FastCTD Processing Notes: A rewrite of Rob's FastCTD Processing

Alexander Andriatis

24th March 2023

1 Introduction

The purpose of the following undertaking is to make Rob's FastCTD processing scripts, namely the de-hysteresis and salinity despiking routines, broadly compatible across FastCTD deployments without too much back-end adaptation. The desired end result is a routine that converts standard FastCTD output files into a depth-gridded, qc'd, de-hysteresised, and salinity-despiked product. Imporantly, it must be better quality than the existing on-board processing, San's 'FastCTD_GridData.m', and easier to use than Rob's set of routines, 'I_2016_D_deHysteresisFLEAT2016.m' and 'K_2016C_FCTD_Response_Fleat.m'.

This writeup will walk thorugh my package 'FCTD_DehystDespikingGridding_AAA' and compare the resulting data vs. unprocessed FastCTD data and the output from San's gridding routine.

2 FastCTD Processing

The top-level script is 'FCTD_DehystDespikingGridding_AAA.m'. It walks through the processing of FastCTD data step-by-step, saving the intermediate outputs between steps as it goes.

2.1 Inputs

FastCTD data, after being converted to '.mat' format, is stored as a large collection of 2MB .mat files with timestamps in a folder called '/MAT'. These files are the start point of this procesing routine. Set the path to this '/MAT' folder in the variable 'datapath'.

As the processing runs it saves intermediate files between steps. Set the top-level folder where these should be saved as 'processpath'.

I use some of San's processing functions from the FastCTD processing code. Add the latest version of these to the matlab path.

If working in a different folder than the processing code, add the processing code to the matlab path.

During the processing script some plots are generated as diagnostics and are useful to save. Set a location for these as 'figpath'. I recommend a 'figure' folder within 'processpath'.

To give the output files from a particular run of this processing code a unique name, set a name as 'experiment'. Something needs to be put here otherwise filenames will start with an underscore.

2.2 Merge all FastCTD data into one big timeseries

To begin the processing code, the multitude of FastCTD .mat files need to be merged into one long timeseries. This is done using San's 'FastCTD_MergeFCTD.m'. The resulting structure 'FCTDfull' will have the essential variables of time, pressure, temperature, and conductivity a long timeseries. The structure also has many useful auxiliary fields such as acceleration and compas data, winch and GPS information, and additional payload data such as microconductivity. The variables that are the same length as the time vector are carried through the processing through gridding. Other information such as the header is left behind at this step, except for knowledge about the serial number, so if you need to refer back to these, look at the output of this step of the code, saved in 'processpath / Profiles / (experiment)_FCTD_profiles.mat'.

It is important that at this stage that there is a variable called 'header' with a field called 'serialno' which contains the serial number of the ctd for each of the fastCTD files. If there's no header or serial number record, one will need to be added in manually.

2.3 Find pairs of profiles

The next step is to find paris of up-down profiles using 'FCTD_FindProfiles_AAA.m'. This is a replacement of Rob's 'H_2016_C_FCTDreader.m' and uses San's 'FastCTD_FindCasts.m'. The code makes sure that the profiles are at least 10m long and that the up and down sections overlap in pressure. It also requires profiles to be separated by more than 30s. The output of this code is the structure 'FCTDprofiles' which contains fields 'up', 'down', and 'header'. Each direction field contains the data variables as cell arays, with each cell representing one profile in the given direction. The header field contains a vector of serial numbers, one for each profile.

'FCTDprofiles = FCTD_FindProfiles_AAA(FCTDfull, savepath)' requires input variables 'FCTDfull' which is a structure containing the timeseries of variables, and 'savepath' which is the path to save the output structure of profiles 'FCTDprofiles'.

2.4 QC the profiles

To apply some perliminary quality control, the profiles are passed through a number of criteria. Along the way, 'bad' sections of profile are cut out if they are out of spec. If the profile survives all the checks, it is assigned a value of 0 in the field FCTDprofiles_qc.header.qc, else it is assigned a 1. Only profiles with a 0 qc field are used to determine the hysteresis and salinity correction parameters, but all the profiles are ultimately corrected. The sections that have been cut out, however, are forever ignored. This might be undesirable however as we may want to keep the entirety of each profile, in which case I could calculate parameters from good profiles but correct the entirety of all the profiles.

The criteria to meet are as follows:

1. The vertical velocity is at most 10 m/s, to avoid anomalous pressure spikes.
2. The vertical velocity is at least 0.1 m/s, to avoid sections where the fish is just hanging out at the bottom or the surface.
3. The vertical temperature graident is at most 5 degrees per meter, to avoid false CTD readings.
4. The vertical conductivity gradient is is at most 0.5 S/m per meter, to avoid false CTD readings.

Data is cut from the first half of the profile until only data that meets all the criteria remains, and the second half of the profile is kept until the first point where the criteria cease to be met.

If the profiles are then checked manually and additional profiles need to be removed from the processing, set bad profiles to qc=2 to indicate that they have been removed by hand instead of programatically. I recommend sitting down and plotting each pair of profiles, including a rough salinity and density calculation, to look for spikiness and other wierdness, since it always manages to sneak through the processing.

'FCTDprofiles_qc = FCTD_CleanProfiles_AAA(FCTDprofiles, savepath)' requires input variables 'FCTDprofiles' which is a structure containing the cell arrays of variables in each direction, and 'savepath', which is the path to save the output structure of qc'd profiles 'FCTDprofiles_qc'. The output structure carries forward only the time, temperature, conductivity, and pressure fields, since they will be acted on by the rest of the code. The auxilary variables of interest will rejoin the corrected temperature and conductivity at the gridding stage.

2.5 Correct hysteresis

The first substantial processing step is correcting the hystereis associated with the response between oceanic feature and sensor behavior. This hysteresis results in a missmatch between the up and down profiles of the same column of ocean. The hysteresis correction function 'FCTD_deHysteresis_AAA' is a rewrite of Rob's 'I_2016_D_deHysteresisFLEAT2016.m'. The principal improvements are that the hysteresis correction is performed on a per-serial number basis since different CTDs have sensors with different responses, and that the parameters for the hysteresis correction are automatically optimized to minimize the up-down mismatch.

'[FCTDdehysteresis,Parameters] = FCTD_deHysteresis_AAA(FCTDprofiles_qc, savepath)' requires input variables 'FCTDprofiles_qc' wihch contains the qc'd profiles, and 'savepath', which is the path to save the output structure of post-dehysteresis profiles 'FCTDdehysteresis'. The output structure 'Parameters' contains the dehysteresis parameters for each serial number CTD. It is recommended to save this variable in the processing folder to avoid re-running the parameter optimization code if the processing needs to be repeated. The script can be run with optional parameters '[FCTDdehysteresis,Parameters] = FCTD_deHysteresis_AAA(FCTDprofiles_qc, savepath, 'parameters', Parameters, 'figpath', figpath, 'plotevery', 1000)' where 'Parameters' is the structure containing the dehysteresis parameters from a previous run of this code, 'figpath' is the top-level folder where diagnostic figures are saved to, and 'plotevery' is an integer n to plot the profiles of every n profiles in the data.

The dehysteresis code begins by correcting for the offset between the pressure and temperature sensors on the fish. The code applies a time offset to correct the temperature, based on the difference in position between the temperature and pressure cells:

$$T^{\star}(n) = T(n + \text{offset\_T}) \tag{1}$$

where n is an index of the timeseries and offset is an integer. For the data I have been working with a typical value is offset = 1. Does this make sense? If the fall speed is about 2m/s and the sampling is at 16hz, then one index equals a shift of 12 cm, a reasonable estimate of the vertical distance between the temperature and pressure sensors. The vertical distance between the sensors depends on the vertical velocity of the fish and is different going up vs down. A faster fall (greater attack angle), such as at the top of the profile, will result in a larger pressure difference than towards the slow bottom of the profile, resulting in a warm bias near the top and a cold bias near the bottom for a constant offset. Similarly, if the angle of attack is steeper on the way down than the way up, the correction will apply a warm bias on the way down, and a cold bias on the way up. A better way to make this correction would require knowing the distance between the temperature and pressure cells and the instantaneous angle of attack of the fish. The optimal offset_T is chosen as follows:

1. For each profile, load the pressure and temperature
2. Grid the up and down profiles on the same pressure grid
3. Compute the mean square error between up and down
4. Try a variety of offset parameters from 0 to 4. 4 is probably enough. 4 = .25seconds * 4m/s = 1m for 16Hz sampling.
5. Pick the offset that results in the least error
6. Correct the temperature by shifting the temperature the index by the desired integer offset.

A plot of the error vs. the choice of parameter offset_T is shown in Figure 1. It is worth noting that this optimization is done across all serial numbers once because the ctds are built the same and have the same relation between the pressure and temperature sensors.

