or when you have very strong temperature and salinity gradients you might try a lower number, or even 1.

`op.area_function = 1;` Set to 1 to use Garau et al.'s area calculation (requires Matlab's Mapping Toolbox). Set to 2 to use a function developed at GEOMAR that does not rely on the toolbox. The results of the two functions differ and sometimes one and sometimes the other delivers a better result. You can try both of them by changing this parameter and rerun the step.

`op.is_pumped_ctd = 0;` Set to 1 to indicate a pumped CTD. When a pumped CTD was in use there is no need for glider speed dependent corrections.

Rarely changed:

`op.minimum_flow_speed = 0.15;` In step 10 the flow speed through the cell was estimated from changes in pressure. There are times, when this pressure does not change (i.e. the inflections or strong internal waves). In these cases the temperature calculation becomes instable. It is usually already badly determined when the glider is very slow. Here a minimum speed that is used in the calculation can be entered. 0.15m/s appears to be a reasonable number.

`op.flow_speed_degree = 0;` The flow speed through the cell is in the simplest approach simply a constant factor times the speed of the glider. The original papers based on lab and in situ tests of CTDs do however suggest that this factor is not constant but itself dependent on the glider speed. In a few cases the resulting salinities are still spiky when a degree of 0 is applied. Here one can try a degree of 2 and see whether the salinities improve. Of a degree of 2 is set, `op.minimum_flow_speed = 0.05;` should be used.

`op.min_depth4cond_temp = -10;` One can restrict the data used for the determination of the parameters to data only from depth deeper than this value. Often the data near the surface deviates a lot from the rest of the data. It is unclear whether including these values in the optimization leads to better or worse results as they can dominate the value that is minimized. If you set this to e.g. 10, the uppermost values will not be optimized only the deeper ones which are 'cleaner'. The default is -10m, which is *above* the surface. Thus all data will be used in that case.

`op.force_tomeu_params = [alpha_offset,alpha_slope,tau_offset,tau_slope];` Here

you can force the use of a set of 4 parameters for the conductivity calculation. Use this in case the optimization fails or you measured only during down or up casts. The 4 parameters are the 4 free ones from Garau et al. (2011). We used for unpumped CTDs `[0.05,0.07,1e-6,16]` with good results. `[0.0733,0.0303,24.1544,1e-14]` seems to work too. For pumped CTDs the optimization gives quite different results. `[0.0115315,7.43849e-16,53.6428,9.99094e-11]` look good.The 4 parameters appear not to be totally independent. So there is some ambiguity in what are the optimal ones. Please note that in particular the first parameter depends on `op.minimum_flow_speed`. You can play with different parameter sets for the delay functions by calling `func_plot_delay_function` directly.

step_12_prepare_shearlift.m

Description

In some deployments in the equatorial Atlantic we found that the otherwise well working flight model (calculated in the following steps) did not work well. The deviation from model and observed vertical glider speed always occured in the same depths and was dependent on the zonal and vertical direction of the glider movement. After lengthy deliberations it occured to us that one assumption of the glider flight model, namely a constant background reference through which the glider is flying is violated in this special case. The vertical shear in this region is very large in this region with a typical westward surface current of some 0.3m/s and an eastward current of 1.0m/s only 30 to 40m below. A descending glider initially starts out in a westward moving reference frame and then enters an eastward moving one. The inertia of the glider has a memory of the reference frame from which the glider comes and thus the glider has a horizontal speed that is not in equilibrium with the flight state it would have in an unchanged reference frame. It is actually the inclined glider (body and wings) which now create an additional down- or upward force.

This could all be modeled properly in the flight model by adding inertia and calculating the proper speeds in the changing reference frames. However here we parametrized the additional vertical force as the product of sine of glider pitch times shear in the glider movement direction times a constant factor that was optimized in calculations.

In this step the additional shear-lift force is calculated from externally supplied current profile data and stored for use in the subsequent calculations.

The externally supplied current profile can have two forms and needs to be prepared by the script `make_current_profile.m` in the glider deployment folder. Within this script one can either enter a simple profile in vector form or can arrange for time dependent data in a mat-file.

