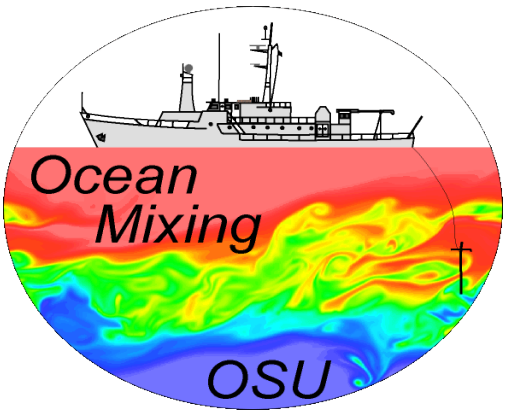


The χpod and gusT processing package

Johannes Becherer, Sally Warner, Deepak Cherian

Last updated: Apr 3, 2018

Contents

Part I

Calibration

Chapter 1

Compass

It can be confusing how the angles from the compass in χpods compare to flow direction, and how to ensure correct compass calibration. This document will cover the following:

1. How the compass is orientated within the χpod
2. How to convert from χpod compass angle to flow angle
3. How to include a compass offset
4. How to include magnetic declinations
5. Checking compass calibration with Pitot and ADCP
6. Errors to be aware of

1.1 Compass orientation within the χpod

If the compass is placed correctly within the χpod, it will be orientated such that it points towards the sensor tips. Therefore, it will read 0° when the sensor tips are pointed northward. Figure 1.1 shows a diagram of how the compass is orientated within the χpod.

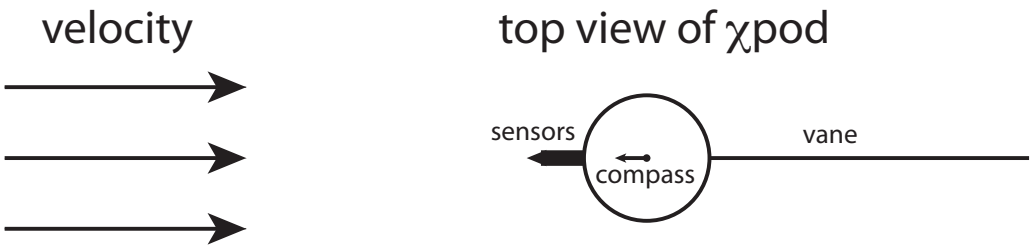


Figure 1.1: *A perfectly installed compass should align with the sensors. The χpod vane will steer the sensors into the flow.*

1.2 Converting from χpod compass to flow direction

Converting between compass angle and flow direction is confusing for 2 reasons:

- The compass uses heading orientation with north = 0°, and increasing clockwise such that east = 90°, south = 180°, and west = 270° = -90°. This is different from velocity for which we typically use vector notation with east = 0°, and angles increasing counterclockwise such that north = 90°, west = 180°, and south = 270° = -90°.

- Additionally, the sensors point *toward* the flow so they indicate the direction the flow is *coming from* not the direction to which it is going. Therefore, flow to the north would give a compass reading of 180° since the sensors would be pointed to the south. See Figure 1.2 for a complete comparison of angles.

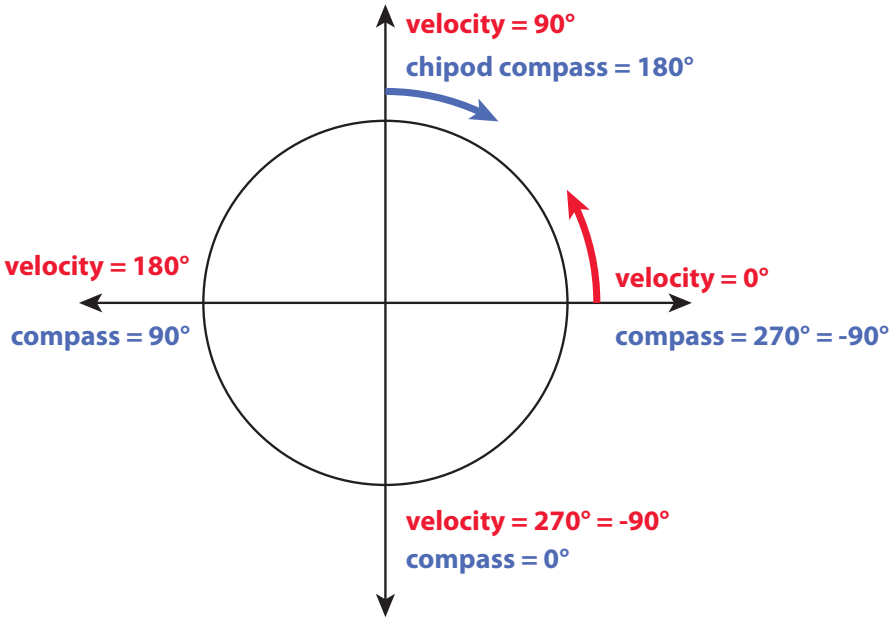


Figure 1.2: *Angles of flow direction in vector notation (red) and the corresponding compass readings within the χpod. This assumes, of course, that the χpod sensors are vaned to face directly into the flow.*

The formula that can be used to convert χpod compass angle to flow direction (in vector reference frame) is:

$$\theta_{flow} = -\left(\theta_{compass} + 90°\right) \tag{1.1}$$

Alternatively, to convert from flow direction to compass:

$$\theta_{compass} = -\left(\theta_{flow} + 90°\right) \tag{1.2}$$

1.3 How to find the compass offset

When the χpods are built, the compass may not be put into the χpod casing precisely such that the compass points directly to the sensors. Therefore, the compass is calibrated prior to deployment so we know what the offset is between the sensor direction and the compass direction. To do this, the engineers take the χpod into the woods away from magnetic sources and point the sensors to the north. They wait a minute, then rotate the χpod 10° clockwise. They repeat this all the way around the circle until the χpod sensors are again pointing toward the north.

Compass calibrations will be saved in a folder called something like "/Calibrations/compass/" within each χpod deployment folder. To view the compass calibrations:

```
% load compass calibration file, which would have a name like CMP_00000000.U
[data, head] = raw_load_chipod('CMP_00000000.UNT');
figure
plot(data.CMP/10)
```

This will create a figure that shows the compass calibrations (see Fig. 1.3 for an example). There should be 37 steps of 10° increments. Use `ginput` to select the first and last values to get the offset between the sensor tips and the compass.

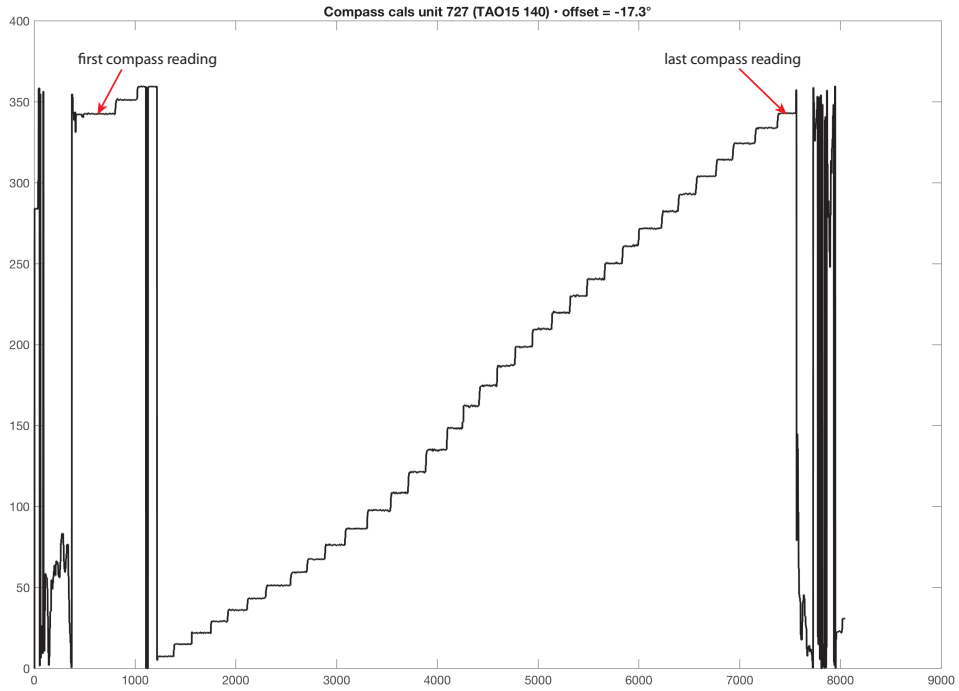


Figure 1.3: *Data from χpod compass calibration. Compass angle in degrees between 0° and 360° on y-axis, unimportant time steps on x-axis. Use `ginput` to get first and last value. In this case, the offset was found to be 342.7° = -17.3°.*

To apply this offset to the header.mat calibration file, use the following commands in MATLAB. `UNT` is the unit number, `DIR` is the directory of the χpod processing which contains folders like calib, mfiles, input, proc, raw, etc., and `OFFSET` is the angle found using ginput from the compass calibration file. It is important to include the negative offset in the header file (head.mat) because this is the value that is *added* to all of the compass readings.

```
% move to the correct directory
    cd DIR
% load the header file that includes all of the calibrations
    load calib/header.mat
% display the compass calibration which should be [0; 1; 0; 0; 0]
    head.coef.CMP
% add the offset to the header. **Input the NEGATIVE of OFFSET**
    head.coef.CMP(1) = -OFFSET
% save the updated header
    save calib/header.mat head
```

1.4 How to include magnetic declinations

Magnetic declination arises because the magnetic field of the earth is not uniform. This is especially important in the Pacific Northwest: a compass pointing directly north will give a reading of about 15° east of north due to the large mass of the Cascade mountains to the east. Note that the magnetic declination changes slowly over time. This needs to be included in the compass calibrations!