Next, the temperature hysteresis correction is applied. Hysteresis occurs because temperature gradients in the ocean are step-like, but the temperature sensor has a non-instantaneous response. On the way down, on encountering a step, the sensor will see a smoother temperature profile, with a warmer average

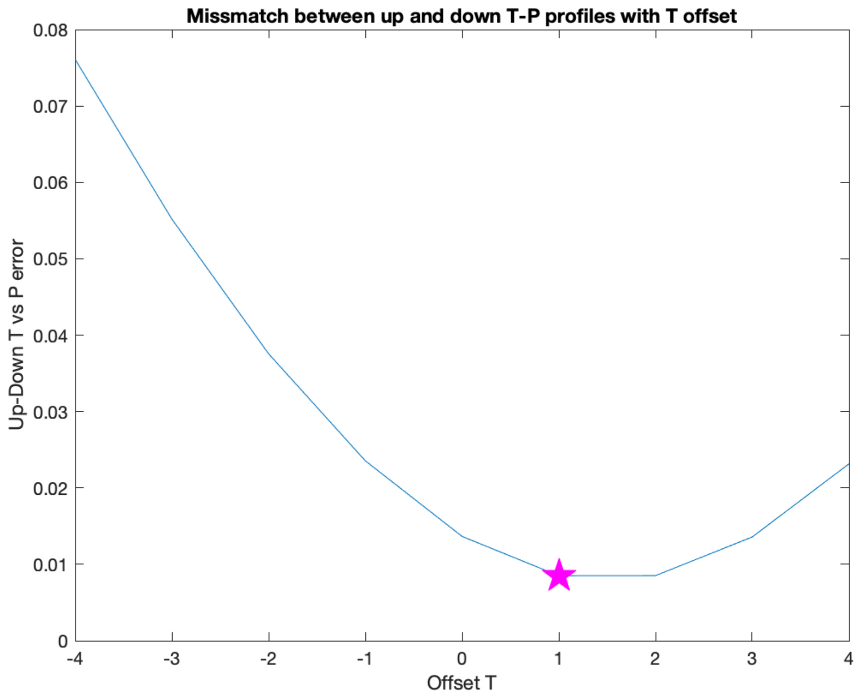

Figure 1: Tuning the parameter offset_T to minimize the difference between up and down profiles in T vs P space.

temperature and with the middle of a step feature appearing deeper. On the way up, the sensor will see a colder average temperatrue, with a shallower middle of a step feature. To correct for this effect, the temperature is sharpened using a "lag-lead" filter given by Lueck and Picklo [1] in their equaiton 3:

$$\hat{T}(n) = (1 - a_0 - a_1)\hat{T}(n-1) + a_0 T(n) + a_1 T(n-1),$$

where $\hat{T}$ is the corrected temperature, T is the observed temperature, and a_0, a_1 are parameters to be tuned. A good correction will result in temperature gradients dT/dz that are the same for a given temperatrue T in the up and down casts.

The temperature hysteresis correction used by Rob is slightly different, but apparently achieves the same effect. Rob's temperature correction, which I implement in 'deHyst_T.m' is

$$\hat{T}(n) = T(n) + \tau(\hat{T}(n) - T(n-L)), \tag{2}$$

where τ and L are parameters to be tuned, and in my code I use $T^{\star}$, the temperature after the first offset correction. To pick the optimal parameters, I sample a range of possible values for τ and L, computing for each combination the average mean square error of the difference between the up and down profiles in dT/dz vs T space. The parameter space is sampled twice, once for a broad range of values and the second time around a narrow band based on the optimal values from the first pass. Ideally, this parameter optimization would instead be done using a clever gradient descent algorithm, but I had difficulty getting this to work because of the poorly-conditioned parameter space and the need to use discrete values of L for the index lag. I should also reconsider using mean squared error as the cost function. Perhaps something like the correlation between the two directions would be a better metric for how well the two directions match. A plot of the parameter space is shown in Figure 2. This optimization is done once for each serial number CTD, and the optimal parameters are saved to a Parameter structure. Once parameters τ and L are chosen they are used to correct the temperature according to 2. The effect of the temperature hysteresis correction is shown in Figure 3.

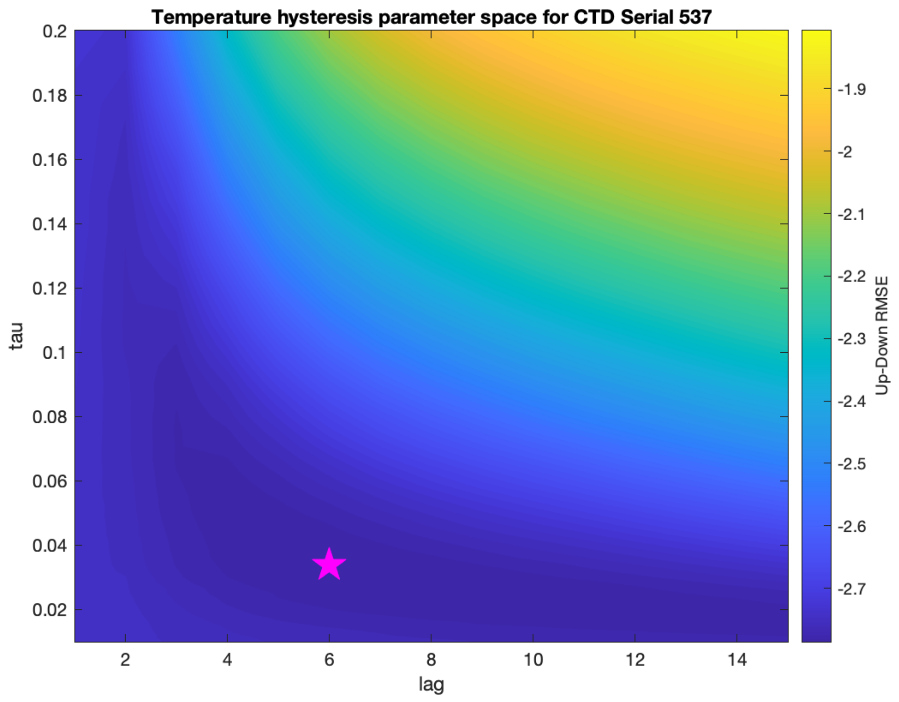

Figure 2: Tuning the parameters τ and L to minimize the difference between up and down profiles in dT/dz vs T space. The minimum error is indicated with a star.

Next, the offset between the pressure and conductivity cells on the fish is corrected using the same method as the temperature offset in equation 1 above,

$$C^{\star}(n) = C(n + \text{offset\_C}) \tag{3}$$

As it turns out, the optimal offset_C is usually zero, either because the conductivity and pressure cells are closer together or because this correction is already applied onboard the CTD.

Now that we have temperature and conductivity referenced to the same pressure, and the hysteresis correction has been applied to temperature, we can perform a conductivity hysteresis correction. Lueck and Picklo [1] use a time-domain conductivity correction,

$$C'(n) = -bC'(n-1) + \gamma a[\hat{T}(n) - \hat{T}(n-1)], \tag{4}$$

$$\text{with } a = \frac{4 f_{Ny} \alpha \beta^{-1}}{1 + 4 f_{Ny} \beta^{-1}} \tag{5}$$

$$\text{and } b = 1 - 2a/\alpha, \tag{6}$$

where C' is the conductivity error, f_{Ny} is the Nyquist frequency, and α, β, and γ are tuneable parameters. The same conductivity correction is used by Rob and in this code, with a slight difference that the temperature difference is taken between temperatures that are L indices apart,

$$C'(n) = -bC'(n-1) + \gamma a[\hat{T}(n) - \hat{T}(n-L)]/L. \tag{7}$$

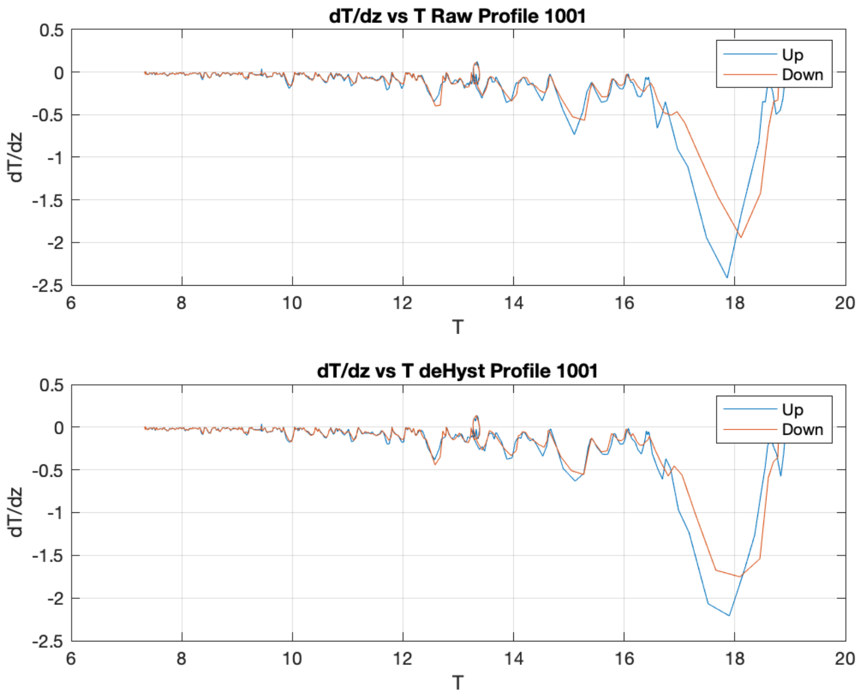

Figure 3: The effect of the temperature hysteresis correction in dTdz vs T space. You'll need to squint - the difference is small.

