

Appendix N - Navigation Processing Notes

Several sources of data are collected independently by the TowCam (e.g., Pictures, Navigation/Position, CTD data, Samples). In order to link these data streams into a final product that will allow you to better interpret your results, some processing is required. This processing is done primarily using a MATLAB script that has been developed to deal specifically with TowCam data. The various data streams are linked by time, which is recorded by each of the different sensors.

Recommended directory structure:

All of the data files from the different sources must be collected in one spot. It is recommended that copies of the data be used and that the originals be kept in a safe location and *backed up*. The following file structure should be used to facilitate data processing. Each tow (e.g., CT01, CT02,...CT0N) should have its own directory. Within that directory you should put the CTD data, a list of the image files, the ship navigation for that tow, and a matlab processing script specific to that tow.

- *ORI-810_processing*
 - *ShipNav*
 - *Pixlist*
 - *CT01*
 - *CT01_CTD* (folder)
 - *ORI-810_CT01_pixlist.txt* (text file)
 - *ORI-810_1001_nav.txt* (text file)
 - *NavMerge_CT01.m* (processing script)
 - *CT02*
 - *CT03*
 - *CT04*

The parent directory should be named after the cruise (e.g., *ORI-810*). Subordinate to that should be...

ShipNav folder that contains raw ship navigation files (e.g., *ORI_1006.col*), ship nav processing scripts (e.g., *ORINavProc.pl*), and processed nav files (e.g., *ORI_1006_nav.txt*, *ORI-810_CT01_nav.txt*).

Pixlist folder that contains the lists of images generated after each tow.

CT01 folder that contains CTD files within directories named for the tow (e.g. *CT01_CTD*), a file containing the list of image names (*ORI-810_CT01_pixlist.txt*), a copy of the processed ship navigation file for that tow (*ORI-810_1001_nav.txt*), and a processing script (*NavMerge_CT01.m*).

How the different files are generated is described below.

Ship Navigation:

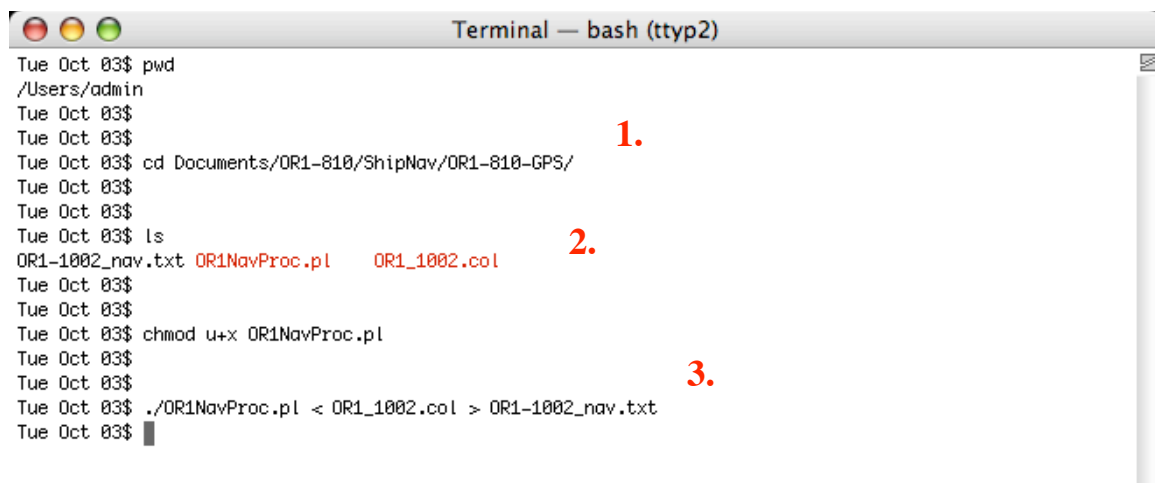
Ship navigation (GPS) files are obtained from the ships crew daily (or after each tow). The files are named as ship_date.col (e.g., *ORI_1002.col*). The data needs to be processed so that it is in the correct format for MATLAB. As an example, in the ship file

the time is stored as a string (hhmmss) and the script will change it to hh, mm, ss. The script is written in the programming language perl, and perl must be installed on the computer (it comes pre-installed on Mac computers).

***A print out of the perl script is attached at the end of this document.*

To **use** the perl script to process the data, place the ship navigation file, processing script (*OR1NavProc.pl*) in the same directory, and follow these instructions:

1. Using the Terminal program navigate to this directory (cd = change directory).
2. Make sure the script is executable (chmod u+x OR1NavProc.pl).
3. Run the script giving it an input file preceded by a '<' and an output file preceded by a '>' (./OR1NavProc < OR1_1002.col > OR1-1002_nav.txt)



```
Terminal — bash (tty2)
Tue Oct 03$ pwd
/Users/admin
Tue Oct 03$
Tue Oct 03$ cd Documents/OR1-810/ShipNav/OR1-810-GPS/
Tue Oct 03$
Tue Oct 03$
Tue Oct 03$ ls
OR1-1002_nav.txt OR1NavProc.pl OR1_1002.col
Tue Oct 03$
Tue Oct 03$ chmod u+x OR1NavProc.pl
Tue Oct 03$
Tue Oct 03$ ./OR1NavProc.pl < OR1_1002.col > OR1-1002_nav.txt
Tue Oct 03$
```

OR1_1002.col (before processing):

```
120.29113150 22.61451983 0.000 00000000 011115 OR1_81
120.29113517 22.61452233 0.000 20061002 011125 OR1_81
120.29113483 22.61452117 0.000 20061002 011135 OR1_81
120.29113450 22.61451850 0.000 20061002 011145 OR1_81
120.29113267 22.61451633 0.000 20061002 011155 OR1_81
120.29113217 22.61451367 0.000 20061002 011205 OR1_81
120.29112933 22.61451033 0.000 20061002 011215 OR1_81
120.29111933 22.61450967 0.000 20061002 011225 OR1_81
```

OR1-1002_nav.txt (after processing):

```
Year, Month, Day, Hour, Min, Sec, Long, Lat, Depth, SOM
0000, 00, 00, 01, 11, 15, 120.29113150, 22.61451983, 0.000, 4275
2006, 10, 02, 01, 11, 25, 120.29113517, 22.61452233, 0.000, 177085
2006, 10, 02, 01, 11, 35, 120.29113483, 22.61452117, 0.000, 177095
2006, 10, 02, 01, 11, 45, 120.29113450, 22.61451850, 0.000, 177105
2006, 10, 02, 01, 11, 55, 120.29113267, 22.61451633, 0.000, 177115
2006, 10, 02, 01, 12, 05, 120.29113217, 22.61451367, 0.000, 177125
2006, 10, 02, 01, 12, 15, 120.29112933, 22.61451033, 0.000, 177135
2006, 10, 02, 01, 12, 25, 120.29111933, 22.61450967, 0.000, 177145
```

*The last column, **SOM**, is a unique time ID for each navigation point that is the number of the seconds from the first of the month ($\text{Day} \times 3600 \times 24 + \text{Hour} \times 3600 + \text{Minute} \times 60 + \text{Second}$).