The simple profile version requires the following lines in the script (of course you need to set the currents properly): `u_prof = [-0.3,-0.3,0.7,0.3,0,0]; v_prof = [0,0,0,0,0,0]; z_prof = [-100,0,65,110,180,2000]; save deployment_name_currentprofile u_prof v_prof z_prof`

The time dependent variant needs again to supply the mat-file `deployment_name_current_profile.mat` e.g. via the following `!cp`

`../current_data/eq_10w_2011.mat deployment_name_currentprofile.mat`. This file then requires the 4 variables `u_prof`, `v_prof`, `z_prof`, and `time_prof`. `u_prof` and `v_prof` need to be current component arrays (in m/s) with the first dimension covering the depths and the second dimension covering the time. `z_prof` needs to be a depth vector of the current data and `time_prof` a time vector (in Matlab datenum format) of the current data. You should make sure that the profile data covers the full depths of the glider deployment. I.e. from surface to maximum depth. One can make sure of this by adding dummy current values above the surface (-10m) and very deep (2000m).

Output

Results from this step will be stored in `deployment_name_1sec_derived.mat`.

processing_parameters.m

`op.shear_lift_factor=40;` This is the factor determining the strength of the shear-lift. Values between 20 and 60 resulted in the best agreement in our deployment in the equatorial Atlantic.

step_13_optimize_dynamical_model.m

Description

he flight of gliders through the water should be very stable as only few forces act on them. In the simplest way of thinking this would be gravity (reduced by the glider's buoyancy), drag, and lift from the body and the wings. Merckelbach et al. wrote a paper in 2010 outlining the forces and deriving a way to calculate the speed of the glider through the water in cases when all the forces balance, i.e. when the glider does not accelerate any more. They derive from the equations of motion that four physical parameters of the glider need to be determined. They found that these parameters can be determined through a comparison of a flight model with the actual observed flight parameters. With these parameters they calculated the speed the glider should have and compared it to the actual observed one. We have expanded this approach to the calculation of the glider's flight even during times when it is not in a steady state flight. For this we simply integrate the equations of motion.

In this step we determine the 4 parameters of Merckelbach et al. with the help of a non-linear optimization.

Output

The result from this calculation is stored in `deployment_name_dynamics.mat`.

processing_parameters.m

A few physical parameters of the glider need to be set in `processing_params.m`:

`op.glider_volume = 55.2/1000;` This is the volume of the glider in m^^3^^. At GEOMAR different gliders with different volumes are in use:

- 55.2 G1 short deep
- 63.8 G1 long deep
- 61.0 G1 Microrider
- 57.8 G2 short deep
- 72.4 G2 long deep
- 59.3 G2 SUNA
- 50.5 G1 short shallow

`op.frontal_area = 0.038;` This is the frontal area of the glider in m^^2^^. At GEOMAR three different versions are in use:

- 0.038 G1/G2 long and short, shallow and deep regular glider
- 0.057 G1 Microrider
- 0.049 G2 SUNA

`op.effective_wing_area = 0.1;` Wing area of the gliders in m^^2^^. Taken from Merckelbach et al.

`op.air_V = 0;` Air in the oil system could have an effect on the last few meters during the ascent of deep gliders. Here you can enter an estimated air volume in cubic centimeters. If we have the strong suspicion of air in the system we usually set this to 20.

`op.start_dive_parameter = [2,0.5,10];` During the beginning of a dive it is not 100% clear what a glider is doing. Some of this might be attributed to air bubbles sticking between fairings and hull. These bubbles delay the beginning of the dive. These three parameters are used in modelling the delay. For further understanding of the parameters you have to look into the code. The parameters given here seem to work well.