The conductivity is then corrected using

$$\hat{C} = C(n) + C'(n). \tag{8}$$

The free parameters are chosen using the same parameter space sampling as for the temperature. The cost function being minimized is the up-down missmatch in dC/dz vs C space. Again, I would have prefered to use gradient decent to find parameters that minimize the cost function instead of brute-forcing a scan over the parameter space, but the parameter space and cost function proved to be poorly suited for a quick implimentation. Also, I should reconsider whether the mean squared error of the up-down missmatch is the best quantity for a cost function. Now there are three free dimensions, so Figure 4 shows the parameter space for two dimensions at a time, keeping the third constant, for a particular CTD. Using these parameters the hysteresis correction in equation 7 is applied to the data. The effect of the conductivity correction is shown in Figure 5.

The end of the script plots the corrected profiles in T-P, C-P, and T-C space. The effect of the hysteresis corrections is particularly visible in the T-C space plots. Figure 6 shows the correction on a particular profile.

2.6 Correct Salinity Spiking

Salinity spiking results from the difference between the response of the temprature and conductivity cells to changes in temperature. The conductivity cell has a higher frequency response than the temperature sensor, so when salinity is calculated from temperature and conductivity, the "wrong" temperature is used for a particular conductivity, leading to anomalous salinty values. The relation between the measured and true

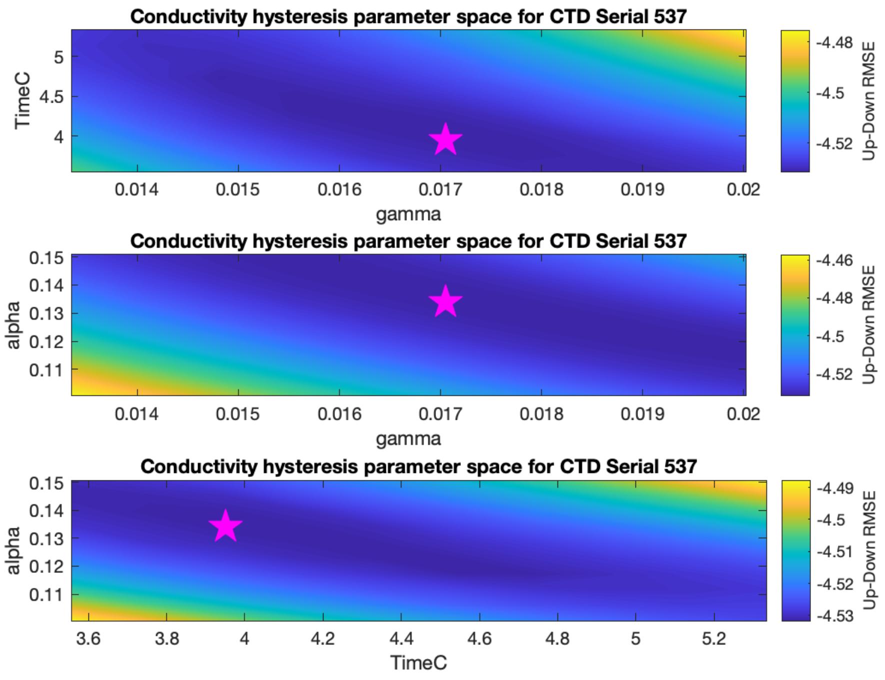

Figure 4: The parameter space for α, TimeC $= \beta^{-1}$, and γ. The parameter combination giving the lowest mismatch is indicated by the star.

quantities is given by

$$\hat{T}(f) = H_T(f)T(f), \tag{9}$$

$$\hat{C}(f) = H_C(f)T(f), \tag{10}$$

where $\hat{T}(f)$ and $\hat{C}(f)$ are the observed frequency-space temperature and conductivities, $T(f)$ is the true temperature, and $H_T(f)$ and $H_C(f)$ are the frequency response functions of the temperature and conductivity sensors respectively. We make the assumtion here that salinity is constant, so that the response of the conductivity cell depends only on temperature. Substituting the response functions into the spectra and cross-spectra of temperature and conductivity gives

$$\frac{\hat{S}_{TC}(f)}{\hat{S}_{TT}(f)} = \frac{H_C(f)}{H_T(f)}, \tag{11}$$

$$\frac{\hat{S}_{TC}(f)}{\hat{S}_{CC}(f)} = \frac{H_T(f)}{H_C(f)}. \tag{12}$$

Because the conductivity cell has a faster response, we'd like to correct the temperature sensor to have the same response as the conductivity cell

$$\hat{\hat{T}}(f) = \alpha\hat{C}(f) = \alpha H_C(f)T(f). \tag{13}$$

The corrected temperature can also be expressed as a transfer function applied to the observed temperature

$$\hat{\hat{T}}(f) = \hat{T}(f)P(f) = H_T(f)T(f)P(f). \tag{14}$$

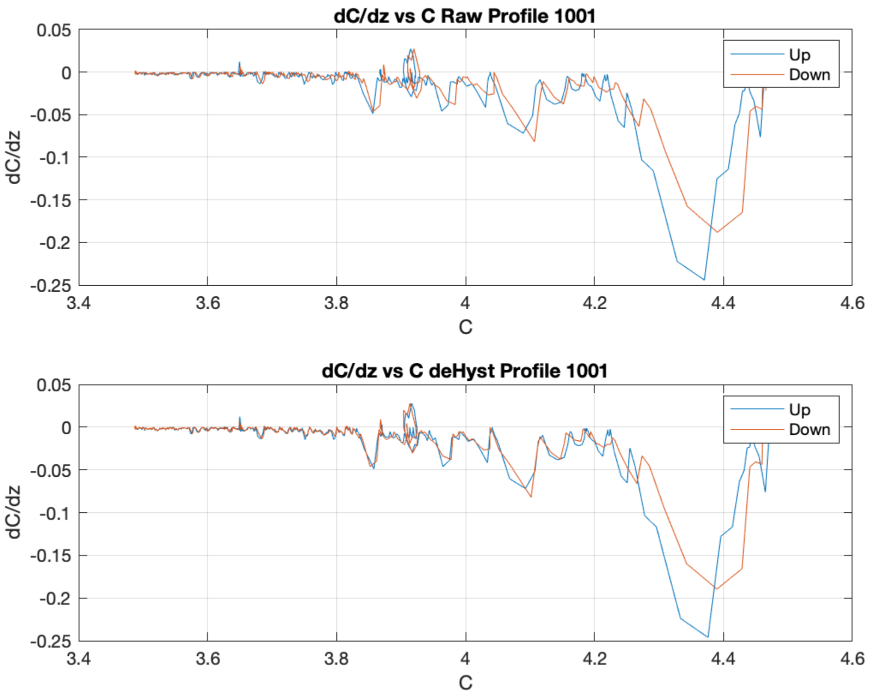

Figure 5: The effect of the conductivity hysteresis correction in dCdz vs C space for a single profile.

Equating these expressions for the corrected temperature and substituting 11 gives

$$P(f) = \alpha\frac{H_C(f)}{H_T(f)} = \alpha\frac{\hat{S}_{TC}(f)}{\hat{S}_{TT}(f)}, \tag{15}$$

so that the corrected temperature can be expressed in terms of the measured quantities $\hat{(T)}$ and $\hat{(C)}$. The transfer function P(f) consists of a real-valued gain $|P(f)|$ and a phase shift $e^{i\phi(f)}$. In practice, the response function is calculated from the fourier transforms of the *gradient* of the temperature and conductivity time-series since we don't care about low-frequency changes in the water column, and want to pre-witten the spectra to avoid spectral leakage.

The function 'FCTD_responsematch_AAA.m' performs this temperature response correction, and is a rewrite of Rob's 'K_2016C_FCTD_Response_Fleat.m'. The principal improvement is that a unique transfer function is created for each serial number CTD. Calling the function 'FCTD_responsematched = FCTD_responsematch_AAA(FCTDprofiles_qc, FCTDdehysteresis, savepath, 'figpath', figpath, 'plotevery', 100, 'depthrange', [200 300])' requires the qc'd profiles FCTDprofiles_qc, the profiles after the hysteresis correction, FCTDdehysteresis, and the path to save the response matched profile structure. The both pre- and post-hysteresis profiles are used in order to compare the combined effects of response matching and hysteresis correction with just response matching. This functionality can be removed in future code versions to streamline the process. Optional inputs are 'figpath' to save diagnostic fiugres, 'plotevery' to make diagnostic plots of every n profiles, and 'depthrange', the depth range over which to compute the response function. It is important to choose a depth range over which changes in conductivity are temperature driven - a good starting point is to set the range below the pycnocline. The output structure, FCTD_responsematched, will have one field for the response correction after dehysteresis and another for the response correction without dehysteresis. This has been done mainly for diagnostic purposes and can be removed in future versions.