Making the Pixlist file:

The MATLAB processing scripts require a list of the file names of all the pictures taken by the camera. This list should start with the first bottom picture and end with the last bottom picture. You will need to generate this list and cut off files that are not pictures of the bottom.

- Using the Terminal program, navigate to the location where the images are stored.
- Type the command 'ls' followed by a '>' and an output file name. This will pipe the list generated by 'ls' into the output file.
 - example: `ls > OR1-810_CT02_pixlist.txt`
- Look at the images to see where the first and last bottom pictures are and cut off file names that are earlier and later. ****Check the end of the pixlist file to make sure that the filename of the list itself (e.g., OR1-810_CT02_pixlist.txt) is not included.**

MATLAB processing script:

The MATLAB processing script will allow you to link the ship navigation, CTD data, and image data. Linking is achieved by using the time stamp that is present in each record. The script will allow you to cut-off excess data, filter the data, produce plots for each tow, and output a series of data files containing the linked data. The MATLAB script is divided into sections (i.e., cells) that contain a block of code that conducts a specific task. Below the function of each cell is described and notes are provided to explain how this is achieved. Almost all of the functions used in this script are native MATLAB commands and descriptions of these can be found by typing 'help <command>' or 'doc <command>' (e.g., *doc textread*). As each camera tow is different, new challenges will arise each time data is processed. Although the code is flexible and designed to meet these challenges, invariably some modification of the code will be required. It is **highly recommended** that a new copy of the processing script be used for each tow and saved along with the data. This will allow data to be reprocessed later without having to troubleshoot the code again. Notes describing common errors and how they may be corrected are marked by ******.

******A complete, documented copy of the code is included at the end of this document.

%% Clear the memory

This section clears any stored data from memory and closes any open figure windows.

%% Paths to data

Paths to the location of input files are hard coded. This is one reason it is useful to create a new version of the code for each camera tow.

**The paths need to be changed for each tow. If the next section (load data) fails, be sure to check that the path names (including all dashes '-' and underscores '_' are correct.

%% Load data

Data is loaded using the function *textread*. The format of each data element is indicated in common latex syntax where %f is a floating point number, %s is a string, %d is an integer, etc. Arguments after the textread command include the delimiter between data elements, the number of headerlines above the data.

%% Preallocating memory

Memory is preallocated for variables that will be used later to speed up processing. Variables are created with the proper size (number of elements) and filled with zeroes.

%% Change times to second of day

A unique ID is created to link data elements from different input files. The ID is the number of seconds since the beginning of the month and is calculated as:
(Day*3600*24 + Hour*3600 + Minute*60 + Second).

%% Sample CTD data at image times

Variables are created that subsample the CTD data, which is recorded at 1 Hz, at a time interval dictated by the time of the photos.

%% Cut off unnecessary data

Data is recorded in the ship navigation, CTD, and image records over a time window that exceeds that of the time the camera is on the bottom.

**Check the figures that are generated here to see that the cut data looks like the right section of the total data. If it is incorrect, make sure that the pixlist file really covers the first to last bottom pictures.

%% Calculate layback

Layback, the distance between the ship and the camera system is difficult to determine. In general, a constant layback (100-300 m) can be applied to the ship position in the direction opposite that which the ship is traveling. When the ship makes tight turns, a constant layback direction does not work well. The ships track often has small jogs as it tries to maintain course. In reality, small jogs in the ship's course are not felt by the camera system and the camera track is likely much smoother than the ship track. However, accurately and reliably removing these jogs can not be done automatically

To determine the layback direction a MATLAB command that determines the azimuth between successive ship navigation points is used. Due to course jobs, however, the point-to-point azimuth is a very noisy record, so a filtered version of this is used to determine the layback position. The matlab command 'reckon' is then used to find a new layback position for each ship position.

You should be aware of several options when calculating layback:

1. Using the aforementioned filtered course data to determine layback.
2. Using a constant layback azimuth in place of the course (or filtered course data).
3. Using no layback at all.

****The latter is advised if the ship made many turns during the tow.**

%% Interpolate ship and layback nav to 1 Hz

With layback applied to the ship's navigation, the data are interpolated to 1 Hz, the same frequency as the CTD data. The MATLAB command 'interp1' uses ship nav time, ship nav position, and CTD time to achieve this.

%% Calculate distances between nav points

The distance along the track is an effective way to plot data such as seafloor depth and temperature records. This is especially true if the ship speed was not constant during the tow. Distances are calculated using the simple pythagorean theorem.

%% Sample data

Sample times are used to pull CTD data at sample times.

%% Filter depth data

Seafloor depth can be a noisy record, so data is filtered using a low-pass butterworth filter. A function (not a native MATLAB command) is used to do this. The function called lowfilt.m must be on the computer and in the MATLAB path!!

%% Plot seafloor profile and samples

The next 5 sections make plots of the data. The first is a plot of the seafloor depth and sample locations. Sample names are shown on this plot, but the code that executes writing text on the plot will have to be adjusted for each tow. Depending on the profile, some of the sample labels work better above the profile and others below. In the last 2 'for' loops in this section change the code 'for o = [1,4,5]' to adjust this. Also in these loops change the code 'SAMwdep(o)+80' so that the labels appear at the correct height relative to the seafloor profile.

%% Plot depth as a function of time

Self-explanatory.

%% Multi-axis plot – temp

Self-explanatory.

%% Multi-axis plot – sal (developed by Yee Ching)

Self-explanatory.

%% Depth, temperature, salinity, turbidity

Self-explanatory.

%% Making a sample file

The first output file contains comma separated data at each sample time.

%% Making new flash file

The second output file that is created has the following elements for each flash record (image) during the tow:

DATE,TIME,LAT,LON,UTMx,UTMy,SHIPLAT,SHIPLON,DEPTH,ALT,WDEP,TEMP,TURB,SAL,HLINK

This information is contained in a headerline at the beginning of the file.