Look up the magnetic declinations in Corvallis where the instrument was calibrated and at the deployment location on the following website:

`https://www.ngdc.noaa.gov/geomag-web/#declination`

In 2017, the magnetic declination in Corvallis was about 15°E (Fig. 1.4). The magnetic declination at 0°, 140°W, where many χpods are deployed is 9.3°.

Now, instead of just inputting the OFFSET into the header calibrations, you'll want to include the declinations. Repeat the steps in the previous section, but include the declinations:

```
% add the offset and declinations to the header
    head.coef.CMP(1) = -OFFSET - DECL_CORVALLIS + DECL_DEPLOYMENT
% don't forget to save the updated header
    save calib/header.mat head
```

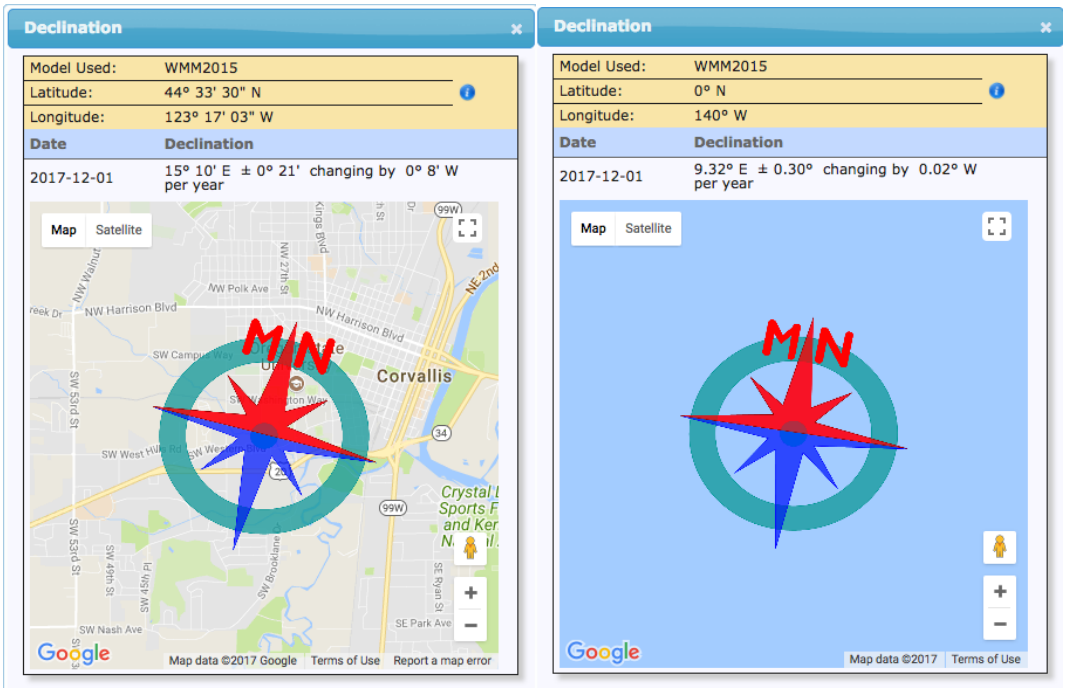


Figure 1.4: *The magnetic declination in Corvallis and at 0°, 140°W on December 1, 2017.*

For our example, the offset was -17.3°, the magnetic declination in Corvallis was 15.2°, and the magnetic declination at the deployment location was 9.3°. Therefore, we set the header coefficient to be 11.4° ($--17.3-15.2+9.3=11.4$).

Alternately, within the χpod_gust software, you can include the compass offset and magnetic declinations in `pre_driver.m` (shown with example values input into code):

```
LINE 24       modify_header = 1;    % if 1, specify header corrections below
LINE 25                             % (e.g. declination)
LINE 26
LINE 27       % get declination: https://www.ngdc.noaa.gov/geomag-web/#decli
LINE 28       CompassOffset = -17.3; % exact value from calibration file
LINE 29                             % (no sign changes!)
LINE 30       DeployDecl = 9.3; % at deployment location
LINE 31       CorvallisDecl = 15+10/60; % at corvallis
...
LINE 81       head.coef.CMP(1) = -CompassOffset - CorvallisDecl + DeployDecl,
```

1.5 Checking compass calibrations with Pitot and ADCP

If you have ADCP data and the compass has been calibrated correctly, pitot derived velocities should match up very well with ADCP velocities. Fig. 1.5 shows the output from `calibrate_pitot`. Additonally, when `calibrate_pitot` is run, an angle offset between the ADCP and pitot is printed to the screen. This should be small (i.e. $< 5°$).

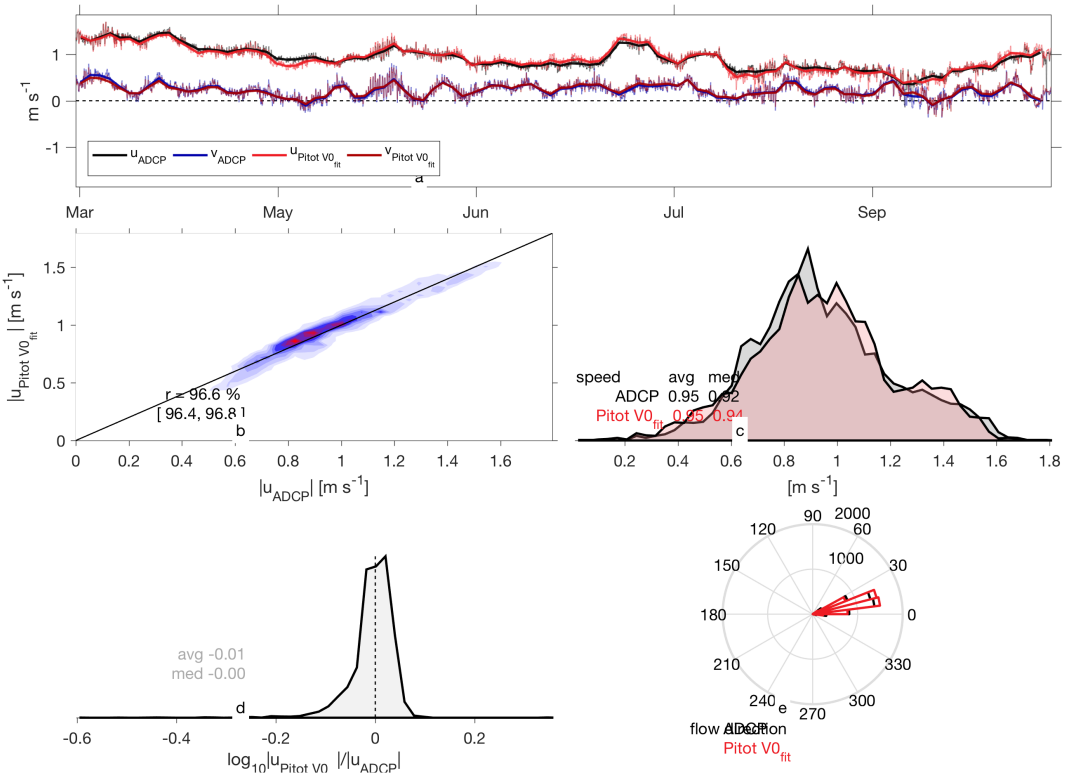


Figure 1.5: *Output of `calibrate_pitot` when the compass has been correctly calibrated. ADCP and pitot velocities match closely.*

1.6 Errors to be aware of

There are a few problems that can arise.

1.6.1 Nonlinear compass calibrations

There have been a few cases where the compass calibrations look very nonlinear (e.g. Fig. 1.6). In other words, the compass steps during the calibration are not all 10°. Some are as big as 50°, and many are $< 5°$.