'FCTD_responsematch_AAA' begins by taking the fourier transforms of the first difference of temper-

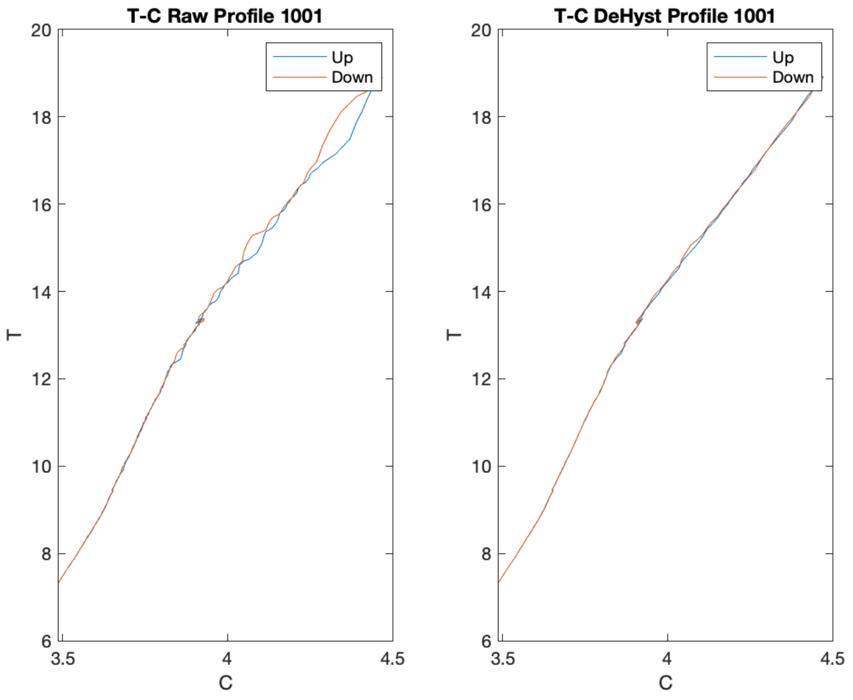

Figure 6: The effect of the temperature and conductivity hysteresis corrections in T-C space.

ature and conductivity

$$\hat{T}(f) = \text{FFT}\left(\frac{\partial \hat{T}(t)}{\partial t}\right), \tag{16}$$

$$\hat{C}(f) = \text{FFT}\left(\frac{\partial \hat{C}(t)}{\partial t}\right), \tag{17}$$

where $\hat{T}(t)$ and $\hat{C}(t)$ are the observed (hysteresis-corrected) temperature and conductivity timeseries. If a depth range is specified, the timeseries are constrained to pressures within the desired depths. The spectrum and cross-spectrum of temperature and conductivity,

$$\hat{S}_{TT} = \left|\hat{T}(f)\right|^2, \tag{18}$$

$$\hat{S}_{CC} = \left|\hat{C}(f)\right|^2, \tag{19}$$

$$\hat{S}_{TC} = \hat{T}^*(f)\hat{C}(f), \tag{20}$$

are saved for each profile. For each serial number CTD, the spectra from all the profiles of that CTD are averaged. Figure 7 shows the mean temperatrue and conductivity spectra for a particular CTD, in each direction, for both pre and post-dehysteresis timeseries.

From the average spectra I compute the gain of the transfer function

$$|P(f)| = \frac{\left|\hat{S}_{TC}\right|}{\hat{S}_{TT}}, \tag{21}$$

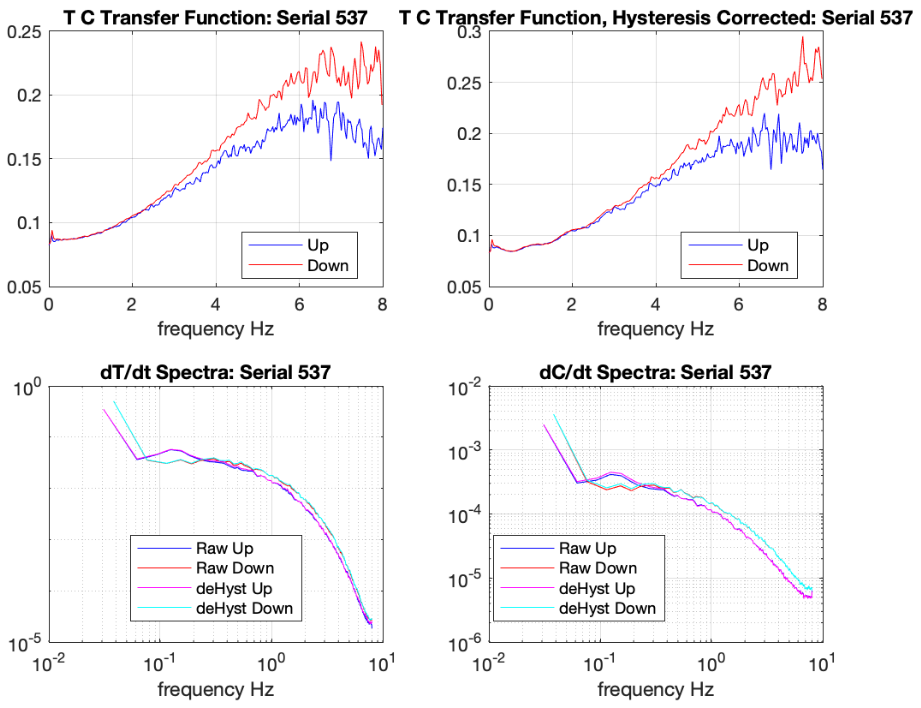

Figure 7: The average spectra $\hat{S}_{TT}$ and $\hat{S}_{CC}$ as well as the transfer function $P(f)$, for one CTD.

and the angle

$$\phi(f) = \tan^{-1}\left(\frac{\mathrm{Im}(\hat{S}_{TC})}{\mathrm{Re}(\hat{S}_{TC})}\right). \tag{22}$$

I then fit a 20th-order polynomial to the gain and phase as functions of frequency to give a smoothed function, requiring that gain is an even function of frequency and phase is an odd function of frequency. The polynomial fits to the average gain and phase for the de-hysteresis down profiles of CTD 537 are shown in Figure 8. Figure 9 shows the polynomial fits to the transfer function for both up and down profiles and both the pre- and post-dehysteresis profiles for CTD 537. Notice how the large linear trend across frequencies in the phase has been removed by the hysteresis correction.

Now we can correct the temperature frequency response to behave like the conductivity, using the whole profile instead of a specified depth range. For each profile, I take the fourier transform of the first derivative of temperature, as in equation 17. I evaluate the polynomial fits for gain and phase from the appropriate serial number, profile direction, and profile type at the frequencies of the fourier transform. The temperature is then corrected by multiplying the profile by the gain and phase,

$$\hat{\hat{T}}(f) = \frac{|P(f)|}{P(0)}e^{i\phi(f)}. \tag{23}$$

The gain has been scaled so that the mean temperature gradient is unchanged. The corrected temperature frequency series is then inverse fourier transformed and re-integrated to give the corrected temperature timeseries.

$$\hat{\hat{T}}(t) = \hat{T}(1) + \int_0^t \mathrm{invFFT}(\hat{\hat{T}}(f))\,\mathrm{d}t\,. \tag{24}$$

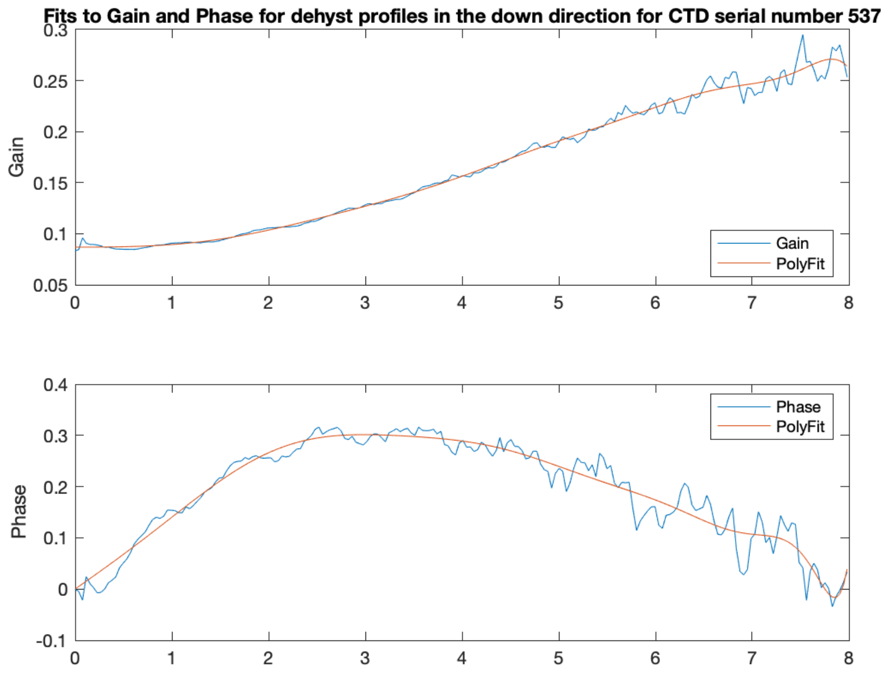

Figure 8: The gain and phase of $P(f)$ for post-dehysteresis profiles in the down direction from serial no. 537 are shown along with their 20th order polynomial fits.