Lat,Lon,UTMx,and UTMy are position of the camera using layback (if used). SHIPLAT and SHIPLON are the original ship's position. The last column, HLINK, is the name of the image path preceded by 'path:/' . When imported in Arc, this column can be used to link the pictures to the map. The data is comma separated.

%% Saving stuff

The last file created is a .mat file. This is a binary file that is only read by MATLAB. All the variables created during the processing are stored in this file. This data can be loaded for further processing without re-running the processing script.

```
#!/usr/bin/perl -
#
# OR1NavProc.pl
# Written by A. Soule, 27 Sept. 2006
# contact ssoule@whoi.edu for further instructions
#
# This perl script will parse the navigation files produced by the OR1 to a
# format suitable for use in TowCam processing scripts. A unique, time-based
# ID is generated for each GPS point to allow merging of the ship navigation
# with TowCam imagery, sampling, and CTD data.
#
# Place the script in a folder with ascii navigation files obtained from OR1.
# Make sure the script is executable. If it is not, type the following command
# in the terminal window where the script resides '>> chmod u+x OR1NavProc.pl'.
#
# Usage: At the command prompt, call the script and give the name of the input
# file preceeded by a less than sign and the name of the output file preceeded
# by the greater than sign.
#
# prompt>> ./OR1NavProc.pl < 'input file' > 'output file'

$line = <>;
while ($line = <>) {          #perform parsing for each line in the file
    chomp($line);            #remove any 'end of line' characters

    ($A, $B, $C, $D, $E, $F) = split(" ", $line); #split data into variables
                                                #using a 'space' delimiter

    $LON = $A;                #longitude in decimal degrees
    $LAT = $B;                #latitude in decimal degrees
    $DEP = $C;                #heading in decimal degrees
    $DATE = $D;               #date in format: YYYYMMDD
    $TIME = $E;               #time in format: hhmmss

    $YY = substr($D,0,4);     #sample date string for year
    $MM = substr($D,4,2);     #sample date string for month
    $DD = substr($D,6,2);     #sample date string for day

    $hh = substr($E,0,2);     #sample time string for hour
    $mm = substr($E,2,2);     #sample time string for minute
    $ss = substr($E,4,2);     #sample time string for second

    #second of day provides a unique id for each gps measurement that can be
    #linked to CTD data. second of month removes any problems for tows that
    #cross a gmt day (NOTE: this will fail if a tow crosses a gmt day from the
    #end of one month to the beginning of the next month!!!)

    $SOD1 = ($hh*3600) + ($mm*60) + ($ss); #calculate second of day
    $SOD2 = ($DD)*24*3600;                #seconds in the month until current day
    $SOM = $SOD1+$SOD2;                   #second of month

    #print the data to a new file.

    print "$YY $MM $DD $hh $mm $ss $LON $LAT $DEP $SOM \n";
}
}
```

Contents

- [Layback code for NTU-CGS TowCam](#)
- [Paths to data](#)
- [Load data](#)
- [Preallocating memory](#)
- [Change times to second of day](#)
- [Sample CTD data at image times](#)
- [Cut off unnecessary data](#)
- [Calculate layback](#)
- [Plot course \(azimuth\) data](#)
- [Interpolate ship and layback nav to 1 Hz](#)
- [Calculate distances between nav points](#)
- [Sample data](#)
- [Filter depth data](#)
- [Plot seafloor profile and samples](#)
- [Plot depth as a function of time](#)
- [Multi-axes plot-temp](#)
- [Multi-axes plot-sal](#)
- [Depth, temperature, salinity, turbidity](#)
- [Making a sample file](#)
- [Making new flash file](#)
- [Saving stuff](#)

Layback code for NTU-CGS TowCam

Adam Soule, 01 October 2006

Variable naming conventions: SOURCE + variable + description

SOURCES: SNAV = ship nav, CTD = ctd, BTL = bottle, PIX = picture, LB = layback nav, SAM = sample. descriptions: C = cut, i = interpolated, F = filtered

```
clear all
close all
clc
```

Paths to data

Paths are hard-coded and need to be modified for different computers, cruises, deployments, etc.

```
SN = 'C:\Towcam\Cruise\OR1-2006\process\CT07\OR1-1006_nav.txt';
CT = 'C:\Towcam\Cruise\OR1-2006\process\CT07\CT07_CTD\CT07_ctd.txt';
BT = 'C:\Towcam\Cruise\OR1-2006\process\CT07\CT07_CTD\CT07_core.txt';
PI = 'C:\Towcam\Cruise\OR1-2006\process\CT07\OR1-810-CT07-pixlist.txt';
```

Load data

data loaded by textread function based on the syntax and structure within each datafile. Updates for loading will read as shown below code block.

```
disp('Loading Ship Navigation...')
[SNAVyear,SNAVmonth,SNAVday,SNAVhr,SNAVmin,SNAVsec,...
 SNAVlon,SNAVlat,SNAVdep,SNAVsod]= textread(SN,...
 '%d %d %d %d %d %f %f %f %f','headerlines',1,...
 'delimiter',' ');
disp(['Done!'; ' '])

disp('Loading CTD data...')
[CTDdate,CTDtime,CTDprSM,CTDtemp,CTDc0m,CTDupoly,CTDturb,...
 CTDdep,CTDalt,CTDupoly,CTDsalsal,CTDflag]=...
 textread(CT,'%s %s %f %f %f %f %f %f %f %f',...
 'headerlines',1,'delimiter',' ');
```



```

CTDwdep = -(CTDdep+CTDalt);
disp(['Done!'; ' '])

disp('Loading Bottle data...')
[BTLnum,BTLdate,BTLtime,BTLprSM,BTLtemp,BTLcond,BTLpoly,...
 BTLturb,BTLdep,BTLalt,BTLpoly1]=...
textread(BT,'%d %s %s %f %f %f %f %f %f %f %f',...
 'headerlines',1,'delimiter',' ');
disp(['Done!'; ' '])

disp('Loading Pixlist data...')
PIXtime = textread(PI,'%s');
disp(['Done!'; ' '])

Loading Ship Navigation...
Done!

Loading CTD data...
Done!

Loading Bottle data...
Done!

Loading Pixlist data...
Done!