Rarely Used

`op.limit_drag_fit = 3;` In cases of strong biofouling the drag coefficient of the glider is increasing over the time of the deployment. The flight parameter fit shows the increase of this parameter. In most cases a third order fit in time follows the development well. Sometimes this fit does not work well and the order of the increase can be limited to linear. Should the curve in the final plot of step 13 not fit well, set this parameter to 1.

step_14_glider_flight_model.m

Description

Apply the parameters estimated in the previous step to the whole deployment of the glider and derive the speed the glider likely had through the water.

Output

The result is added to the single mat-file named `deployment_name_1sec_derived.mat`.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_15_optimize_temp_for_cond_params_model.m

Description

This step is similar to the calculation of step 11 but uses the glider speeds of the dynamical model to estimate the flow through the conductivity cell.

Output

All results from the optimizations are stored in a mat-file named `deployment_name_model.mat`. Should this step be rerun with different of the above settings this will be stored as additional results. Salinities from the best result will be calculated and stored in `deployment_name_1sec_derived.mat`.

processing_parameters.m

The same parameters as for step 11 are used here.

step_16_pick_model_or_dpdt.m

Description

In steps 11 and 15 two different variations of the salinity calculation were optimized. In this step the results from both are compared and the better one is retained as the optimal salinity derivation.

Output

The best salinity result is stored in `deployment_name_1sec_derived.mat` as `ctd_salinity`. The information which version was used for the calculation is stored in `deployment_name_best_optimizer.mat`.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

Rarely used

`force_tomeu_dpdt_or_model = ' ';` Set this to `'_dpdt'` or `'_model'` to force the use of the respective flow speed. This is only applied in case you have already forced the 4 parameters from Garau et al. (2011).

step_17_correct_optode.m

Description

On all GEOMAR gliders Aanderaa optodes are installed to measure dissolved oxygen. Aanderaa optodes are relatively slow in measuring the oxygen content with internal lags of around 20 seconds. They also use a temperature for the calculation of the oxygen content from their basic optical measurement. This temperature should be the temperature of the thin foil in which the oxygen changes the optical properties. The foil is sitting adjacent to a glass window on the optode. This whole system has a considerable thermal mass and the foil temperature is thus a lagged version of the ambient water temperature. Optodes do their own temperature measurement, but this is not necessarily the foil temperature. In our calculations we use the glider's CTD temperature and try different lags to obtain an optimal oxygen measurement. Additionally we use another set of calibration coefficients to derive oxygen from the optical measurements. In this step we calculate oxygen for about 10 different optode lags and 10 different CTD temperature lags. For all these oxygen values the difference between up and down profile pairs is summed up and the pair of lags is chosen for which the difference sum is minimal. Typical results for the optode lag are 30 seconds and 60 seconds for the CTD temperature lag. This calculation is done both for both the original Aanderaa and, if available, for the GEOMAR-derived optode coefficients. In case the GEOMAR-derived coefficients exist, the best one of these is chosen for the optimal oxygen, in case not, the best one of the Aanderaa values is taken.

In case you did only record the oxygen content and not the underlying optode phase information we do a simpler optimization that only tries to determine the optimal delay of the optode, not the optimal temperature for the calculation.

Output

The results from the optimal calculations are added to the mat-file `deployment_name_1sec_derived.mat`. There you can find several different variables. Depending on what calibration coefficients you have provided some of the following might be missing.

- `oxygen`: These are the original values from the optode.
- `aanderaa_oxygen_undelayed`: A reverse time filter has been applied to the original oxygen data to overcome the slowness of the optode.
- `aanderaa_oxygen_calculated`: A delayed CTD temperature, salinity and pressure have been used to recalculate oxygen based on Aanderaa's coefficients.
- `aanderaa_oxygen_calculated_undelayed`: The reverse time filter has been applied to `aanderaa_oxygen_calculated`.
- `aanderaa_oxygen_undelayed_filtered`: A two-sided filter has been applied to `aanderaa_oxygen_undelayed` to remove the noise introduced with the undelaying. If you provided no Aanderaa foil coefficients, this should be the best Aanderaa based oxygen.
- `aanderaa_oxygen_calculated_undelayed_filtered`: A two-sided time filter has been applied to `aanderaa_oxygen_calculated_undelayed`. If you provided a file with Aanderaa foil coefficients and recorded the B-phase of the optode, this should be the best Aanderaa based oxygen.
- `geomar_oxygen_calculated`: Oxygen values based on a delayed CTD temperature and GEOMAR's own optode calibration coefficients.
- `geomar_oxygen_calculated_undelayed`: Same as the above but a reverse time filter has been applied.
- `geomar_oxygen_calculated_undelayed_filtered`: Same as the above but a two-sided filter has been applied to reduce the noise. If you provided GEOMAR calibration coefficients, this should be the best oxygen values.