Appying the correction to the temperature gradient deviates from Rob's code, which corrects the temperature. I think the two methods have pretty much the same effect, since the gain and phase adjustments to the low frequencies are small anyway, although I'm still not clear on why they're relatively interchangeable. An advantage of correcting the temperature gradient is that the endpoints of the timeseries are close to zero, so they don't introduce large wrap-around effects in Fourier space, which then create problematic end-points in the time domain after being phase-shifted. Becasue the end points don't go to zero, however, I keep Rob's method of cutting off the "transient" endpoints. The size of this transient is found by converting the phase shift to an indice shift and taking the maximum of this indice shift as the number of points to cut off. Instead of performing the cutting in the response matching code, information about this "transient" is carried through until the gridding so that the end points can be uniformly cut off from all the data.

One potential problem with the temperature gradient correction is that the re-integration starts from the uncorrected first point in the temperature profile. Any random error associated with that point is propagated through the entire profile. Rob suggested that to test the effect of this random error, I plot the spectrum of the timeseries of depth of an isopycnal. The random error in the initial temperature should introduce variance to the isopycnal depth. Figure 10 shows the spectra of the depth of the $\sigma = 26$ isopyncal from both the temperature correction and the temperature gradient correction. There seems to be no significant difference between the two response matching methods.

The final step in the response matching code is to low-pass filter the data. A low-pass filter adresses the issue that beyond a certain frequency, the observed conductivity stops being coherent with the observed temperature. At high frequencies, therefore, the temperature correction is meaningless and creates flase data. The correlation between temperature and conductivity is plotted in Figure . The coherence drops below 0.9 at around 3 Hz. A zero-phase 4th order lowpass Butterworth filter with a cutoff of 1/3 the Nyquist frequency (2.6 Hz for 16 Hz sampling)is applied to the corrected temperature as well as the conductivity

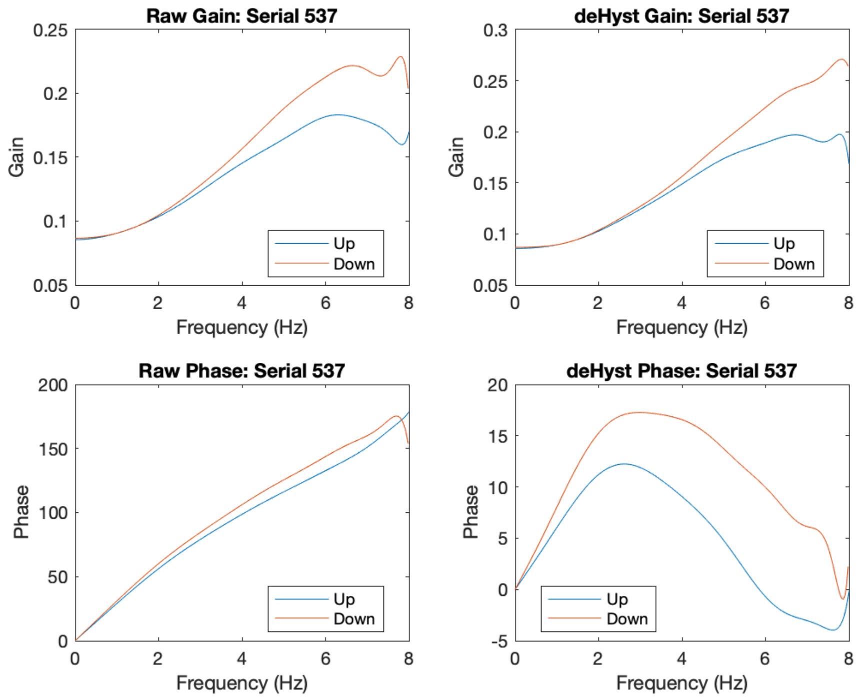

Figure 9: The gain and phase of $P(f)$ for post-dehysteresis profiles in both directions and for both pre- and post-dehysteresis profiles for serial no. 537.

and pressure timeseries. It is worth noting that auxiliary variables are not low-pass filtered before gridding. This would need to be done by the user or can be built into a future version of the code.

After saving the output 'FCTDmatched' structure, some diagnostic plots are made showing the effect of response matching. Figure 11shows, from left to right, the T-C plot of up and down profiles (1) before any corrections, (2) after just the hysteresis correction, (3) after just the response-matching correction, (4) after both hysteresis and reponse-matching. The combination of hysteresis and response matching corrections is most effective at removing the missmatch between up and down profiles.

2.7 Checking the Respone Matching

The perfomance of the response matching correction can be verified by generating plots using 'FCTD_responsecheck_AAA', which plots the transfer function between the conductivity and the corrected temperature, and 'FCTD_SalinityCheck_AAA', which compares the salinity calculated from temperature and conductivity after various steps in the processing.

Figure 12 shows the mean spectra of the temperature and conductivity gradients. The cutoff of the low-pass filter is seen in the corrected spectra. Figure also shows the gain of the transfer function of the temperature correction. The corrected temperature requires no further correction, as the gain is flat, until the low-pass cutoff. Figure 13 shows that the phase of the transfer function is also flat, meaning that the temperature correction has been correctly phase-shifted.

Figure shows the salinity for a single pair of profiles at different stages in the processing. The raw data (1) shows both an up-down mismatch and salinity spiking. The hysteresis-only correction (2) brings the profiles together and somewhat reduces spikiness. The response-only correction (3) removes the salinity spiking but leaves an up-down hyteresis effect. Performing both a hysteresis and response-matching correction (4) results

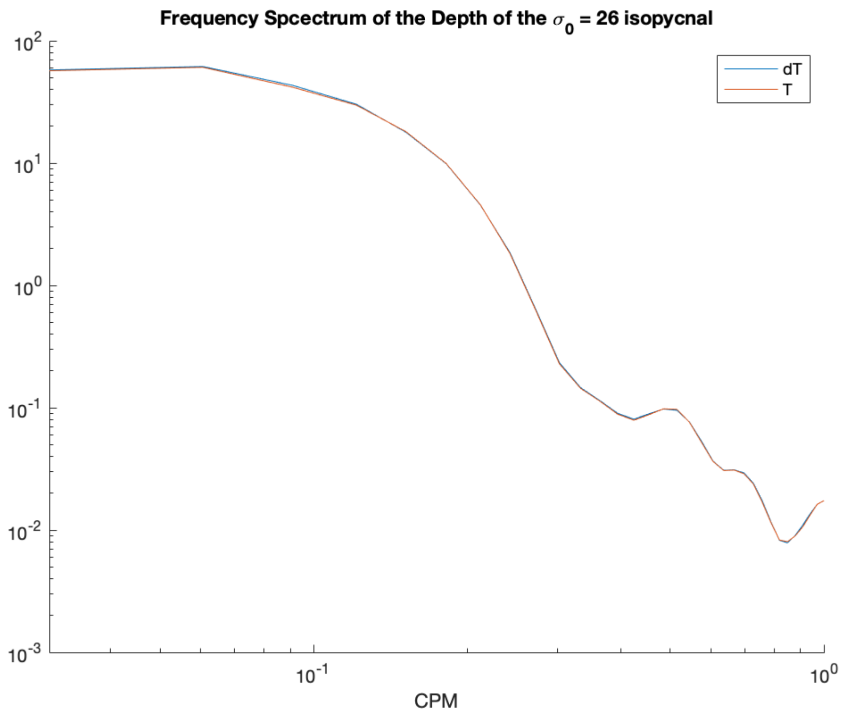

Figure 10: Spectrum of the depth of the $\sigma = 26$ isopycnal using two different methods for temperature response correction.

in a pair of well-behaved matching profiles.

2.8 Corrected Profiles

Now that we've convinced ourselves that the de-hysteresis and response-matching codes performed as desired, we can combine the corrected variables with the rest of the FastCTD data. This happens in 'FCTD_CorrectedProfiles_AAA' which has as inputs FCTDprofiles, the raw data containing pairs of profiles, and FCTDmatched, containing the corrected temperature, conductivity, and pressure. These data are combined into one structure. Then the endpoints of the profiles containing false data are cut off. There are two sources of false data: in the de-hysteresis code, the temperature and conductivity corrections relied on a lagged temperatrue difference, and on a recursive conductivity correction, therefore the first $n = L + 1$ points in the timeseries are bad. The second source is the transient arrising from the phase shift of non-periodic timeseries. Since I've kept all the data so far, the number of points to be cut is the maximum of the two effects. One could consider that I shouldn't be keeping the bad data from the hysteresis correction when applying the response correction. In this case the number of points to cut would be $n = L + 1 + \text{transient}$. This can be added to a future version.