```

Preallocating memory

Memory is preallocated for all arrays that will be grown by indexing to speed processing time.

```

CD = zeros(1,length(CTDtime));
CH = zeros(1,length(CTDtime));
CM = zeros(1,length(CTDtime));
CS = zeros(1,length(CTDtime));
PD = zeros(1,length(PIXtime));
PH = zeros(1,length(PIXtime));
PM = zeros(1,length(PIXtime));
PS = zeros(1,length(PIXtime));
BD = zeros(1,length(BTLdate));
BH = zeros(1,length(BTLdate));
BM = zeros(1,length(BTLdate));
BS = zeros(1,length(BTLdate));

```

Change times to second of day

```

disp('Changing time (hh:mm:ss) to second of day...')

for k = 1:length(CTDtime)
    tt = CTDtime{k};
    gg = CTDate{k};
    CD(k) = str2double(gg(1:2));
    CH(k) = str2double(tt(1:2));
    CM(k) = str2double(tt(4:5));
    CS(k) = str2double(tt(7:8));
end

CTDsod = (CD.*3600*24)+(CH.*3600)+(CM.*60)+(CS);

SNAVsod = floor(SNAVsod);

for j = 1:length(PIXtime)
    aa = PIXtime{j};
    PD(j) = str2double(aa(9:10));
    PH(j) = str2double(aa(12:13));
    PM(j) = str2double(aa(15:16));
    PS(j) = str2double(aa(18:19));
end

PIXsod = (PD.*3600*24)+(PH.*3600)+(PM.*60)+(PS);

disp(['Done!'; ' '])

for g = 1:length(BTLdate)
    bb = BTLdate{g};
    aa = BTLtime{g};
    BD(g) = str2double(bb(1:2));
    BH(g) = str2double(aa(1:2));
    BM(g) = str2double(aa(4:5));
    BS(g) = str2double(aa(7:8));
end

```

```
BTLsod = (BD.*3600*24)+(BH.*3600)+(BM.*60)+(BS);
```

```
Changing time (hh:mm:ss) to second of day...
Done!
```

Sample CTD data at image times

This function replaces the flash file created by the seabird software. Occasionally the flash record will become misaligned with the image times and this ensures that CTD data is sampled at the correct intervals.

```
for k = 1:length(PIXsod)
    aa(k) = find(CTDsod==PIXsod(k));
end
FLAsod=CTDsod(aa);
FLAdate=CTDdate(aa);
FLAwdep=CTDwdep(aa);
FLAtime=CTDtime(aa);
FLAdate=CTDdate(aa);
FLAdep=CTDdep(aa);
FLAalt=CTDalt(aa);
FLAtemp=CTDtemp(aa);
FLAturb=CTDturb(aa);
FLAsal=CTDsalsal(aa);
```

Cut off unnecessary data

CTD data is collected during the descent and ascent. This function will subsample the CTD data 10 seconds prior to and after images start and stop, OR over an interval selected from a GUI.

```
plot(CTDsod,CTDwdep)
% [x,y] = ginput(2); % uncomment to select the interval
% through a GUI and comment next
% line.

x = [PIXsod(1)-10,PIXsod(end)+10];
cc = find(CTDsod<x(2)&CTDsod>x(1));
clear x y

CTDsodC = CTDsod(cc);
CTDwdepC = CTDwdep(cc);
CTDtempC = CTDtemp(cc);
CTDsalsalC = CTDsalsal(cc);
CTDtimeC = CTDtime(cc);
CTDturbC = CTDturb(cc);
CTDdateC = CTDdate(cc);
CTDdepC = CTDdep(cc);
CTDaltC = CTDalt(cc);

figure
plot(CTDsodC,CTDwdepC)

be = CTDsodC(1);
en = CTDsodC(end);

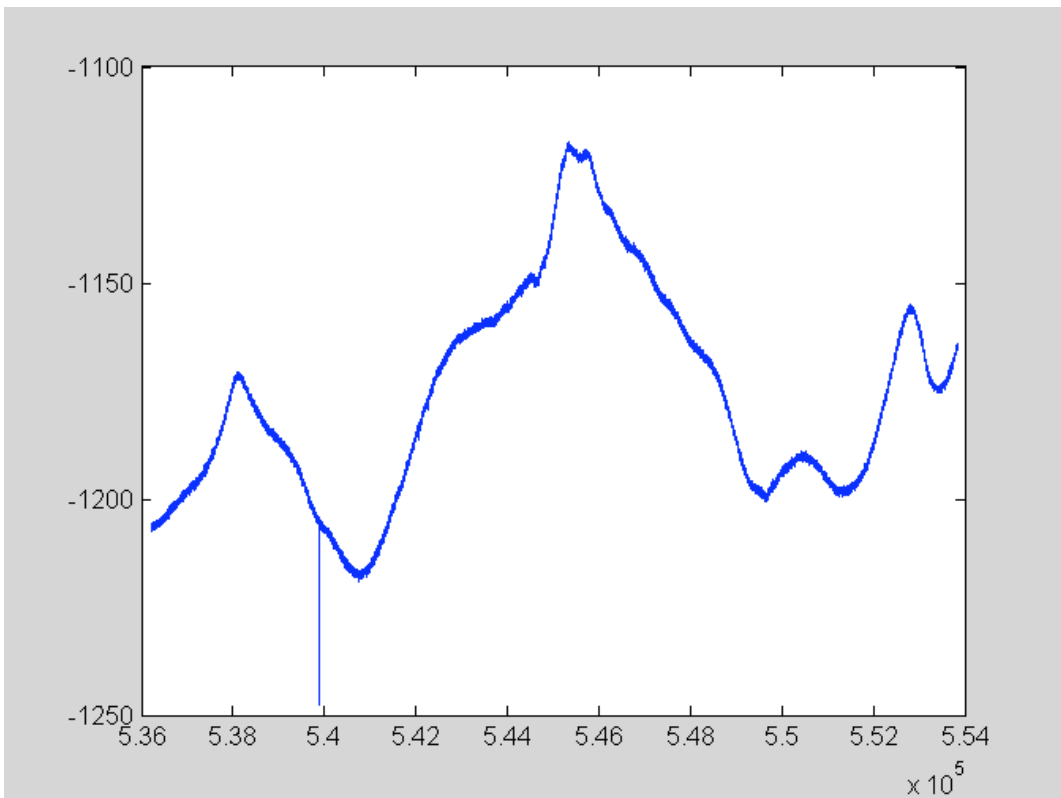
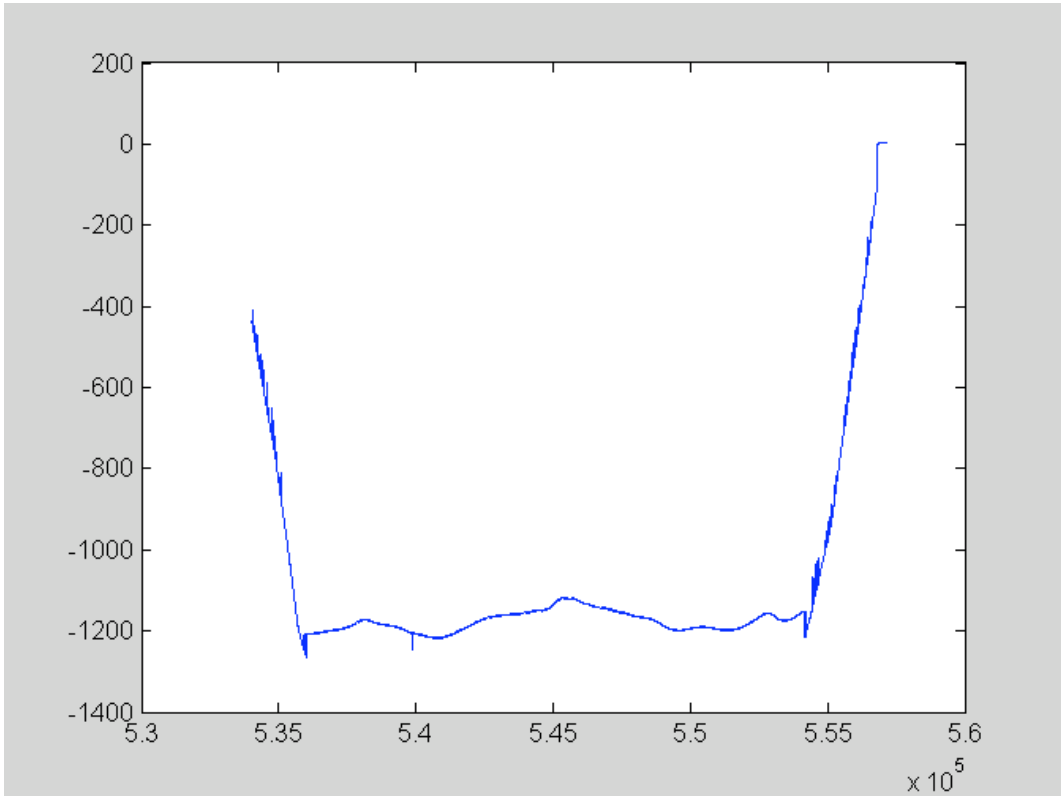
temp = find(SNAVsod<=be); %%% fix this!!
x(1) = temp(end);
x(2) = find(SNAVsod>=en,1);
cs = [x(1):x(2)];
clear x

SNAVsodC = SNAVsod(cs);
SNAVlatC = SNAVlat(cs);
SNAVlonC = SNAVlon(cs);
[SNAVx,SNAVy,zone] = ll2utm(SNAVlatC,SNAVlonC,1);

FLAsodC = FLAsod;
FLAwdepC = FLAwdep;
FLAdateC = FLAdate;
FLAtimeC = FLAtime;
FLAtempC = FLAtemp;
FLAturbC = FLAturb;
FLAdepC = FLAdep;
FLAaltC = FLAalt;
FLAsalsalC = FLAsalsal;

PIXtimeC = PIXtime;
PIXsodC = PIXsod;
```

