# $\chi$pod and gusT processing package

Johannes Becherer, Sally Warner, Deepak Cherian

Last updated: December 11, 2017

# Contents

# Part I

# Calibration

# Chapter 1

# Compass

It can be confusing how the angles from the compass in $\chi$pods 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 $\chi$pod
2. How to convert from $\chi$pod 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 $\chi$pod

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

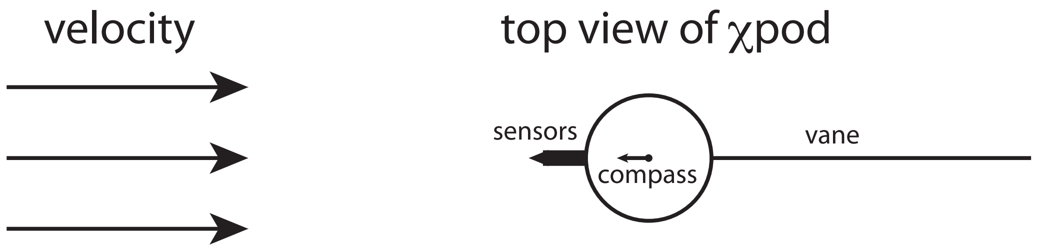


Figure 1.1: A perfectly installed compass should align with the sensors. The $\chi$pod vane will steer the sensors into the flow.

## 1.2 Converting from $\chi$pod 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.

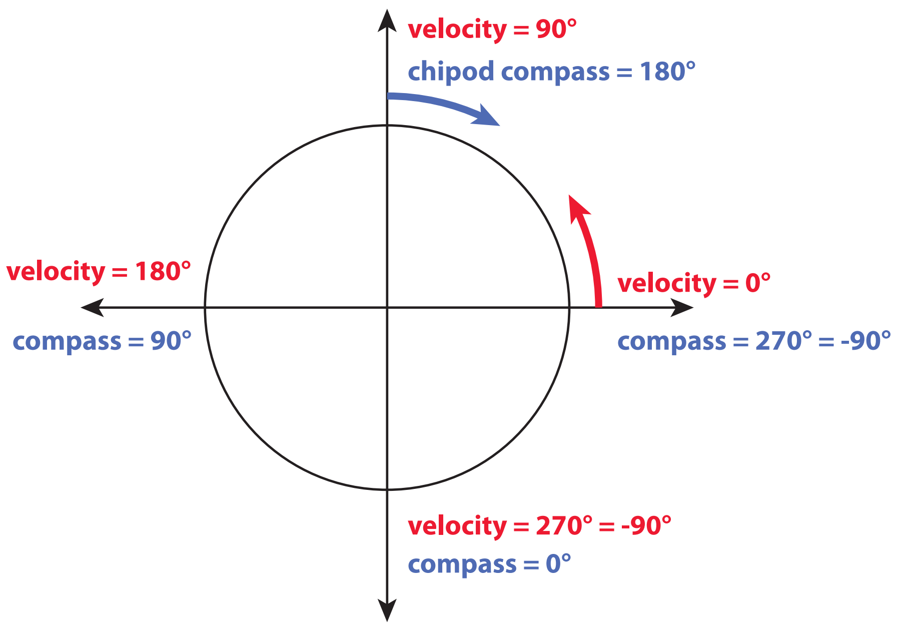


Figure 1.2: Angles of flow direction in vector notation (red) and the corresponding compass readings within the $\chi$pod. This assumes, of course, that the $\chi$pod sensors are vaned to face directly into the flow.

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

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

Alternatively, to convert from flow direction to compass:

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

## 1.3 How to find the compass offset

When the $\chi$pods are built, the compass may not be put into the $\chi$pod 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 $\chi$pod into the woods away from magnetic sources and point the sensors to the north. They wait a minute, then rotate the $\chi$pod 10° clockwise. They repeat this all the way around the circle until the $\chi$pod sensors are again pointing toward the north.