The output structure 'FCTDcorrected' contains 'up' and 'down' fields that contain variables in cell array format, with each cell containing one vertical profile. This is the final form of the data before gridding. The code then plots the T-P, C-P, and T-C plots of up and down for raw and corrected data for validation. Figure 15 shows an example of the corrected and raw data for one pair of profiles.

2.9 Gridding

It is best practice to calculate the salinity and density before gridding in depth, to avoid reintroducing salinity spiking that we tried so hard to remove. I calculate the using the Gibbs Seawater Toolbox, which

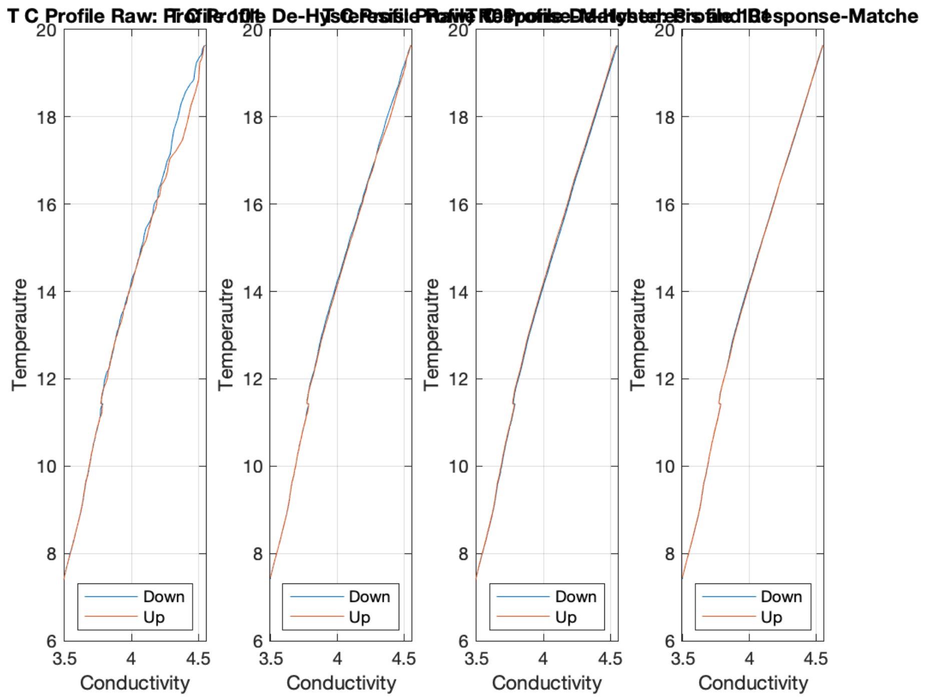

Figure 11: From left to right, the T-C plot of up and down profiles (1) before any corrections, (2) after just the hysteresis correction, (3) after just the response-matching correction, (4) after both hysteresis and reponse-matching.

requires latitude and longitude for the salinity calculation. Therefore I need to add position information to the data before proceeding with gridding. My processing requires a structure 'GPS' that has latitude and longitude as decimal degrees in one long timeseries, along with their respective time. This could be made from the 'GPS' field that comes from the FastCTD .mat files in San's 'FastCTD_MergeFCTD.m', or if those are blank can be made from the ship met data, accomplished easiest using San's 'SN_readShipMET.m'. The GPS structure is combined with the profile data in 'FCTD_AddPosition_AAA.m'. If no position data is provided and this step is skipped, I use default values of 35° N, 0° E.

Depth gridding is performed in 'FCTD_grid_AAA.m'. The depth interval is set to a default of 0.5m and can be specified using the optional parameter 'dz'. The depth grid is assigned to range from the minimum to maximum depths in the data. Before gridding, the salinity, in-situ density, and potential density with respect to the surface are calculated using gsw. I suppose I could do this in a separate code and add these to the final set of profiles for convenience. I might do this when I clean up the code.

For each direction, for each variable, for each profile, the data are binned in depth using rectangular bins. I briefly experimented with some clever windowed binning methods to better represent the center of the bins but I quickly found the marginal improvements in accuracy to be not worth the increased complexity and computation time. Bins without data are filled with linear interpolation, which migh happen at the top of a downward profile if the depth bins are too small for the fall speed. The depth grid and mean time for each profile are added to the structure of gridded variables, as well as the header containing the serial numbers and qc flag, and an info field explaining each variable. The output 'FCTD_gridded' is saved in 'savepath'.

Up to this point, the 'up' and 'down' profiles have been kept separate for processing and easy validation. Since we have now corrected the hysteresis, we can combine them into one depth-gridded structure using 'FCTD_combine_grid_AAA.' This is the format in which I recommend distributing and doing science with

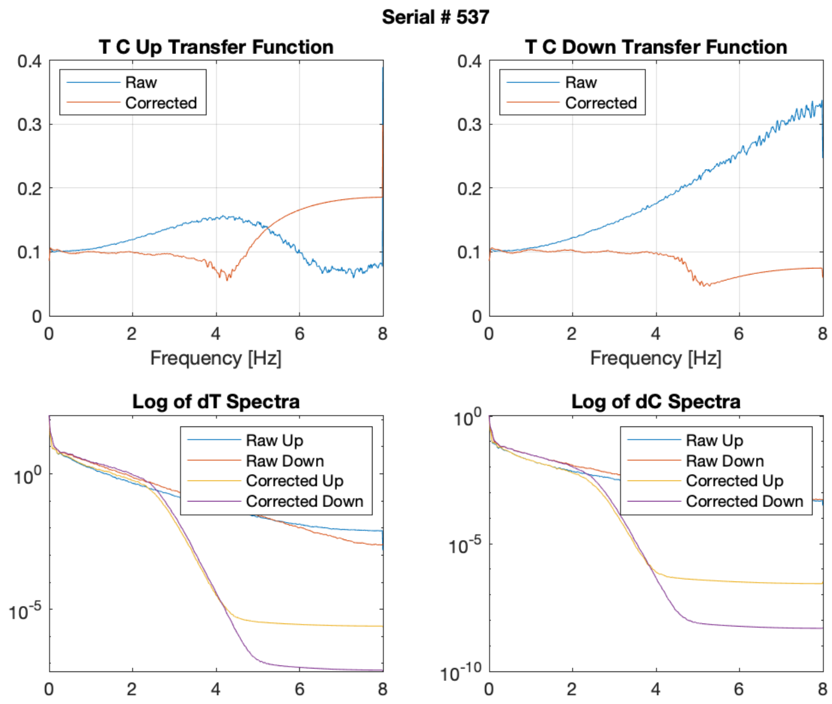

Figure 12: Frequency spectra and transfer function of the corrected temperature and conductivity vs the raw data.

the FastCTD data.

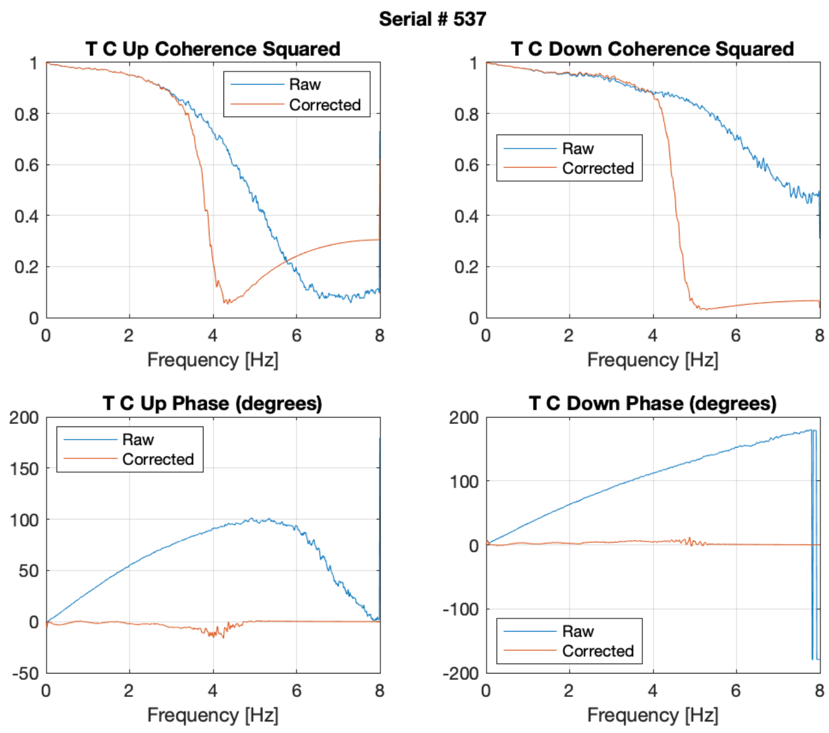

Figure 13: The average coherence of the corrected temperature and conductivity vs the raw data, along with the phase of the transfer function.