UTM zone is 50
Central Meridian Longitude is 117.0000



Calculate layback

```

lb = km2deg(0.01);

[course,dist]=legs(SNAVlatC,SNAVlonC);
course = [course(1);course];
courseF = lowfilt2(course);

%Options:
% If layback does not look right you can:
% a) change the layback from "courseF" to a constant number (line 207).
% b) change the amount of layback (line 194)to a very small number 0.001.

for g = 1:length(course)
    [LBlat(g),LBlon(g)] = reckon(SNAVlatC(g),SNAVlonC(g),...
    lb,(315-180));
end

[LBx,LBy,zone]=ll2utm(LBlat,LBlon,1);

UTM zone is      50
Central Meridian Longitude is  117.0000

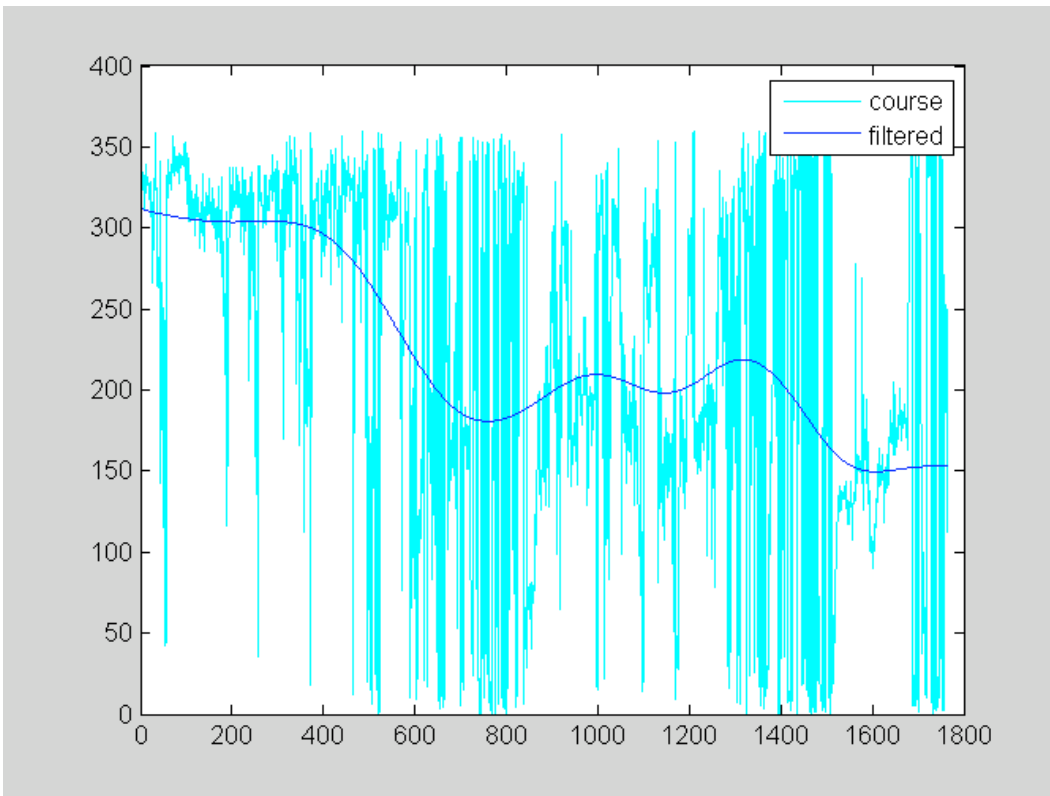
```