To fix this, we did not use the compass calibration from Fig. 1.6, which was 195.6°. Instead, we manually found the offset that got the compass direction to match the flow direction as measured by the ADCP (Fig. 1.7). The new offset is found to be 91.4°. A big difference! This would not work as well in cases where the current was not uniformly flowing in one direction as it was for this χpod.

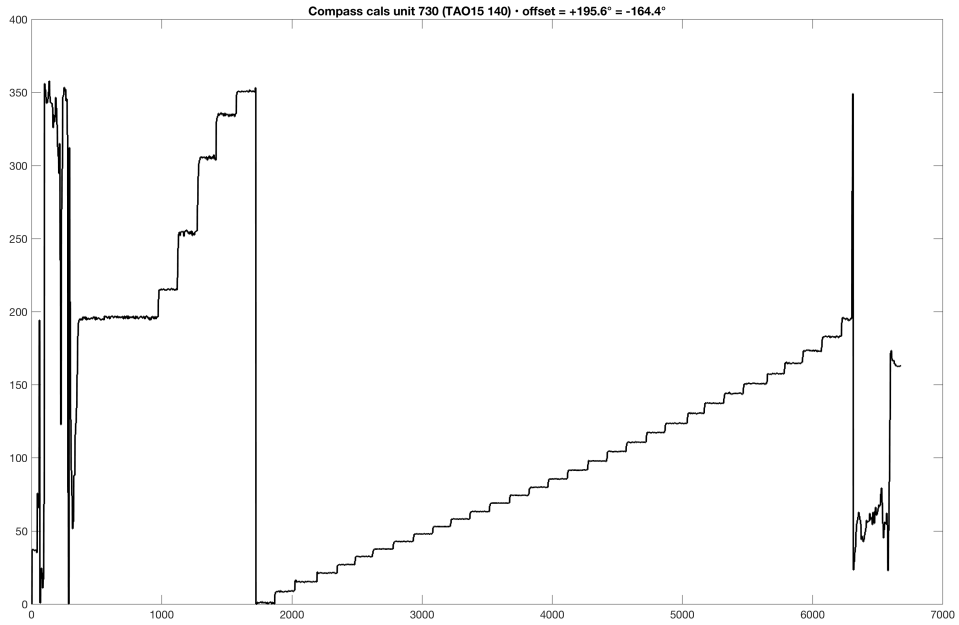


Figure 1.6: *Example of bad compass calibrations. Not all steps are equal.*

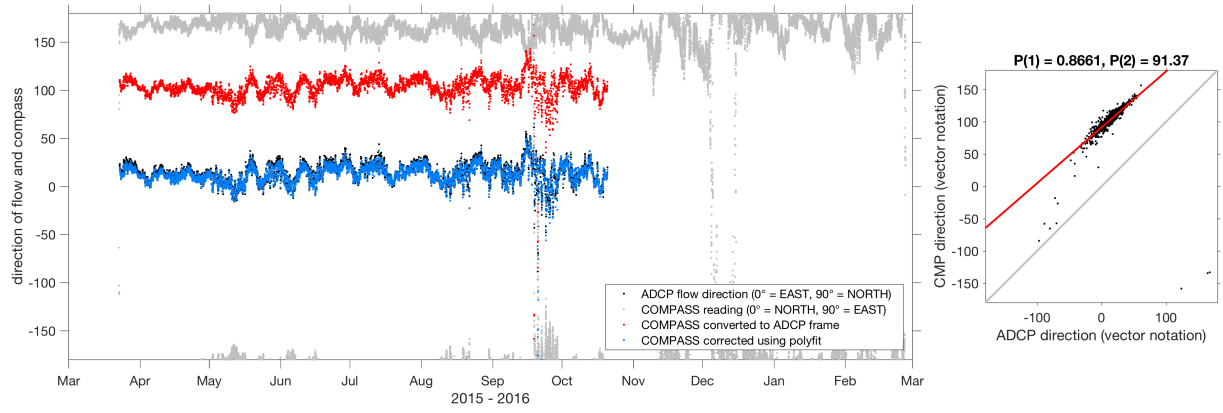


Figure 1.7: *Manually finding the compass offset that makes the compass match the ADCP. Flow direction from ADCP (black dots), compass readings in heading coordinates (gray dots), compass readings converted to vector coordinates (red), compass readings in vector coordinates with an offset of 91.37° applied.*

1.6.2 Bad comparison between ADCP and Pitot

Sometimes, the comparison between the ADCP and the pitot are just plain bad. This is somewhat hard to diagnose. Maybe the pitot goes bad. Try doing the calibration over a

shorter period of time and look for jumps in the signal. Maybe the compass calibrations are bad, in which case trust the angle given in the ADCP vs pitot comparison.

You can ignore the compass in the χpod processing by setting

```
pflag.master.use_compass = 0;
```

in `main_driver.m`. This will assume that the χpod is always perfectly vaned directly into the flow. When the compass is bad, the pitot will be able to give speed but not direction unless there is an ADCP to compare to.

1.6.3 Declinations already included in the calibration

There were some cases where the magnetic declination was being included in the compass calibrations. I'm not sure how this was being done. It's impossible to know whether they were or weren't included unless a note is added in the calibration folder. If you suspect this may be the case, (1) ask Pavan if he remembers how the compass was calibrated, (2) try adding and subtracting the declination in Corvallis to see if that helps, or just (3) use the ADCP to pitot comparisons to get the compass offset.

Chapter 2

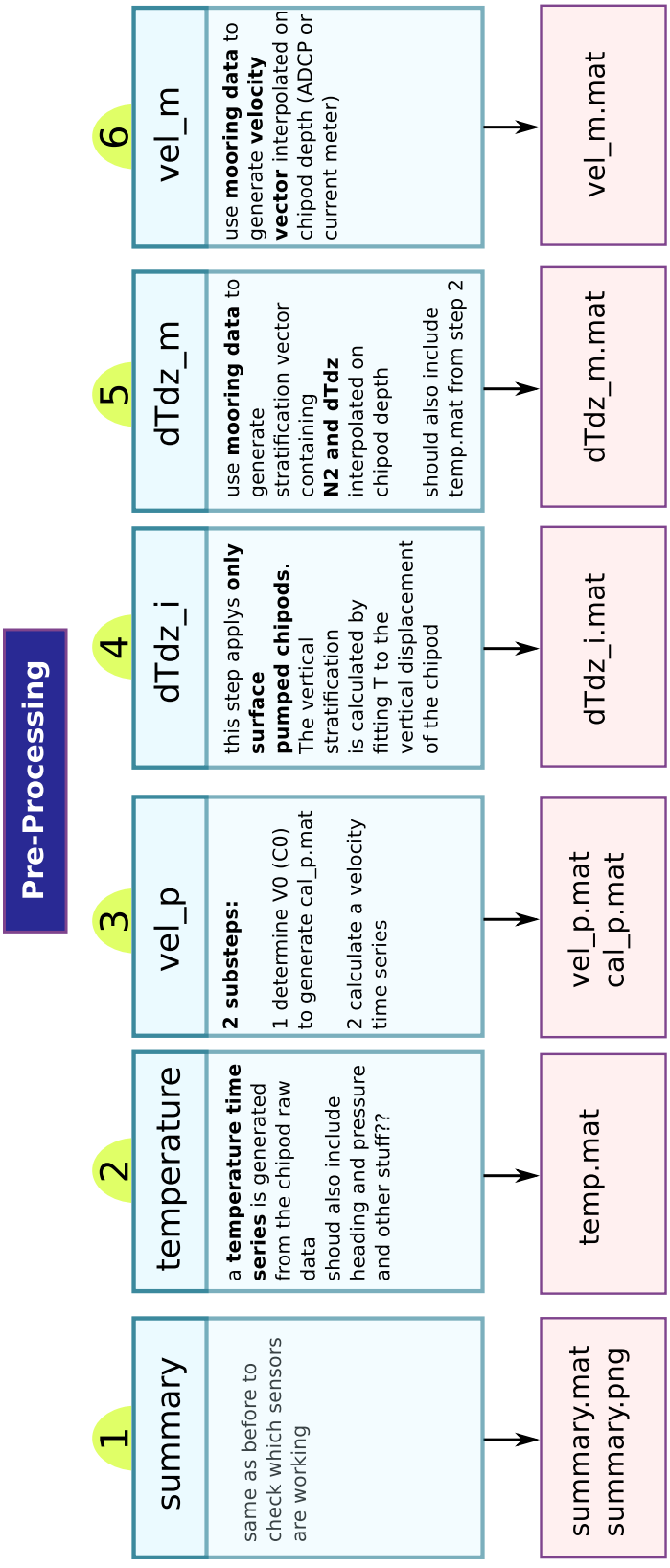
Pitot-static tubes

Part II

Processing

Chapter 3

`pre_driver`


Pre-Processing
1
summary
same as before to check which sensors are working
summary.mat
summary.png
2
temperature
a temperature time series is generated from the chipod raw data
shoud also include heading and pressure and other stuff??
temp.mat
3
vel_p
2 substeps:
1 determine V0 (C0) to generate cal_p.mat
2 calculate a velocity time series
vel_p.mat
cal_p.mat
4
dTdz_i
this step applys only surface pumped chipods. The vertical stratification is calculated by fitting T to the vertical displacement of the chipod
dTdz_i.mat
5
dTdz_m
use mooring data to generate stratification vector containing N2 and dTdz interpolated on chipod depth
should also include temp.mat from step 2
dTdz_m.mat
6
vel_m
use mooring data to generate velocity vector interpolated on chipod depth (ADCP or current meter)
vel_m.mat

Chapter 4

`main_driver`

Chapter 5

`combine_turbulence`

This is the final step.