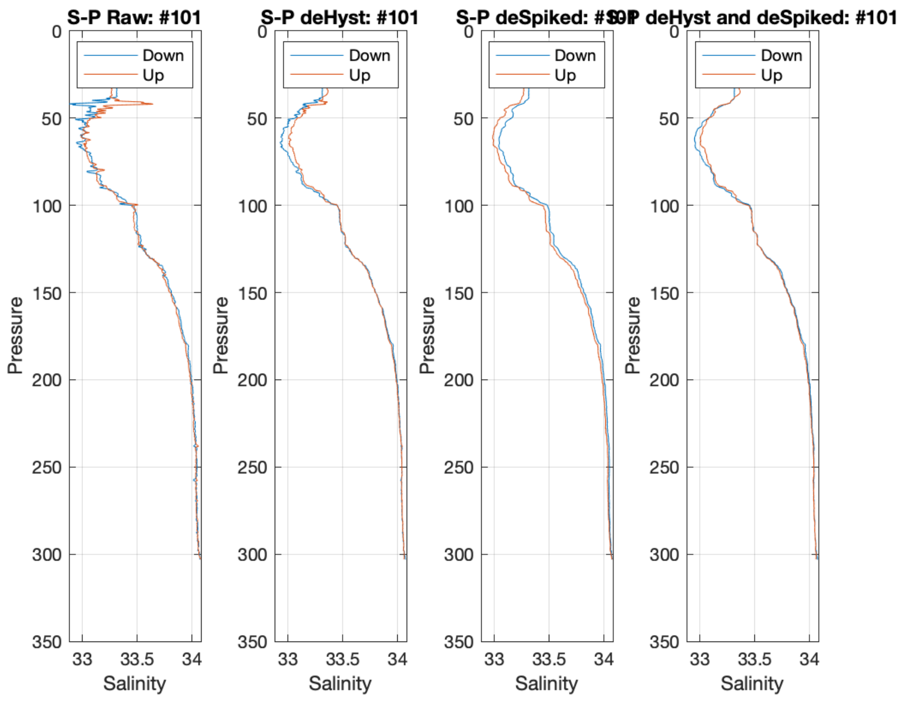

Figure 14: The salinity vs pressure for a single pair of profiles at different stages of the processing code. From left to right: (1) raw data, (2) hysteresis-only correction, (3) response-only correction, (4) both hysteresis and response corrections.

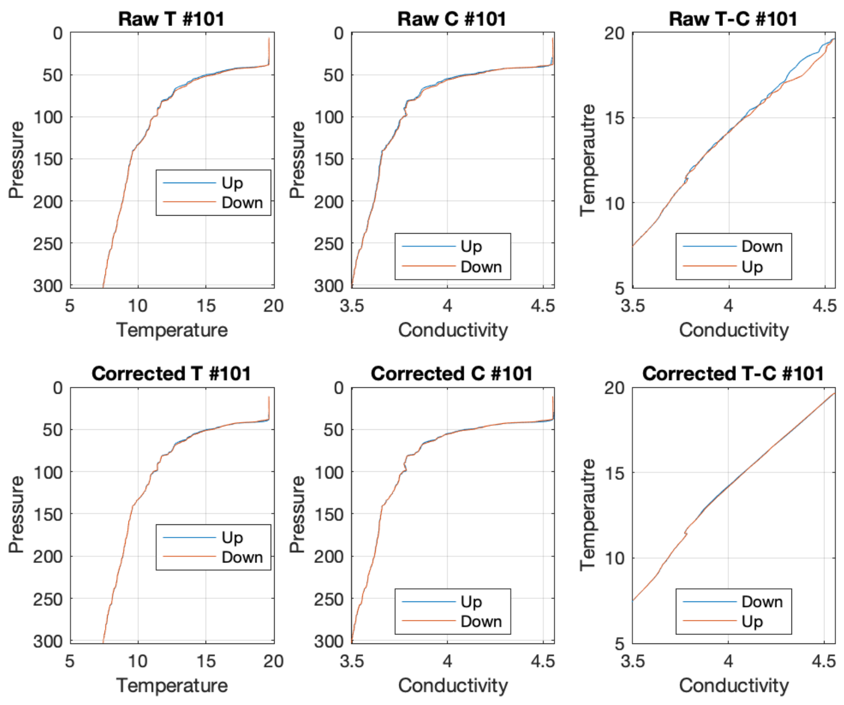

Figure 15: Temperature and conductivity for one pair of up-down profiles before and after correction.

3 Validation

The goal of this whole FastCTD processing routine is to improve the data quality vs the raw data, and to improve over the current defacto processing routine, San's 'FastCTD_GridData.m'. Improving the data means (1) removing hysteresis which creates a difference in the observed water column on the way down vs the way up, and (2) removing salinity spiking which results in spurious density inversions.

3.1 Average Profiles

An initial checkpoint is looking at the average profiles over the course of the deployment. Without any corrections to the raw data, the salinity spiking should average out to have zero effect on the average, and the hysteresis between up and down profiles should average out to the correct value. Thereore we expect the average of the corrected dataset to match the average of the raw data. I plot the average of the 500 test profiles in both the up and down directions for the three methods in Figure 16. Unfortunately, my processing seems to have introduced some fairly large deviations away from the observed salinity and density averages at the base of the mixed layer and through the salinity minimum. In the zoomed in plot of Figure 17, these deviations are as big as 0.02 PSU in salnity and $0.05\,\mathrm{kg\,m^{-3}}$ in density.

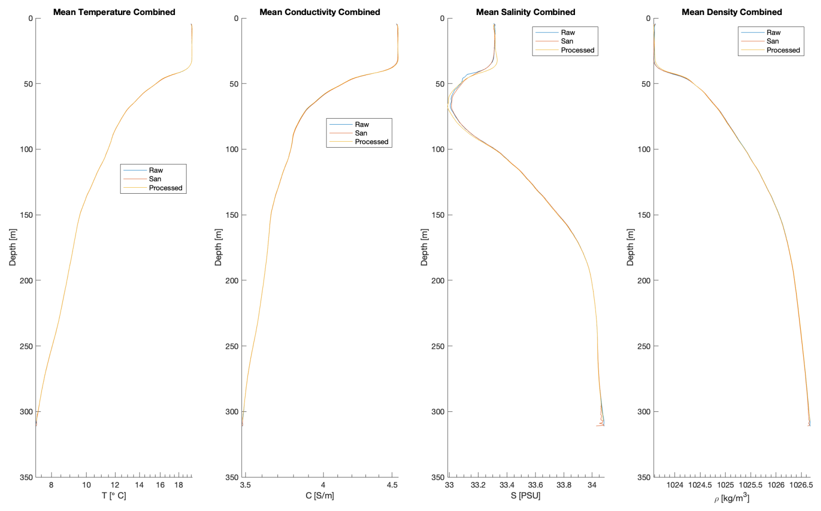

Figure 16: Mean combined profiles.

Looking at the means of up and down profiles separately in Figure 18, the positive salinity anomaly at the base of the thermocline is clearly caused by the up profiles, and the negative salinity anomaly at the salinity minimum is caused by the down profiles. Figure 19 zooms in on these regions.

3.2 Hysteresis

Checking the quality of the hysteresis correction is relatively straightforward. For each pair of profiles, I take the gridded up variable minus the gridded down variable, and do the time-mean to get the mean anomaly profile vs depth, shown in Figure 20.

For temperature, the magnitude of the up-down difference is much smaller than both the raw data and San's product. Throughout the profile, and sepecially in the thermocline, I am consistently left with a warm bias, with the up profile appearing warmer than the down profile. This represents and over-correction to the hysteresis, as it is approximately the same magnitude but the opposite sign to the difference in the raw data.

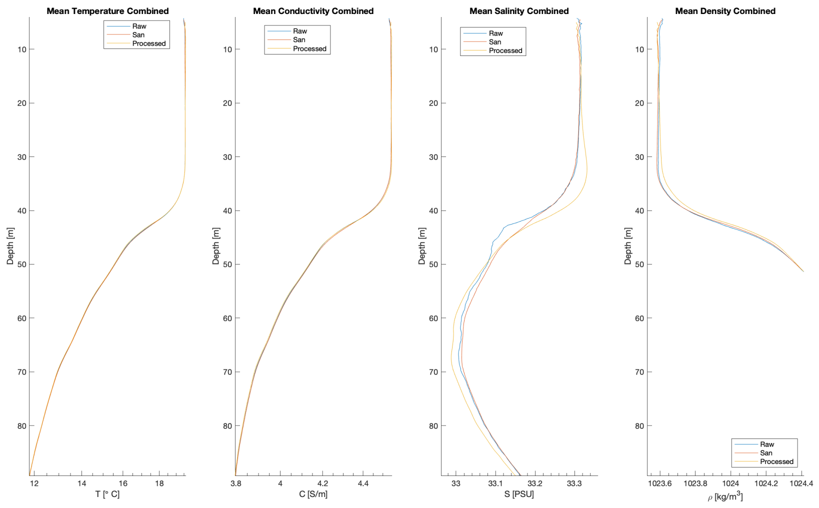

Figure 17: Mean combined profiles, zoomed in on the thermocline.

I don't have any ideas on how to improve on this error. In the mixed layer, I have a cold bias, indicating that not enough of a hysteresis correction has been applied.