Plot course (azimuth) data

```

figure
plot(course,'c')
hold on
plot(courseF,'b')
legend('course','filtered')

```



Interpolate ship and layback nav to 1 Hz

Interpolate position data to 1 hz by time.

```

SNAVxi = interp1(SNAVsodC,SNAVx,CTDsodC);
SNAVyi = interp1(SNAVsodC,SNAVy,CTDsodC);

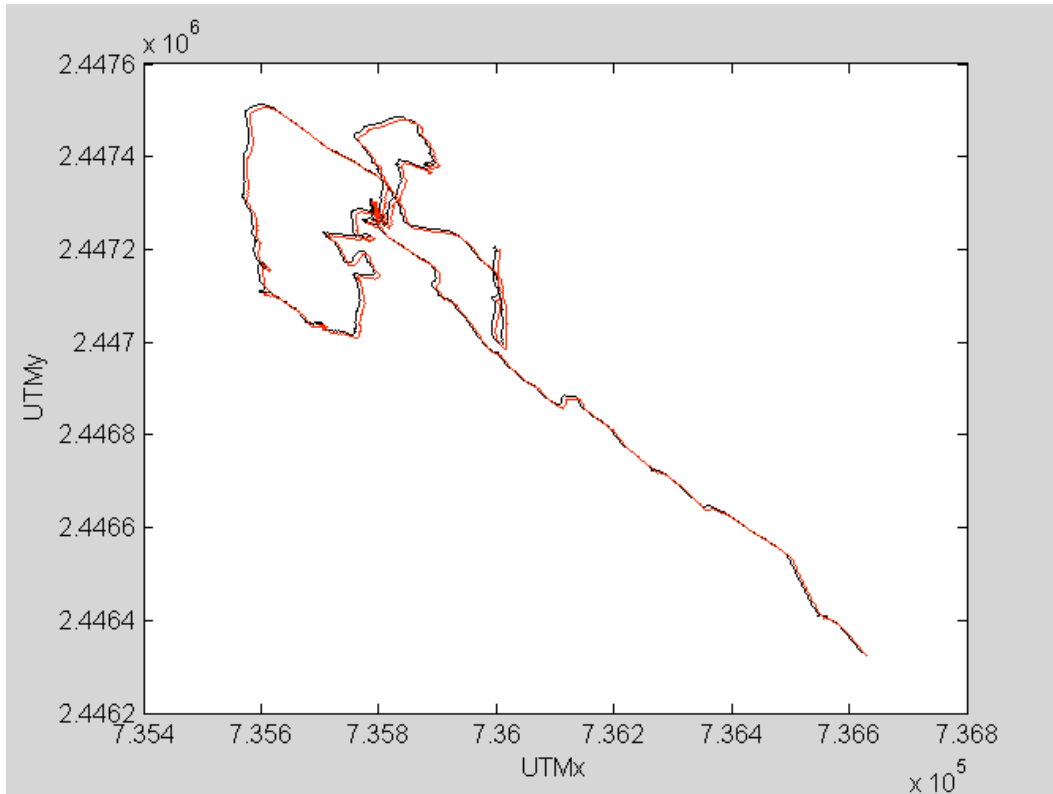
[SNAVloni,SNAVlati] = utm2ll(SNAVxi,SNAVyi,zone);

LBxi = interp1(SNAVsodC,LBx,CTDsodC);
LByi = interp1(SNAVsodC,LBy,CTDsodC);

```

```
[LBloni,LBlati] = utm2ll(LBxi,LByi,zone);

figure
plot(SNAVxi,SNAVyi,'-k')
hold on
plot(LBxi, LByi, '-r')
xlabel('UTMx');
ylabel('UTMy');
print -depsc OR1-810_CT07_layback
print -dpdf OR1-810_CT07_layback
```



Calculate distances between nav points

```
CTDdist = 0;
for t = 2:length(LBxi)
    Ac = (LBxi(t)-LBxi(t-1))^2;
    Bc = (LByi(t)-LByi(t-1))^2;
    Cc = (Ac+Bc)^(1/2);
    CTDdist(t) = Cc+CTDdist(t-1);
end

% create a buffer (200 m to either side) for plotting

d1 = CTDdist(1)-200;
d2 = CTDdist(end)+200;
```

Sample data

Subsample data at sample (bottle) times

```
for u = 1:length(BTLsod)
    SampID(u) = find(CTDsodC==BTLsod(u));
end

% SampID = SampID(2:end);

SAMwdep = CTdwdepC(SampID);
SAMdist = CTDdist(SampID);
% SAMtime = BTLtime;
SAMtime = BTLtime;
SAMdate = CTDDateC(SampID);
```

```
SAMlat = LBlati(SampID);
SAMlon = LBloni(SampID);
SAMx = LBxi(SampID);
SAMY = LByi(SampID);
```