5.1 Masking criteria/thresholds

5.1.1 Background stratification: `min_dTdz, min_N2, additional_mask_dTdz`

Minimum dT/dz, N^2 required for valid computation of χ, K_T, J_q^t

The Seabird SBE-37 datasheet says (`http://www.seabird.com/sbe37si-microcat-ctd`)

- T is accurate to $2 \times 10^{-3}\,^{\circ}\mathrm{C}$
- conductivity is accurate to $3 \times 10^{-3}\,\mathrm{psu}$ (approx!)

i.e.,

- dT/dz is accurate to (2x2e-3)/dz
- dS/dz is accurate to (2x3e-3)/dz

5.1.2 Sensor deaths: `T1death, T2death; nantimes{1,2,3}`

5.1.3 Averaging: `avgwindow, avgvalid`

5.1.4 Orientation / sensed-volume-flushing

5.1.5 Background flow speed: `min_inst_spd, min_spd, additional_mask_`

5.1.6 Maximum thresholds on $\epsilon, \chi, K_T, J_q^t$

5.1.7 Deglitching: `deglitch_window, deglitch_nstd`

5.2 Useful Functions

5.2.1 ChooseEstimates

5.2.2 chiold (variable)

combine_turbulence saves the *unmasked* chi structure as chiold after calculating Kt, Jqt but before doing any processing. Useful in checking masking.

5.2.3 TestMask

Lets you quickly iterate through various thresholds for a particular criterion.

Throws up a figure with histograms of counts to compare.

Example usage:

```
TestMask(chi, abs(chi.dTdz), '<', [1e-4, 3e-4, 1e-3], 'Tz');\\
```

This will iterate and mask using chi.dTdz < 1e-4, then chi.dTdz < 3e-4 and finally chi.dTdz < 1e-3. Each iteration is **independent** of the previous one.

5.2.4 DebugPlots

Usually, you want to see the effect of different masking thresholds in a small subset of the time series of χ, ε etc.

Example usage:

```
chi = chiold; \% reset to unmasked structure\\
\% apply 3 different thresholds\\
chi1 = ApplyMask(chi, abs(chi.dTdz), '<', 1e-4, 'T\(_{\text{z}}\) < 1e-4');\\
chi2 = ApplyMask(chi, abs(chi.dTdz), '<', 1e-3, 'T\(_{\text{z}}\) < 1e-3');\\
chi3 = ApplyMask(chi, abs(chi.dTdz), '<', 2e-3, 'T\(_{\text{z}}\) < 2e-3');\\
\vspace*{1em}
t0 = datetime(2016, 12, 10);\\
t1 = datenum(2016, 12, 12);\\
tavg = 600;\\
\vspace*{1em}
hf = figure;\\
\% plot 10 minute averages (tavg) of quantities in structure chi\\
\% between [t0, t1] and label them as 'raw' in figure window hf\\
DebugPlots(hf, t0, t1, chi, 'raw', tavg);\\
\vspace*{1em}
\% compare different masking; label appropriately\\
DebugPlots(hf, t0, t1, chi1, '1e-4', tavg);\\
DebugPlots(hf, t0, t1, chi2, '1e-3', tavg);\\
DebugPlots(hf, t0, t1, chi3, '2e-3', tavg);\\
```

5.2.5 DebugRawData

The idea is to connect Turb.() to the raw-ish data.

Requires structure T from temp.mat.

5.2.6 Histograms2D

Chapter 6

Winters & D'Asaro (1996): K_T, J_q^t from χ

6.1 Usage

First generate files containing the sorted gradient by running `do_dTdz_i_proc.m` with `wda_params.do_winters_dasaro = 1` (enabled by default). This will save `input/dTdz_w.mat` which contains the structure `Tz_w` with the fields:

- `Tz1, Tz2` - time series of volume-averaged sorted gradients T_z calculated using sensors 1,2 respectively.
- `zT1, zT2` - time series of volume-averaged inverse sorted gradients z_T calculated using sensors 1,2 respectively.
- `sgn_moor, sgn_int1, sgn_int2` - Sign of the mean gradient inferred from the mooring gradient or the sensor-appropriate "internal" gradient (fitting to a cloud of points).
- `time` - time vector for above fields
- `wda1, wda2` - Bins and other information required to estimate K_T, J_q using the Winters & D'Asaro (1996) equation.

`do_dTdz_i_proc` automatically runs `compare_dTdz` to generate a plot that compares the sorted gradient, internal gradient and the mooring gradient (if available). This image is saved as `pics/Compare_dTdz.png`.

When enabled, `combine_turbulence` will read in `dTdz_w.mat` and call `process_wda_estimate` to obtain K_T, J_q for *each VC estimate*. The output `Turb.mat` file will contain `wda` substructure under each estimate i.e. for estimate `mm1` the fields `mm1.wda.Jq, mm1.wda.Kt` are estimates computed using the sorted temperature field. *These are different from `mm1.Jq`, `mm1.Kt` which are set using the old way of doing things i.e. dividing by mooring gradient in this case.* Because we are using the sorted T_z field, there is no reason to mask using `min_dTdz`. All other masking (and deglitching) is applied.

6.1.1 Options

`do_dTdz_i_proc`:

- `wda_params.do_winters_dasaro` : switch on (1) or off (0)
- `wda_params.wda_dt` : (seconds) length of time chunk that is processed at a time. Default is 60 seconds which works OK in the top 30m or so. For deeper χpods you can go longer (3 minutes?). Note that number of quantiles (bins) is scaled appropriately assuming that 5 quantiles for 60 seconds of data is good coverage.

The sensitivity of the mean sorted gradient to `wda_dt` is small (Figure 6.1)

`combine_turbulence`:

- `CP.wda_Tz_sign` : (`'m'` or `'i'`) - gradient time series used to assign sign to heat flux. Chooses the appropriate field from `Tz_w`.

6.2 Theory

Osborn & Cox:

$$J_q^t \sim \frac{\langle \chi \rangle}{2\langle \frac{\partial T}{\partial z} \rangle} \tag{6.1}$$

These angle brackets are interpreted as a time average for χpods and a depth-bin average for profilers like Chameleon.

The Osborn & Cox formulation is not well behaved when stratification is low. It gets worse in places with large salinity influence: large enough that $T_z < 0$ some of the time. When T_z crosses through zero, the estimates of K_T, J_q^t blow up. In these cases, getting "reasonable" values for K_T and J_q^t requires masking out estimates when T_z is less than some ill-defined threshold.