For conductivity, I again have a smaller magnitude up-down error than both San and the raw data, except in the thermocline. The conductivity is also consistently over-corrected, and I end up with a bigger warm bias than the cold bias in the raw data. The mixed layer is under-corrected. I aslo don't have ideas on how to improve here.

For salinity and density, the magnitude of the difference is larger than both San's product and the raw data at various points in the thermocline, but is again an improvement at depth.

It makes sense that my corrections perform well at depth because both the de-hysteresis and response matching codes are tuned using the bottom of the profiles. I don't have any leads as to how I can improve these mehtods to work in the thermocline.

3.3 Salinity Spiking

Checking the performance of salinity despiking requires looking at individual profiles instead of averages. For this I plotted profiles from the raw data, San's product, and this processing in a loop. Figure 21 shows an example of the up-down profile comparion for temperature, conductivity, salinity, and density for each of the three products. A full file of 500 profiles is available here. Figure 22 compares the up profiles across the different methods, and is compiled here. Figure 23 compares the down profiles across methods and is compiled here.

It is clear that the response matching works, removing the salinity spikes and density inversions seen in the raw data. It also appears that this code performs about as well as San's. One glaring error, however, is visible in all the up profiles. At the top of the thermocline I appear to overshoot the conductivity conductivity correction, leading to anomaloysly salty and dense water. This creates a persistent density overturn at the base of the mixed layer and is problematic. I don't have ideas on how to address this. The error must be in the conductivity correction of the hysteresis code.

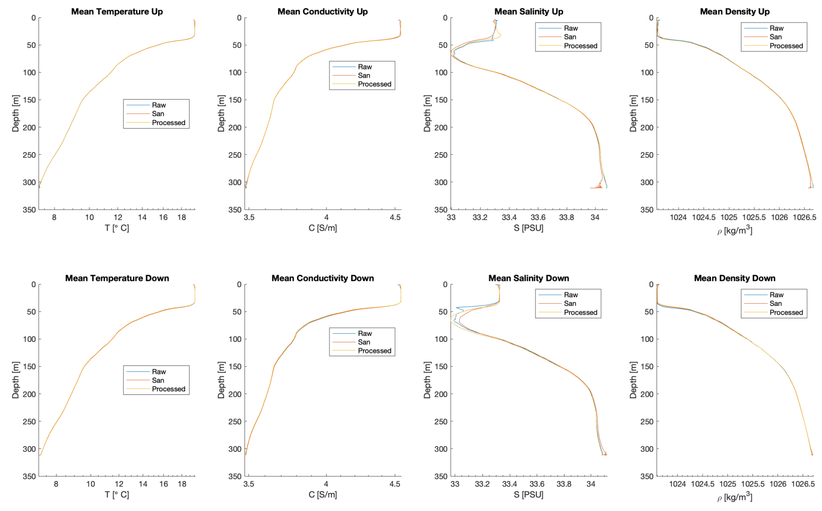

Figure 18: Mean combined profiles, up and down separately.

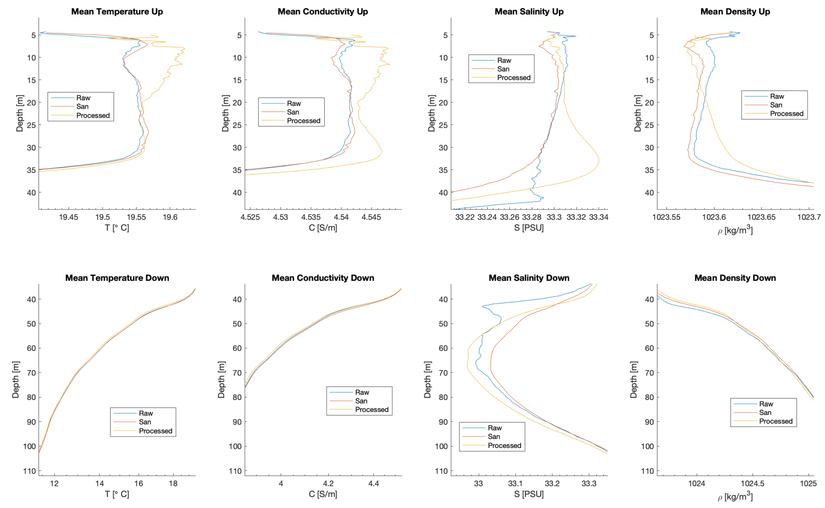

Figure 19: Mean combined profiles, up and down separately, zoomed in on the thermocline.

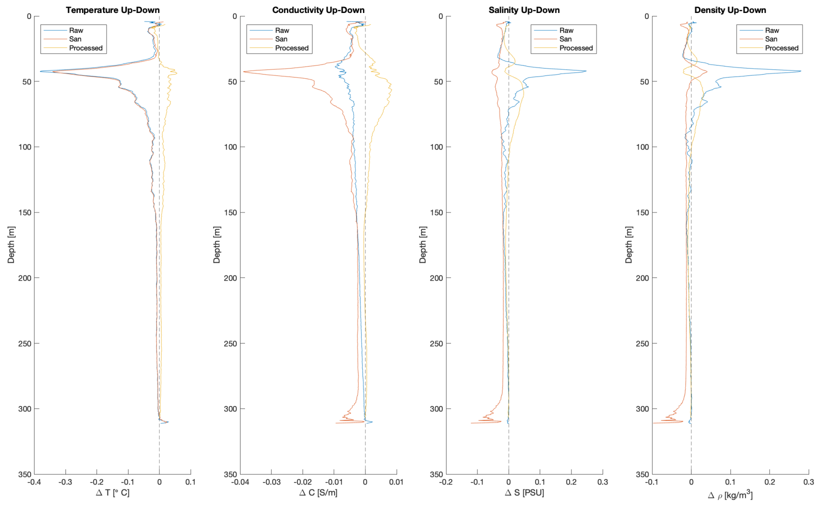

Figure 20: Mean anomaly of up-down profiles resulting from hysteresis.

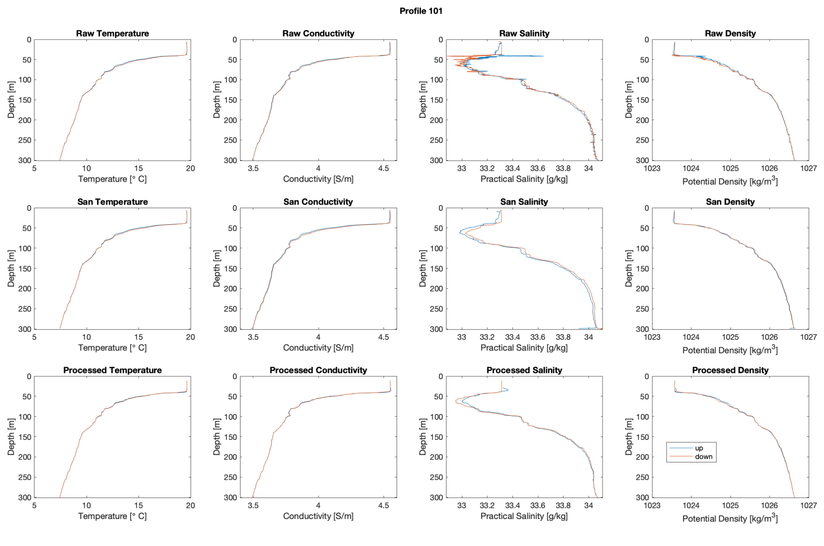

Figure 21: Data for a single pair of profiles without processing (Raw), using San's gridding code (San), and with this processing code (Processed).

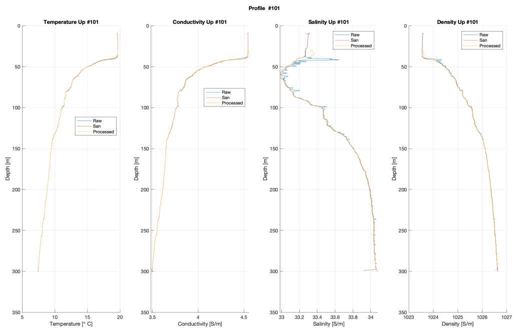

Figure 22: Data for a single profile in the up direction without processing (Raw), using San's gridding code (San), and with this processing code (Processed).

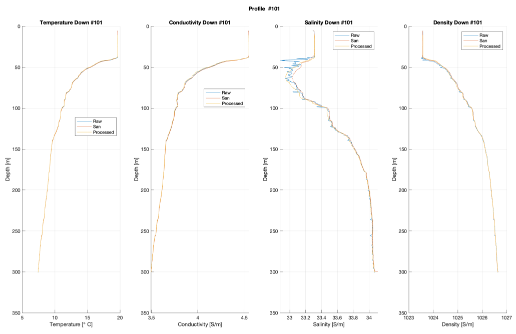

Figure 23: Data for a single profile in the down direction without processing (Raw), using San's gridding code (San), and with this processing code (Processed).

References

[1] Rolf G. Lueck and James J. Picklo. Thermal inertia of conductivity cells: Observations with a sea-bird cell. *J. Atmos. Ocean. Tech.*, 7:756–768, 1990.