Filter depth data

```
CTDwdepF = lowfilt(CTDwdepC);
```

Plot seafloor profile and samples

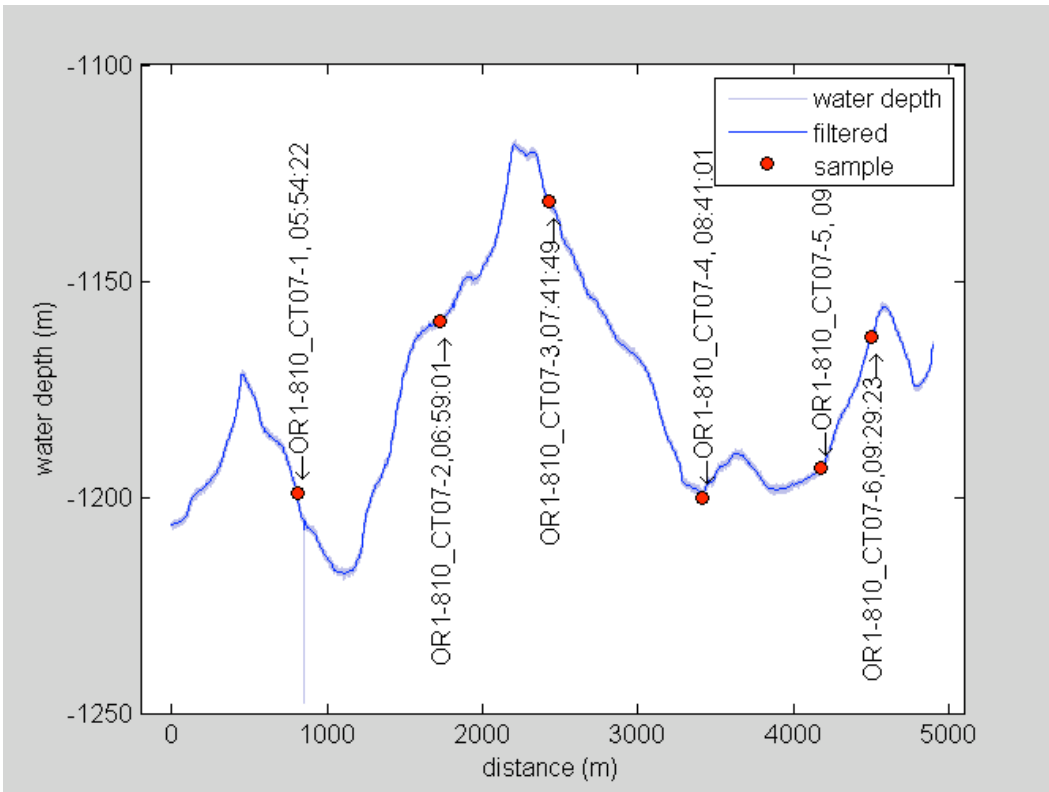
```
figure
plot(CTDdist,CTDwdepC,'Color',[0.7 0.7 0.9])
hold on
plot(CTDdist,CTDwdepF,'-b')
xlim([d1 d2])
plot(SAMdist,SAMwdep,'ok','MarkerFaceColor','r')
legend('water depth','filtered','sample')
xlabel('distance (m)')
ylabel('water depth (m)')
uu = ylim;

samplename = 'OR1-810_CT07-'; % Change sample name for different cruise
                             % and tows

% above
for o = [1,4,5]
    text(SAMdist(o),(SAMwdep(o)+80),['\leftarrow', samplename, num2str(o)...
    ', ' SAMtime{o}], 'HorizontalAlignment','right', 'rotation',90)
end

% below
for o = [2,3,6]
    text(SAMdist(o),(SAMwdep(o)-80),[samplename, num2str(o) ', ' SAMtime{o}...
    '\rightarrow'], 'HorizontalAlignment','left', 'rotation',90)
end

print -depsc OR1-810_CT07_dist_depth
print -dpdf OR1-810_CT07_dist_depth
```



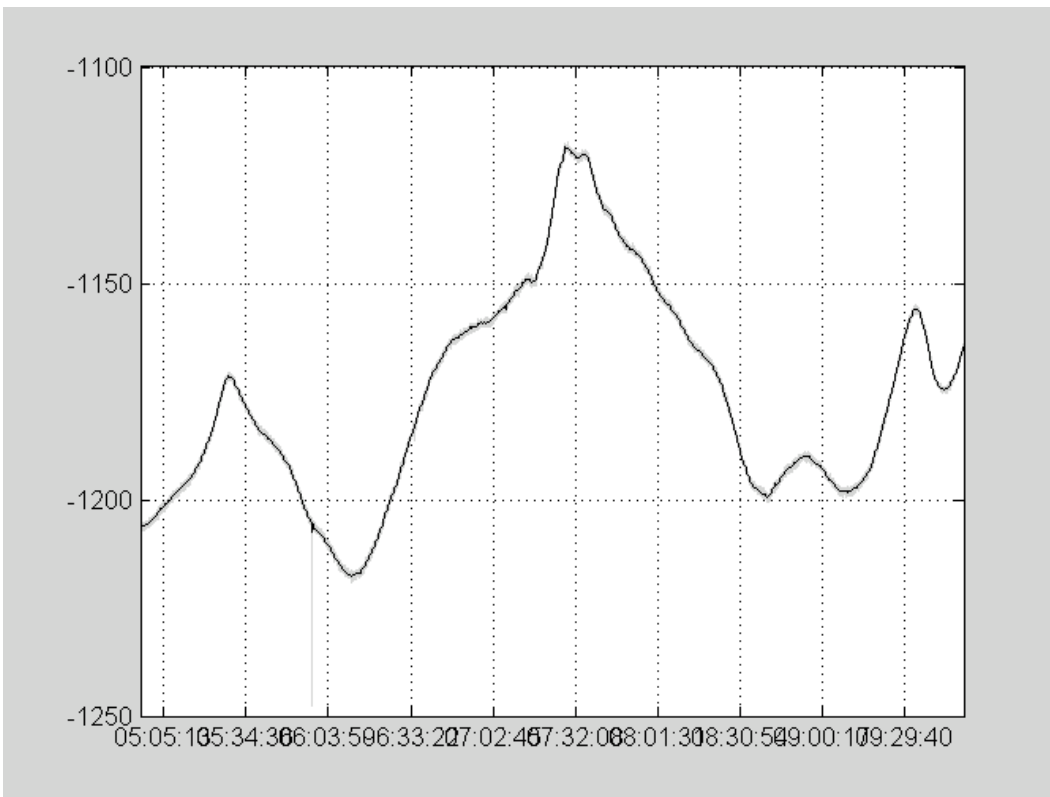
Plot depth as a function of time

```

dd1 = CTDSodC(1);
dd2 = CTDSodC(end);
dd3 = 0;
for nn = 500:(round(length(CTDSodC)/10)):length(CTDSodC)
    dd3 = dd3 + 1;
    plottime(dd3,:) = CTDtimeC{nn};
end
ptime = cellstr(plottime);

figure
plot(CTDSodC,CTDwdepC,'Color',[0.8 0.8 0.8])
hold on
plot(CTDSodC,CTDwdepF,'k')
% hold on
% plot(CTDSodC,-CTDcdepC,'b')
xlim([dd1 dd2])
tk = 500:(round(length(CTDSodC)/10)):length(CTDSodC);
tkmk = CTDSodC(tk);
set(gca,'XTick',tkmk)
set(gca,'XTickLabel',ptime)
grid on
print -depsc OR1-810_CT07_time_depth
print -dpdf OR1-810_CT07_time_depth