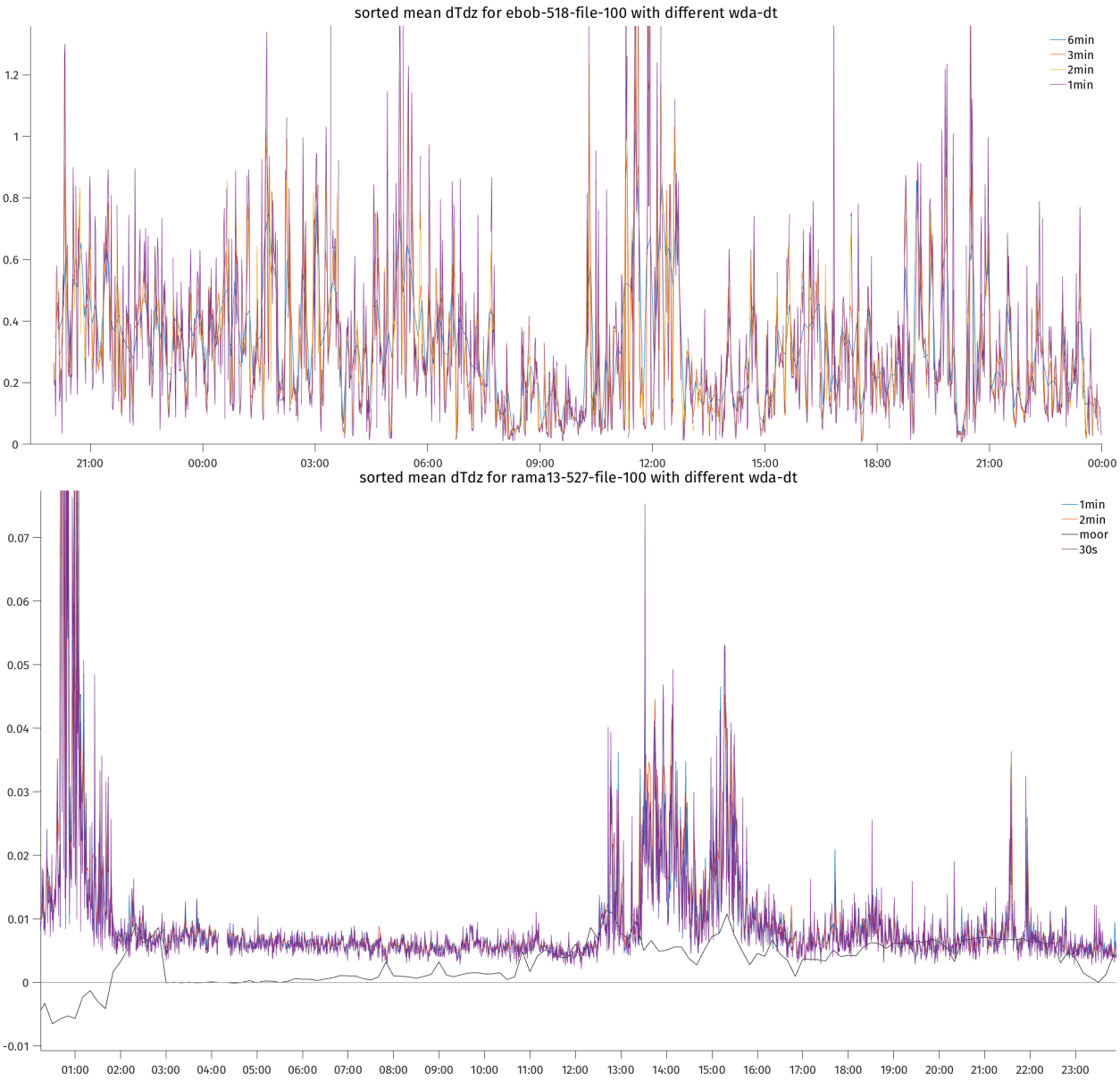


Figure 6.1: *Sensitivity of mean sorted gradient to* `wda_dt` *for a day of data. As expected, for larger time chunks the gradient is smoother. Even when the mooring gradient is tiny, the sorted estimate is well behaved and has converged (the lower panel is a 30m instrument which is pumped quite frequently).*

Winters & D'Asaro derive the continuous form:

$$J_q^t = \frac{1}{2}\frac{\mathrm{d}z^*}{\mathrm{d}T}\langle\chi\rangle_{z^*} \tag{6.2}$$

z^* being the "reference" state — usually a fully sorted 3D scalar field. In practice, we approximate this by Thorpe sorting single vertical profiles. The angle brackets represent averaging in "isoscalar" (isothermal) space.

The Winters & D'Asaro formulation has the advantage that it can be well-defined for low gradients — by thinking about distances between isoscalar surfaces. The idea is that the averaging occurs in a volume between two isothermal surfaces. The required gradient is then the average spacing between these two surfaces. Doing this requires treating the χpod as a profiler, being pumped by surface waves. We can keep track of the isotherms seen by the χpod in a chunk of data; and estimate the average distance between those isotherms as the χpod is pumped up and down. For dense enough sampling this average distance should ideally converge to the distance between the isotherms in the fully sorted field (this being the necessary gradient). Note that the χpod densely samples approximately 1-2 metres of the water column.

Why is the gradient of the (full 3D) sorted field the right gradient to use?

- Mixing acts to change the background potential energy of the fluid — the BPE is determined by computing the PE of the 3D-sorted density field.
- We want to infer the heat flux that goes into changing this BPE i.e. a sensible definition of heat flux is that which changes BPE by a given amount.
- The way to do this is to locate our χ observations in the *sorted reference state* and then compute heat flux — the appropriate gradient being that of the sorted state.

In practice, we approximate the gradient of the full 3D sorted state by differencing moored CTDs, fitting a straight line to 50Hz temperature measurements (internal estimate) or sorting 1D profiles (as we do here).

One objection is that we know T at $100\,\mathrm{Hz}$ and χ at $1\,\mathrm{Hz}$; and we want to sort T at the highest possible frequency. Consider however the discrete form of J_q^t defined in Winters & D'Asaro (1996). Let A_S is the surface-area of an extremely warped (stirred) temperature surface S and A be the area of that surface when it is flattened out so that g is normal to it (Figure 6.2). J_q^t is the diffusive flux normal to the warped surface S i.e. $\kappa\nabla T\cdot\hat{n}$ integrated

over S — κ is molecular diffusivity and $\nabla T \cdot \hat{n} = |\nabla T| \approx \Delta T/\Delta n$.

$$AJ_q^t = \kappa \int_S \nabla T \cdot \hat{n}\, \mathrm{d}S \tag{6.3}$$

$$J_q^t = \frac{1}{A}\int_S \kappa |\nabla T| \frac{\Delta T}{\Delta n}\frac{\Delta n}{\Delta T}\frac{\Delta z_*}{\Delta z_*}\, \mathrm{d}S \tag{6.4}$$

$$= \frac{1}{A\Delta z_*}\int_S \frac{\chi}{2}\frac{\Delta z_*}{\Delta T}\Delta n\, \mathrm{d}S \tag{6.5}$$

$\Delta z_*/\Delta T$, a property of the sorted background state, is independent of the instantaneous shape of the isothermal surface S. So we can write

$$J_q^t = \frac{\Delta z_*}{\Delta T}\frac{1}{A\Delta z_*}\int_S \frac{\chi}{2}\Delta n\, \mathrm{d}S = \frac{\Delta z_*}{\Delta T}\frac{1}{\Delta V}\int_{\Delta V}\frac{\chi}{2}\, \mathrm{d}V, \tag{6.6}$$

since z_* is *defined* such that

$$A\Delta z_* = \int_S \Delta n\, \mathrm{d}S = \Delta V = \text{volume of fluid between surfaces separated by} \Delta z_* \text{ or } \Delta T. \tag{6.7}$$

J_q^t is the product of $\Delta z_*/\Delta T$ and the *volume-averaged* χ between those two surfaces separated by $\Delta T, \Delta z_*$. It would appear that the two can be determined independently of each other. In low stratification, estimating the spacing requires high-resolution temperature measurements but that is exactly what the χpod does.

6.3 Algorithm

Consider a 1 minute chunk of data (configurable parameter `wda_params.wda_dt` in `do_dTdz_i_proc`).

- Once χ has been estimated, we have $\chi \equiv \chi(t) \equiv \chi(T_1)$, T_1 is the 1-second averaged temperature time series.

- We can divide the temperature time series into N quantiles, bin the χ estimates in these temperature bins and then average them to get $\langle\chi\rangle \equiv \chi(\Delta T_{\text{bins}})$. ΔT_{bins} represents the bins between bin edges T_{bins}. This is what Winters & D'Asaro (1996) call "isoscalar averaging" — every χ estimate made between 2 isothermal surfaces is averaged together.

- Next, we need to determine the average distance between the isothermal surfaces T_{bins}.