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

```
% load compass calibration file, which would have a name like CMP_00000000.UNT
[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.

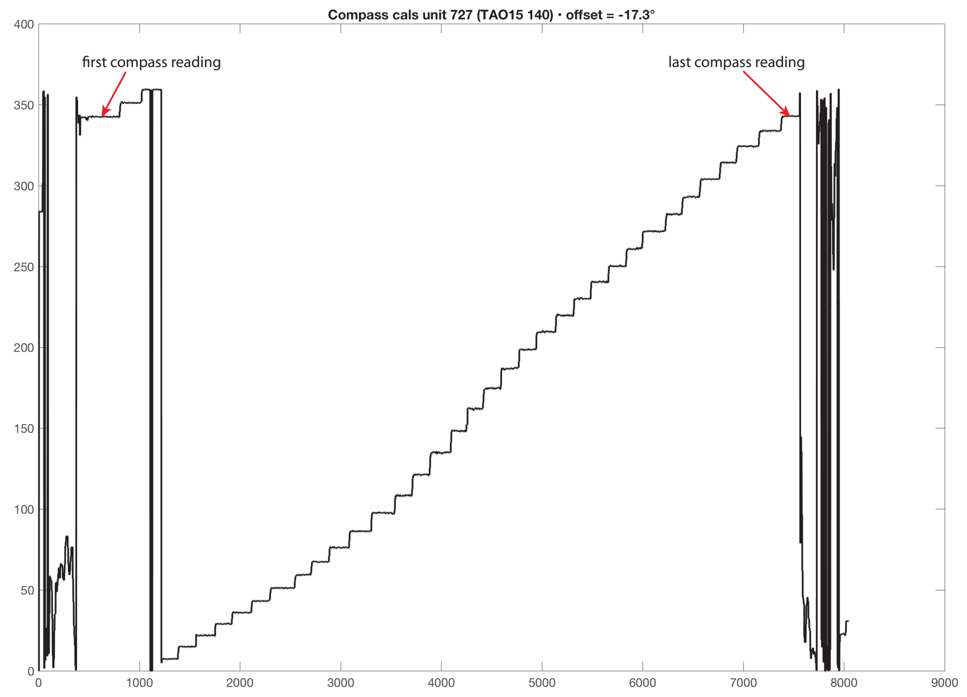


Figure 1.3: Data from $\chi$pod compass calibration. Compass angle in degrees between 0° and 360° on y-axis, unimportant time steps on x-axis. Use ginput to get first and last value. In this case, the offset was found to be 342.7° = -17.3°.

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 $\chi$pod 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!

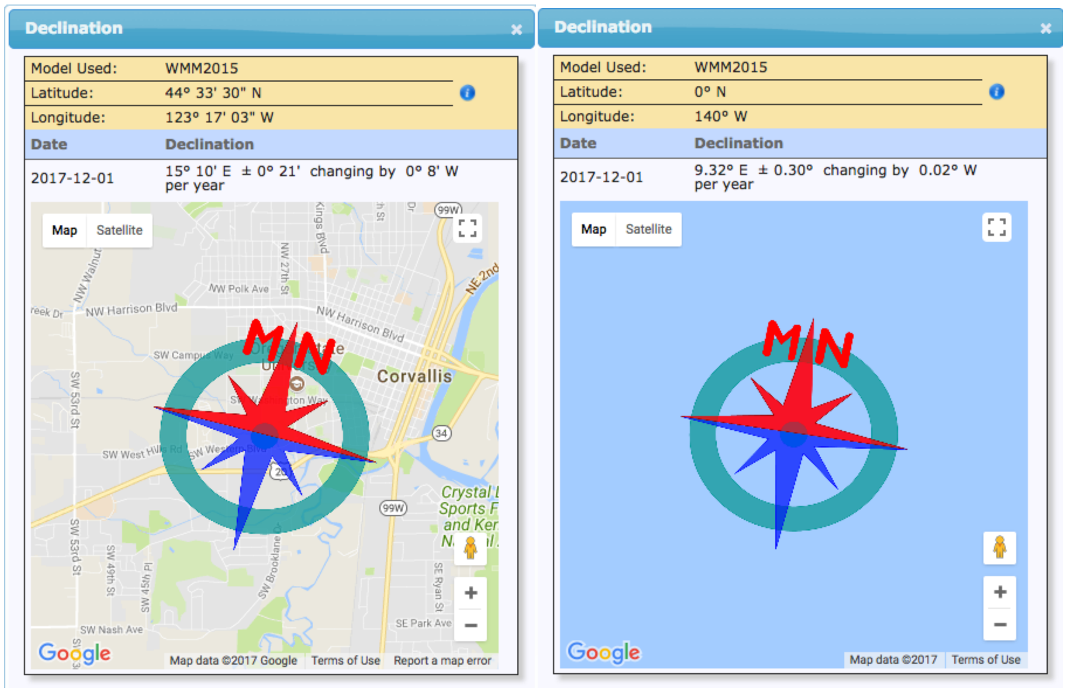
Look up the magnetic declinations in Corvallis where the instrument was calibrated and at the deployment location on the following website:

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 $\chi$pods 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
```



Declination

| | |
|---|---|
| Model Used: | WMM2015 |
| Latitude: | 44° 33' 30" N |
| Longitude: | 123° 17' 03" W |
| **Date** | **Declination** |
| 2017-12-01 | 15° 10' E ± 0° 21' changing by 0° 8' W per year |

Declination

| | |
|---|---|
| Model Used: | WMM2015 |
| Latitude: | 0° N |
| Longitude: | 140° W |
| **Date** | **Declination** |
| 2017-12-01 | 9.32° E ± 0.30° changing by 0.02° W per year |

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 $\chi$pod_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/#declination
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. $< 5°$).