The up-down difference results from the optimization and the respective lags are stored in the files `simple_aanderaa_optode_delays.mat`, `aanderaa_optode_delays.mat` and `geomar_optode_delays.mat`.

processing_parameters.m

Three pieces of information can be entered in `processing_params.m`:

`op.o_yo_numbers = [];` Sometimes bio-fouling causes bad measurements by optodes. Here the number of yos which are used for the lag determination can be restricted. The default is to use all yos (empty vector).

`op.file_with_aanderaa_optode_calibration_parameters = 'filename';` A particular m-file has been developed that contains all the calibration parameters present in Aanderaa optodes. The filename without `.m` needs to be entered here. You can leave this empty, if you do not have the calibration parameters. The respective step will then be skipped.

`op.file_with_geomar_optode_calibration_parameters = 'filename';` A mat-file with GEOMAR optode calibration parameters has been developed. The file name without `.mat` needs to be entered here. Should you have no such file, still create this field but with an empty string. You can leave this empty, if you do not have the calibration parameters. The respective step will then be skipped.

step_18_grid_glider_profiles.m

Description

Up to this step the glider data has been stored gridded or referenced in the time domain. Here we average the data of each yo in the vertical over 1dbar boxes. The data is then interpolated linearly onto the full dbar values.

Output

The results are stored in `deployment_name_gridded.mat`.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_19_create_comparison_data.m

Description

As part of the calibration of the T,S, and O sensors one can compare the glider's data to other instruments such as nearby CTD casts or other gliders. To reduce the computational costs this should be done only with a reduced data set. In this step this reduced data set is calculated on certain pressure and density levels and stored in separate files. This is also done both with offsets already applied to T,S, and O and not.

Output

Four files are stored: `deployment_name_comparison_data_on_press.mat` `deployment_name_comparison_data_on_press_with_offsets_applied.mat`

`deployment_name_comparison_data_on_dens.mat` `deployment_name_comparison_data_on_dens_with_offsets_applied.mat` with gridded and reduced T,S,O data sets.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_20_determine_tso_offsets.m

Description

In this step the resulting data of the glider is compared against nearby CTD or other glider data and median offsets are determined.

Please note that steps 18 and 20 need to be rerun when the values for the offsets in `processing_parameters.m` are changed.

This step involves some hand-executed iteration. Should you have a reference data set of nearby CTD profiles, run this step first with all offsets in `processing_parameters.m` set to 0. The choose the offset for salinity from the values shown. Give preference to deep data and data closer in time. Once you have entered this value, rerun steps 18 to 20. Since the salinity influences the density on which the data is compared, this comparison will deliver a different offset for the corrected data. Add the newly determined offset to the one you have in `processing_parameters.m` and again rerun steps 18 to 20. After 2 or 3 rounds the result should be stable. In all deployments that we analyzed at GEOMAR the temperature was now matching too (better than 0.01 degrees) so that we did not enter any offset for temperature. Now pick a similar offset for Aanderaa oxygen and enter it in `processing_parameters.m`. Rerun steps 18 to 20. In this case the density was not changed and you should not need to iterate the whole process. Should you have GEOMAR optode coefficients, you have to enter the respective offset too.

Output

The results will be shown on the screen and a log of the output is stored in `tso_offsets.txt`.