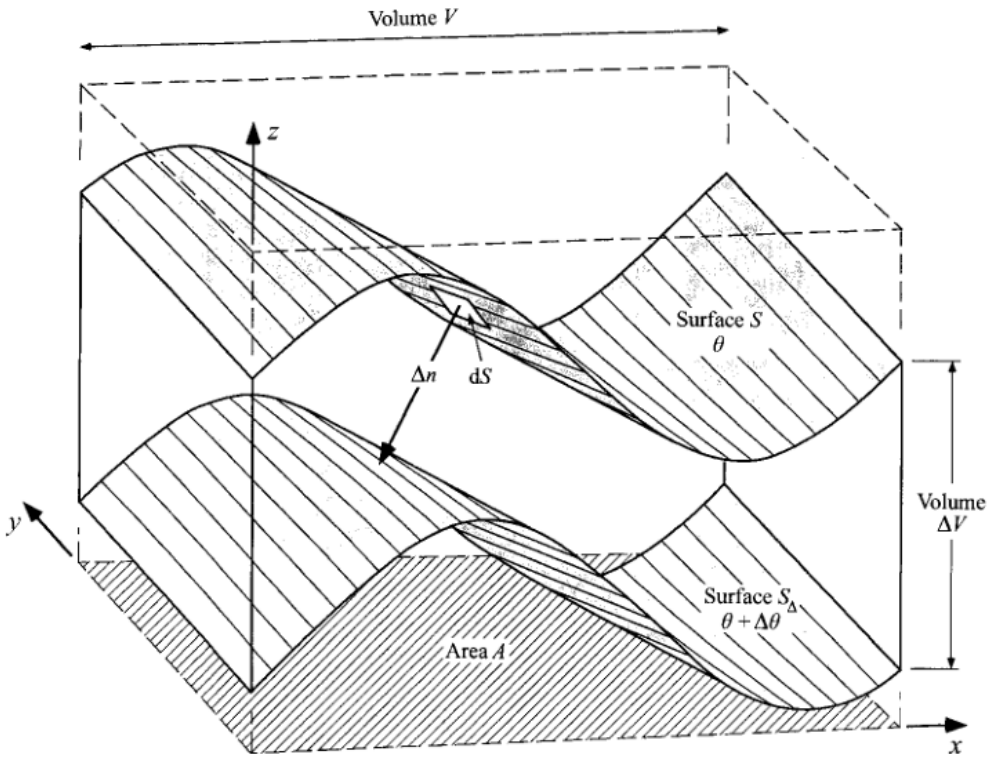


FIGURE 1. Schematic of two isoscalar surfaces in a volume V with cross-sectional area A. Diascalar flux changes ΔV, the volume of fluid between the surfaces S and S_Δ defined by the scalar values θ and $\theta + \Delta\theta$.

Figure 6.2: *Figure reproduced from Winters & D'Asaro (1996).*

- Determine the start and end of "up-" and "down-"casts using the intergrated accelerometer. Discard those that aren't sufficiently long enough.
- Sort the temperature associated with each "up-" and "down-cast" individually.
- Find the location of the chosen isotherms (T_{bins}) in the sorted profiles and difference them to get $\Delta z(\Delta T_{\text{bins}})$ in each profile.
- Average $\Delta z(\Delta T_{\text{bins}})$ in isothermal space to get $\langle \Delta z \rangle$; i.e. average every Δz measurement for each bin. $\langle \Delta z \rangle$ is the average distance between the isotherms represented by the bin edges T_{bins}.
- $\langle \Delta z \rangle / \Delta T_{\text{bins}}$ is the necessary gradient for each bin.

- $$J_q^t = -\frac{1}{2} \frac{\langle \Delta z \rangle}{\Delta T_{\text{bins}}} \langle \chi \rangle \tag{6.8}$$

  We now have a J_q^t estimate for each temperature bin. Depth-average this value to get the *volume-average J_q^t in the volume sampled by the χpod in the 60 second chunk of data*

6.4 Some considerations

6.4.1 Minimum recoverable dz_*/dT + noise floors

We use the method of Mudge & Lueck (1994) to combine the 50Hz temperature time-series T (resolution $\approx$ 1mK) and the 100Hz T' to obtain an "enhanced" time-series (resolution $\approx$ 0.01mK)[1]. The code was copied over from `mixingsoftware/marlcham/combine_ttc.m`. The temperature noise floor is not a problem then.

When T' is at its noise floor, that means there's no temperature fluctuations and we are either passing through a fully homogeneous patch or a non-turbulent patch. For these time instants, the enhanced temperature time series is set to the last valid observation (using `fillmissing(..., 'previous')`). Note that χ can be set to 0 for the same time instants in `do_combine_turbulence` (this choice results in molecular K_T being inferred).

The mean temperature gradient time series in `chi.().wda.dTdz` will *never recover a zero gradient* i.e. there is a lower bound to detectable mean gradients. The easy way to see this

[1]The differentiator uses the analog signal from the temperature sensor as input; so the noise floor in T' and T are different.

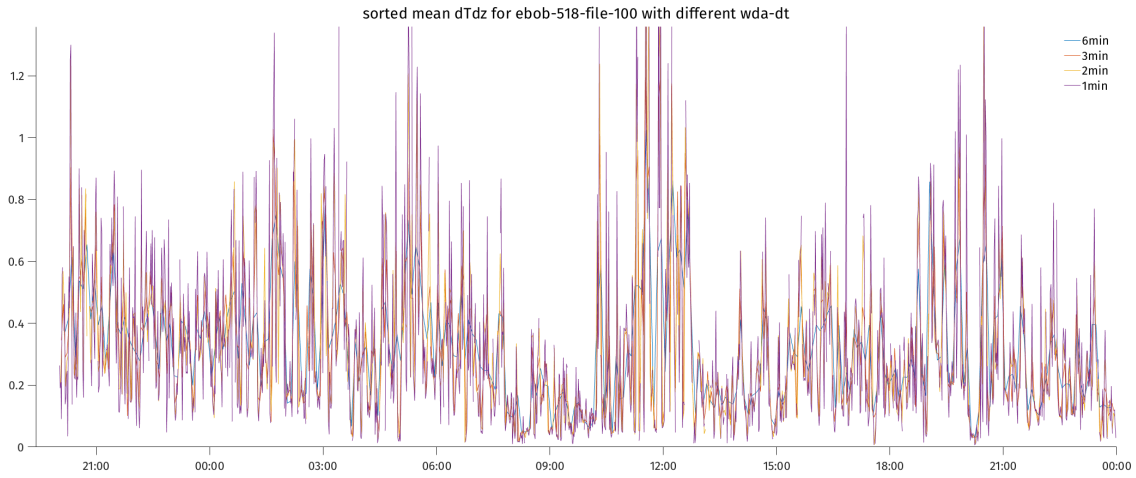


Figure 6.3: *Variation of sorted mean gradient with time interval* `wda_dt`*. Larger time intervals result in smoother gradient. This is for a time period with high stratification.*

is that in homogeneous fluid there are no temperature surfaces to form bins and so we cannot infer a gradient (also $J_q^t = 0$ which we recover by setting $\chi = 0$ when T' is at noise floor). As long as we choose a long enough time chunk within which to sort profiles, the χpod will sense some temperature fluctuations which can then be sorted to yield finite gradients (the sensed ΔT is a function of time chunk Δt for smallish Δt — see Figure 6.3). The lower bound on detectable gradient seems to be around $2 \times 10^{-3}\,^{\circ}\mathrm{C\,m^{-1}}$.

6.4.2 Sign of vertical heat flux

When salinity controls density, it is possible that $dT/dz < 0$ for extended periods of time. Thorpe sorting cannot preserve this sign (unless there are coincident measurements of salinity, in which case you would sort density). The only sensible thing to do is to determine the sign of the mean (or "large-scale") gradient from other sources such as nearby CTDs on the mooring, if available. Currently `combine_turbulence` uses the sign of the hourly running median of mooring T_z (can be switched to internal T_z using `CP.wda_Tz_sign`). During times of low gradients ($T_z <$ `CP.min_dTdz`), it uses the sign of the 2-hour running median since we want some sort of "stable" sign.

6.4.3 Number of bins

Currently the code uses 5 bins per minute of data and scales that according to `pflag.master.wda_dt`. The nunmber of bins is set by setting `nquantiles` in `do_wda_estimate`. More bins mean better use of data but fewer χ measurements to average in each bin. Few bins will result in the inability to determine gradients well because the temperature profiles could be too short to intersect 2 bin edges.

6.4.4 Amount of pumping

The sorted estimate requires a pumped mooring. With deeper χpods this can be a problem. One option might be to use the `wda` estimate when gradients are low and revert to the usual estimate when gradient is high.

6.4.5 Debugging

Say you want to debug the J_q estimate at a particular time. `winters_dasaro_avg`, which does all the work, can make a debugging plot when it receives the argument `plotflag = 1`.

So you'll want to

- Call `chi_main_proc` on the appropriate raw file.
- Uncomment the call to `do_wda_estimate` in `chi_main_proc`, and
- Insert a keyboard statement in that loop that calls `winters_dasaro_avg` in `do_wda_estimate`.
- Set `plotflag = 1` and call `winters_dasaro_avg` with appropriate `t0` for the time instant you want to check.

This will show something like Figure 6.4.