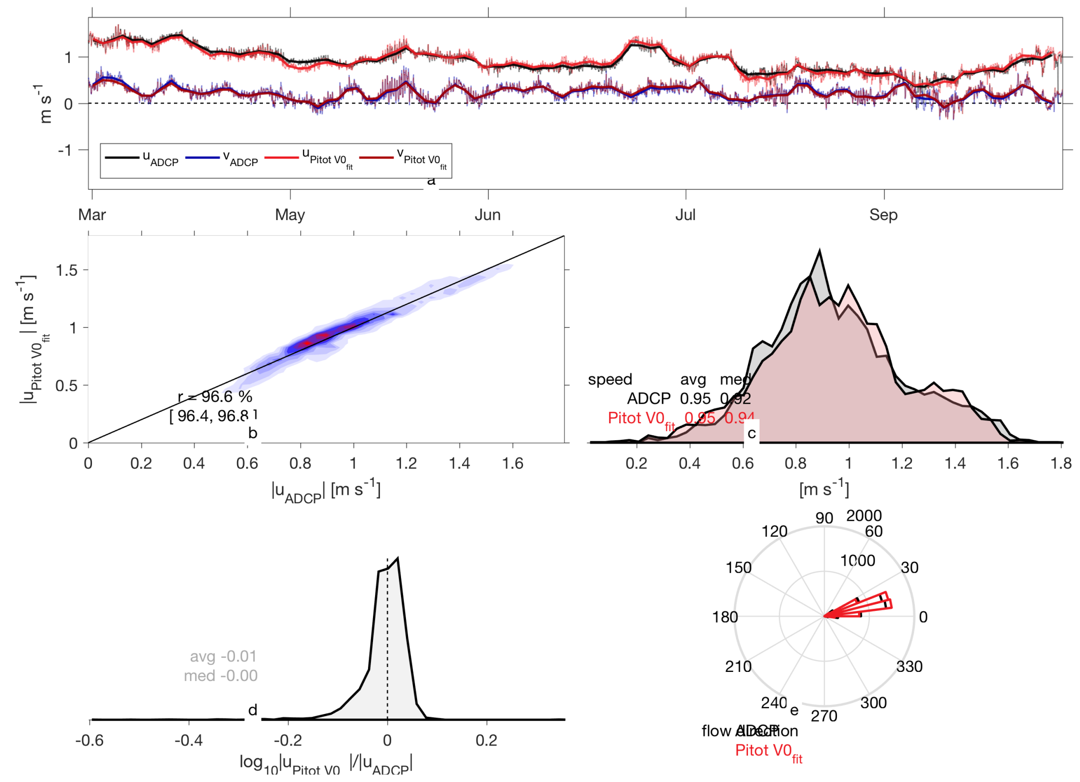


Figure 1.5: Output of calibrate_pitot when the compass has been correctly calibrated. ADCP and pitot velocities match closely.