A large number of histogram plots of the deviations between glider and reference data is stored in the folder `plots`.

processing_parameters.m

Four parameters needs to be entered in `processing_params.m`:

`op.glider_reference_data = {'ifm02_depl01','ifm02_depl02'};` Here you can enter a cell list of glider deployments that which the currently processed glider met during its deployment. If no other glider was met, this parameter should be an empty cell array.

```
op.mooring_reference_data =
'../reference_data/glider_mooring_check_data.mat';
```

```
op.ctd_press_reference_data =
'../reference_data/glider_check_data_press.mat';
```
A comparison data set of CTD casts which took place near glider deployments. Here at GEOMAR we simply have combined ALL CTD casts and the routines picks the relevant ones by itself. No slowdown has been observed even though these are about 1000 casts. In the mat-file the variables `np`, `nt`, `ns`, `no`, `nlat`, `nlon`, `ntim` are required. `np` must be a single vertical vector of the pressure levels `[10:10:1000]` (at the moment this selection of levels is preset and not changeable!). `nlat`, `nlon`, and `ntim` must be (horizontal) vectors with one value for each CTD cast. The units of `ntim` must be matlab datenum. `nt`, `ns`, and `no` must be calibrated in situ temperature, practical salinity, and dissolved oxygen concentration (in mumol/kg), respectively.

`op.ctd_dens_reference_data = '../reference_data/glider_check_data_dens.mat;'` This file has to contain the same information as the one for pressure but has been calculated on the fixed in situ density levels `[ [1022:0.05:1025],[1025.01:0.01:1028] ]`. `np` in this case is also a full array.

Rarely Used

`op.max_ctd_comp_dist = 0.2;` This is the maximum distance between glider and CTD profiles to be used in the comparison. The units are degrees.

step_21_finalize_data.m

Description

In this step the offsets from step 20 are applied and the best version of the scientific variables is stored under non-complicated names.

The offsets determined in step 20 or obtained from other information must correctly have been entered in `processing_params.m`.

Output

The gridded results will be saved `deployment_name_final_gridded.mat` and the 1 second interpolated results will be saved in `deployment_name_final_1sec.mat`.

processing_parameters.m

`op.t_offset = 0;` Temperature offset that will be **subtracted** from the CTD's temperature data.

`op.s_offset = 0;` Salinity offset that will be **subtracted** from the CTD's salinity (either derived via dpdt or via model conductivity cell flow speeds).

`op.use_c_factor = 0;` Set this to `1` to use a correction factor for conductivity instead of an offset for salinity. The correction factor is determined from the conductivity difference between applied and not applied salinity offset. This should be used in cases of large observed salinity ranges. E.g. in the Mediterranean Sea where salinity can range from 35 to 38.5 PSU, or in the Baltic Sea where the range is even wider. In most open ocean conditions this can stay at `0`.

`op.oa_offset = 0;` Disolved oxygen offset that will be **subtracted** from the optodes oxygen values (oxygen calculated using Aanderaa coefficients).

`op.oa_factor = 1;` Disolved oxygen factor that will be **multiplied** to the optodes oxygen values (oxygen calculated using Aanderaa coefficients). The offset is applied after this multiplication.

`op.og_offset = 0;` Disolved oxygen offset that will be **subtracted** from the optodes oxygen values (oxygen calculated using GEOMAR coefficients).

`op.og_factor = 1;` Disolved oxygen factor that will be **multiplied** to the optodes oxygen values (oxygen calculated using GEOMAR coefficients). The offset is applied after this multiplication.

`op.og_pfactor = 0;` Disolved oxygen factor that will be **multiplied** to the optodes oxygen values (oxygen calculated using GEOMAR coefficients). The factor can add a pressure dependent term to the oxygen correction. It is very rarely used.

step_22_summarize_deployment.m

Description

Various statistical information can be collected from the processed and unprocessed data.

Output

The result is shown on the screen.

processing_parameters.m

No information needs to be entered in `processing_parameters.m`.

step_23_data_for_others.m

Description

Some external instruments such as Microriders require for their processing a subset of the final glider data. Here this data is collected and stored in dedicated files.

Output

At the moment one file `deployment_name_for_mr_processing.mat`.

The End.