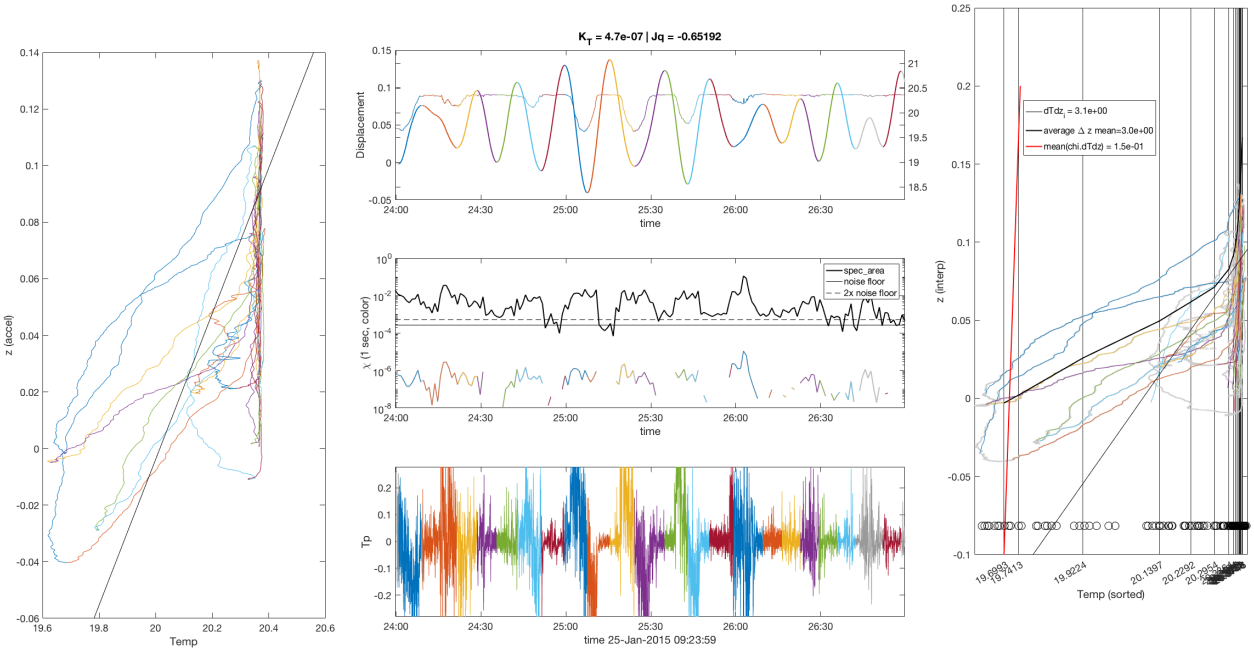


Figure 6.4: *Debugging the sorted gradient and J_q estimate (created by* `winters_dasaro_avg`*). On the left is the temperature observations plotted against depth on the y-axis. The center panels show time series of both vertical displacement and temperature (top panel); χ and area under the spectrum as well as the noise floor (middle panel), and Tp (bottom panel). On the extreme right are the sorted profiles, as well as straight lines showing dT/dz from the mooring and that obtained from the internal fitting. Colors indicate different profiles (see vertical displacement panel = center top). In this particular minute of the data the sorted gradient is extremely large (3°C/m on average, max ≈ 8°C/m), the mooring gradient ≈ 0.15 is quite small (dark red line in extreme right panel). The thick black line in the extreme right panel is a reconstruction of the sorted profile obtained by integrating the inferred $\Delta z_*/\Delta T$. Circles at the base indicate 1 sec averages of temperature, I.e. these are the temperature surfaces on which we have χ estimates. The vertical grid lines are the chosen temperature bins T_{bins}.*

Part III

Database

Chapter 7

data base

This chapter describes how to use the database functionality whoAmI?

At the beginning, there can be found two functions in the mfile directory of your unit directory:

- *whoAmI_generate.m*
- *whoAmI_feedback2database.m*

Then you can use *whoAmI_generate.m* to generate a third m-file, which is called *whoAmI.m*. *whoAmI.m* is the function that contains a complete list of all parameters that are saved in the data base that refer to your particular unit.

You may access these parameters by calling *whoAmI.m*, which outputs a structure containing all these parameters. This structure can then be used by the processing routines to get unit specific information.

If you want to change any of the parameters, you may go to *whoAmI.m* and change the corresponding parameter in the matlab code. Note that these changes are only local and do not immediately affect the corresponding data base entries. If you want to write your local changes permanently to the data base you can use *whoAmI_feedback2database.m*.

Part IV

Appendix

Appendix A

Using the CEOAS MATLAB servers

A.1 SSH configuration

A.2 Usage

1. login in to maltab server 1) open terminal 2) type: mats

2. logout type twice: exit

3. open matlab on server type: mat or : matno #(for no display)

4. How to copy from ganges to server-ganges cd /ganges/ sh pullFromGanges data/(dir you want to copy)/ otherwise use scp

Appendix B

Useful references

Becherer & Moum (2017) : Data reduction for IC schemes.

Moum & Nash (2009) : General χpod paper

Moum (2015) : Pitot tube paper

Perlin & Moum (2012) : Among others, this paper systematically compares estimates after turning off angular rate sensors, accelerometers, pressure sensors and compass in sequence. It also compares χpod estimates with nearby Chameleon estimates. Things look good!

Zhang & Moum (2010) : Initial IC estimate paper

References

Becherer, Johannes, & Moum, James N. 2017. An Efficient Scheme for Onboard Reduction of Moored Xpod Data. *Journal of Atmospheric and Oceanic Technology*, **34**(11), 2533–2546.

Moum, J. N., & Nash, J. D. 2009. Mixing Measurements on an Equatorial Ocean Mooring. *Journal of Atmospheric and Oceanic Technology*, **26**(2), 317–336.

Moum, James N. 2015. Ocean Speed and Turbulence Measurements Using Pitot-Static Tubes on Moorings. *Journal of Atmospheric and Oceanic Technology*, **32**(7), 1400–1413.

Mudge, Todd D., & Lueck, Rolf G. 1994. Digital Signal Processing to Enhance Oceanographic Observations. *Journal of Atmospheric and Oceanic Technology*, **11**(3), 825–836.

Perlin, A., & Moum, J. N. 2012. Comparison of Thermal Variance Dissipation Rates from Moored and Profiling Instruments at the Equator. *Journal of Atmospheric and Oceanic Technology*, **29**(9), 1347–1362.

Winters, Kraig B., & D'Asaro, Eric A. 1996. Diascalar Flux and the Rate of Fluid Mixing. *Journal of Fluid Mechanics*, **317**(-1), 179.

Zhang, Yanwei, & Moum, James N. 2010. Inertial-Convective Subrange Estimates of Thermal Variance Dissipation Rate from Moored Temperature Measurements. *Journal of Atmospheric and Oceanic Technology*, **27**(11), 1950–1959.

Appendix C

git commands and workflows

Appendix D

Sensor orientations

D.1 χpod

D.2 Pitot

from Moum (2015)