## 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 $\chi$pod.

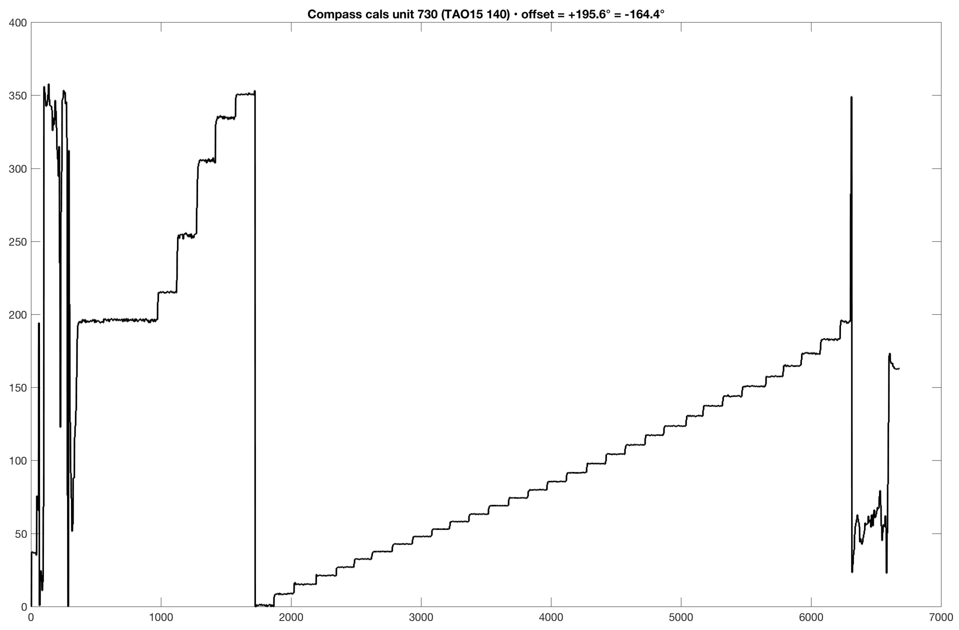


Figure 1.6: Example of bad compass calibrations. Not all steps are equal.

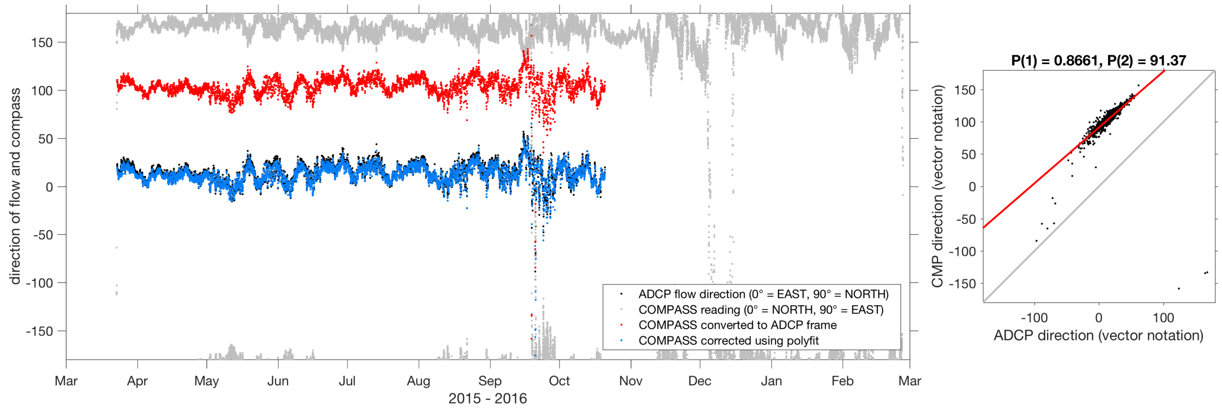


Figure 1.7: Manually finding the compass offset that makes the compass match the ADCP. Flow direction from ADCP (black dots), compass readings in heading coordinates (gray dots), compass readings converted to vector coordinates (red), compass readings in vector coordinates with an offset of 91.37° applied.

### 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 $\chi$pod processing by setting

pflag.master.use_compass = 0;

in main_driver.m. This will assume that the $\chi$pod 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

# Chapter 4

# main_driver

# Chapter 5

# combine_turbulence

This is the final step.

## 5.1 Masking criteria/thresholds

### 5.1.1 min_dTdz

- seabird SBE-37 datasheet says (http://www.seabird.com/sbe37si-microcat-ctd)
  - T is accurate to 2e-3 C
  - conductivity is accurate to 3e-3 psu (approx!)

i.e.,

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

### 5.1.2 time$_{\text{range}}$

### 5.1.3 T1death/T2death

### 5.1.4 nantimes{1,2,3}

### 5.1.5 avgwindow/avgvalid

### 5.1.6 deglitch$_{\text{window}}$ / deglitch$_{\text{nstd}}$

## 5.2 Useful Functions

### 5.2.1 ChooseEstimates

### 5.2.2 chiold (variable)

combine$_{\text{turbulence}}$ 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 , etc.

Example usage:

```
chi = chiold; % reset to unmasked structure
% apply 3 different thresholds
chi1 = ApplyMask(chi, abs(chi.dTdz), '<', 1e-4, 'T_z < 1e-4');
chi2 = ApplyMask(chi, abs(chi.dTdz), '<', 1e-3, 'T_z < 1e-3');
chi3 = ApplyMask(chi, abs(chi.dTdz), '<', 2e-3, 'T_z < 2e-3');

t0 = datetime(2016, 12, 10);
t1 = datenum(2016, 12, 12);
tavg = 600;

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);

% 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

# Part III

# Appendix

# Chapter 6

# Using the CEOAS MATLAB servers

## 6.1 SSH configuration

## 6.2 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

# Chapter 7

# Useful references

# Chapter 8

# git commands and workflows