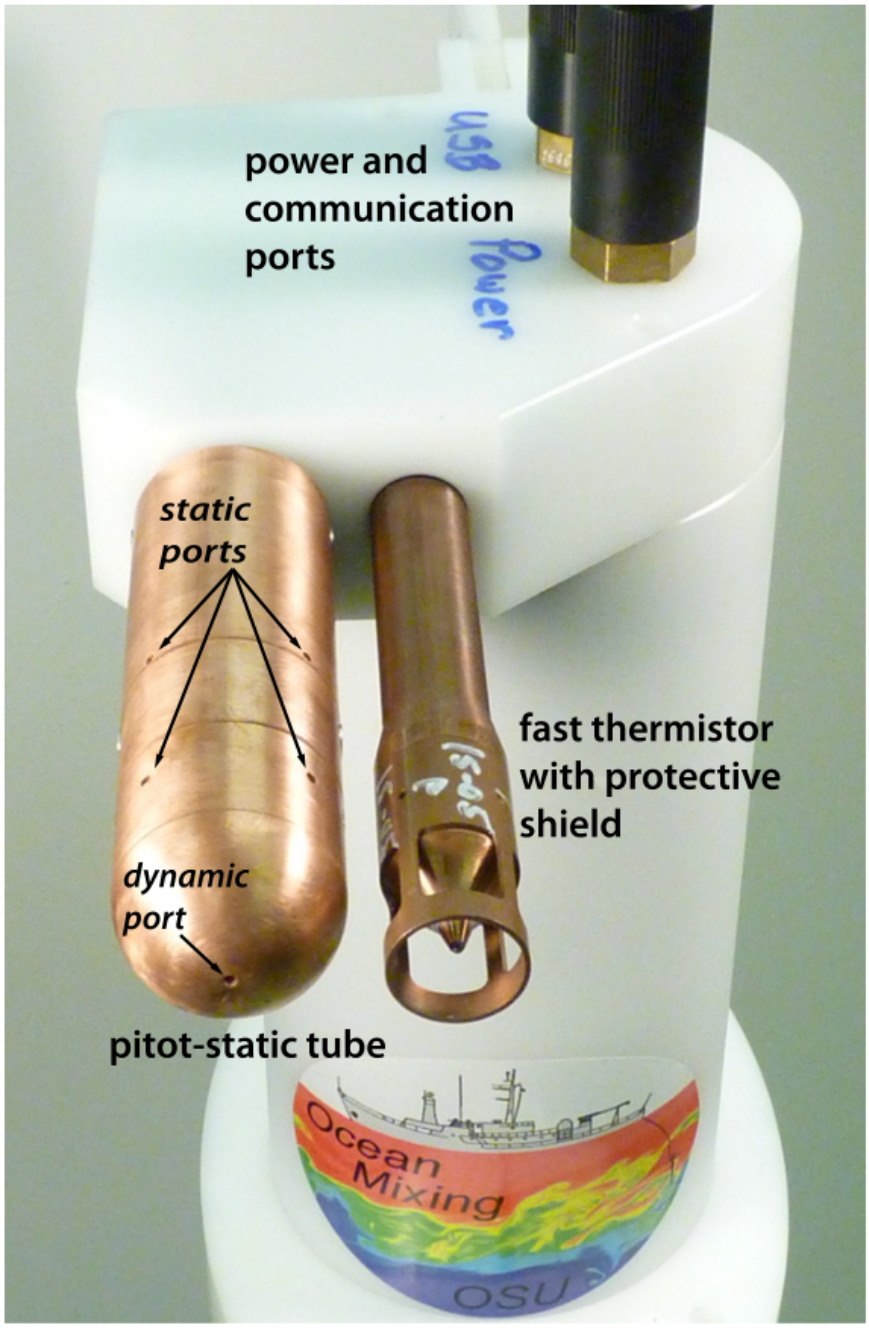


FIG. 1. Pitot-static tube mounted beside fast thermistor on χpod. Each is plugged into a connector in the end cap of the pressure case that houses electronics and batteries. For reference, the visible length of the pitot-static tube extending from the χpod end cap is 10.6 cm. (Photo courtesy of Craig Van Appeldorn.)

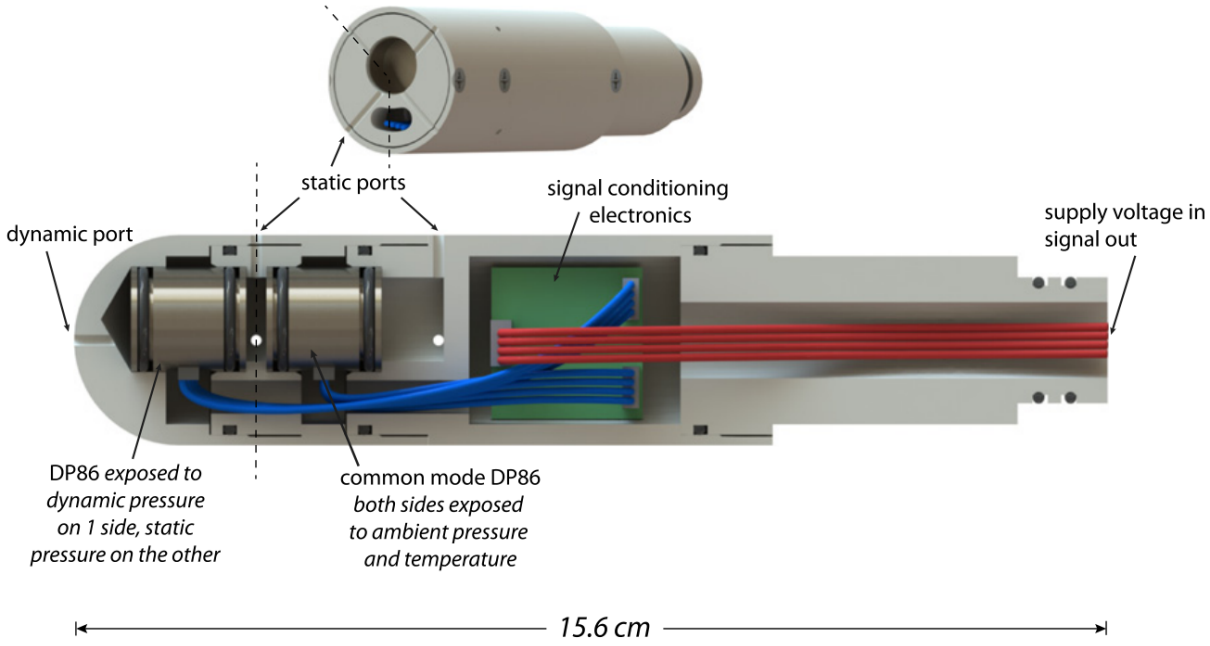
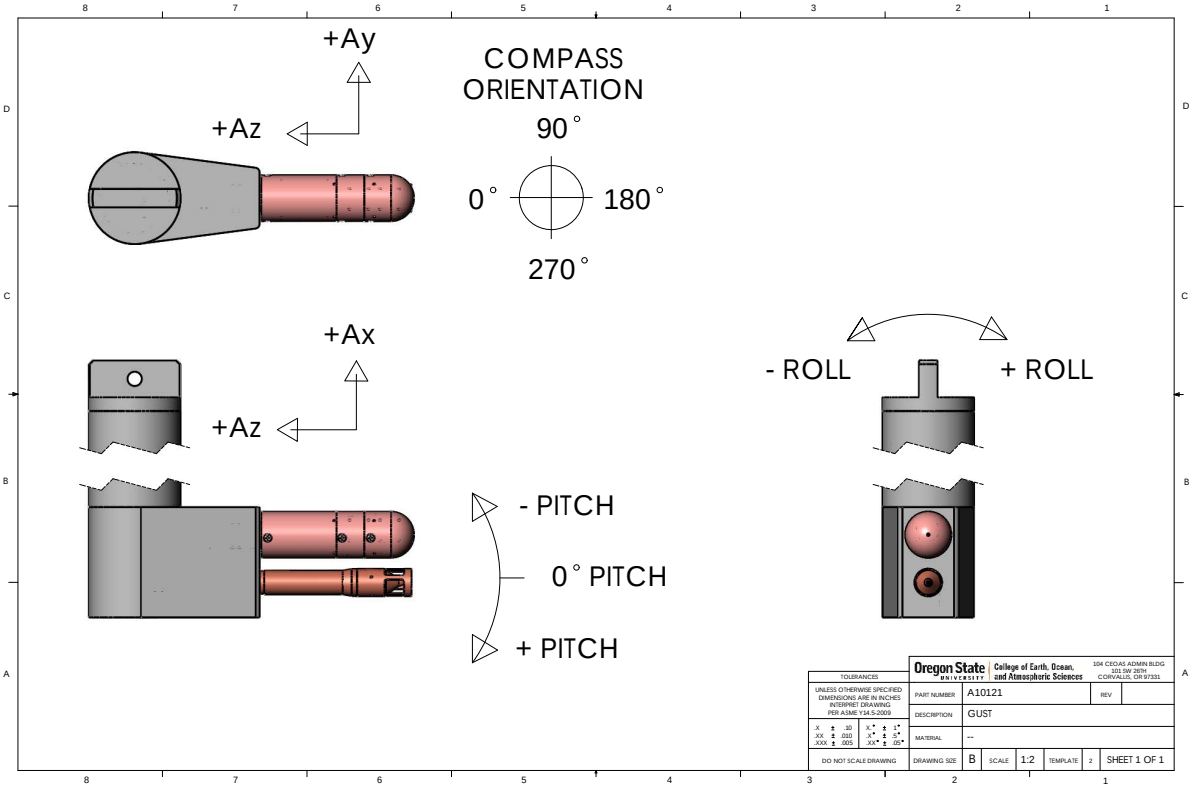


FIG. 2. Cross section of pitot-static tube showing details of the positioning of pitot-static and common mode differential pressure sensors. The cross sections ahead of the signal conditioning electronics are aligned with the dashed lines. The perspective is stylized to provide a clear view of components. (SOLIDWORKS drawing courtesy of Craig Van Appeldorn.)

D.3 gusT


+Ay
+Az
COMPASS
ORIENTATION
90°
0°
180°
270°
+Ax
+Az
- ROLL
+ ROLL
- PITCH
0° PITCH
+ PITCH
Oregon State University
College of Earth, Ocean, and Atmospheric Sciences
104 CEOAS ADMIN BLDG
101 SW 26TH
CORVALLIS, OR 97331
TOLERANCES
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES INTERPRET DRAWING PER ASME Y14.5-2009
DO NOT SCALE DRAWING
PART NUMBER
A10121
REV
DESCRIPTION
GUST
MATERIAL

DRAWING SIZE
B
SCALE
1:2
TEMPLATE
2
SHEET 1 OF 1

D.4 Multi gusT