```

**Multi-axes plot-temp**

```

lowT = min(CTDtempC);
highT = max(CTDtempC);

figure
x1 = CTDDist; y1 = CTDwdepC;
x2 = CTDDist; y2 = CTDtempC;
h1 = line(x1,y1,'Color','k');
xlabel('Distance along tow (m)')
ylabel('Seafloor depth (m)')
ax1 = gca;
set(ax1,'XColor','k','YColor','k');
grid on
ax2 = axes('Position',get(ax1,'Position'),...
          'XAxisLocation','top',...
          'YAxisLocation','right',...
          'Color','none',...
          'XColor','r','YColor','r');

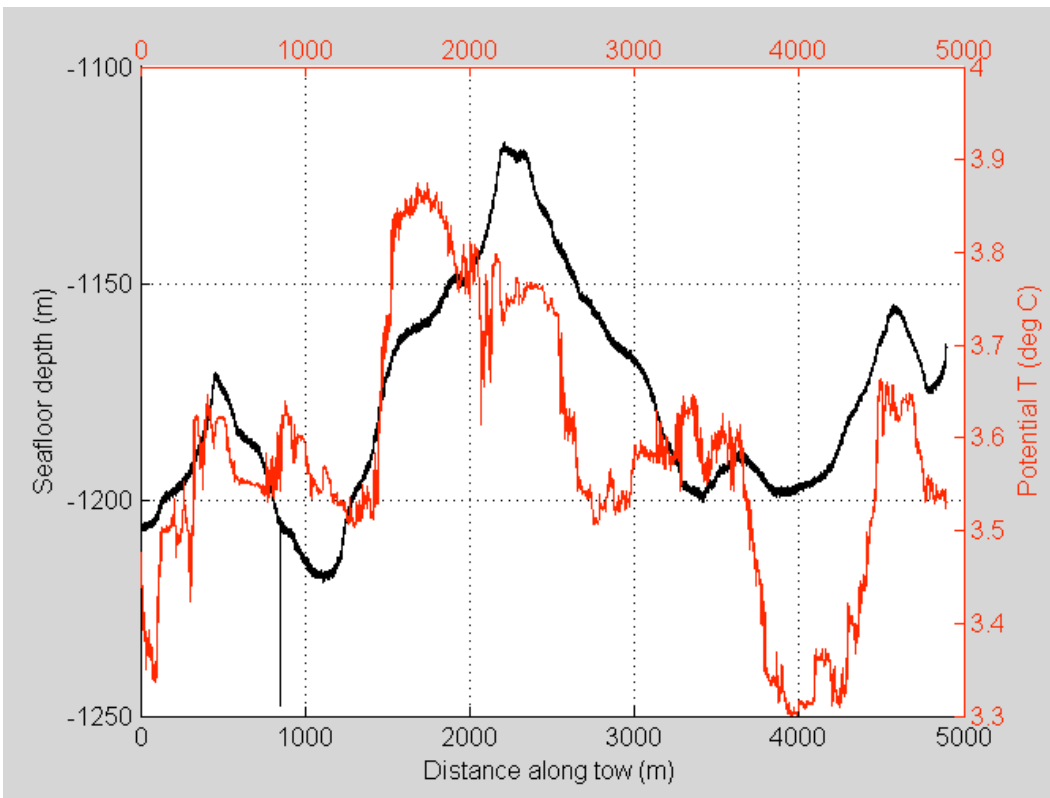
```

```

h12 = line(x2,y2,'Color','r','Parent',ax2);
% ylim([4.0 4.15])
ylabel('Potential T (deg C)')

print -depsc OR1-810_CT07_dist_depth-temp
print -dpdf OR1-810_CT07_dist_depth-temp

```



Multi-axes plot-sal

```

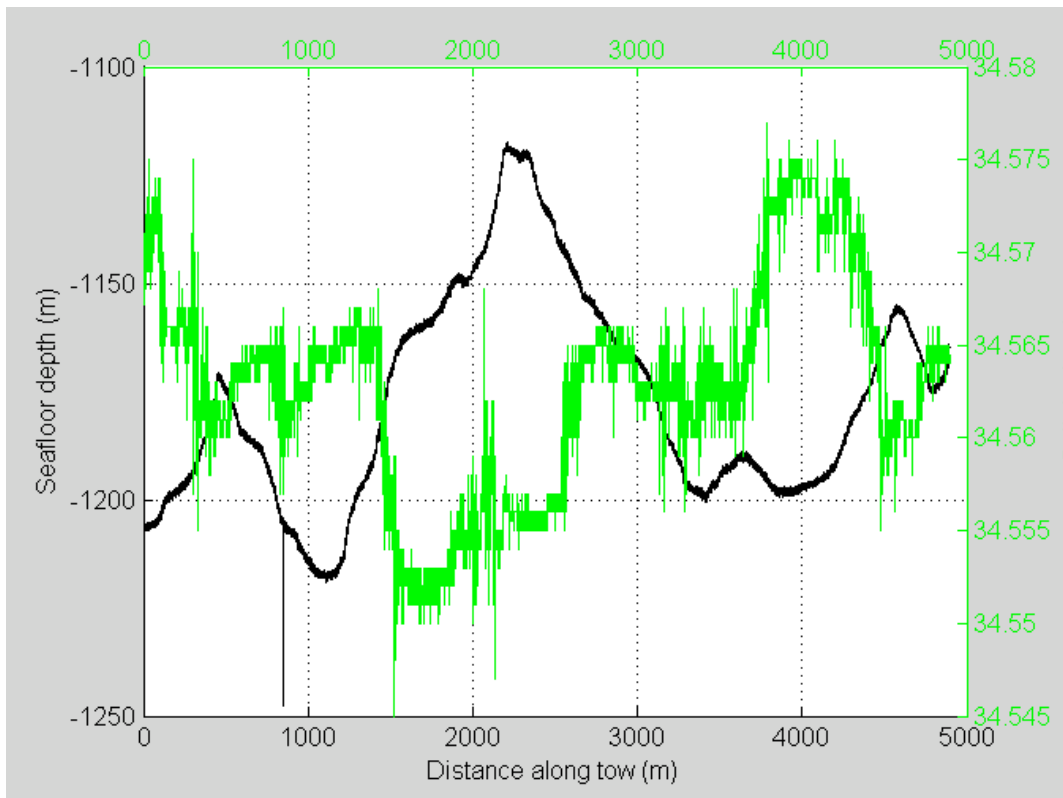
lowT = min(CTDsalC);
highT = max(CTDsalC);

figure
x1 = CTDDist; y1 = CTWdepC;
x2 = CTDDist; y2 = CTDsalC;
h11 = line(x1,y1,'Color','k');
xlabel('Distance along tow (m)')
ylabel('Seafloor depth (m)')
ax1 = gca;
set(ax1,'XColor','k','YColor','k');
grid on
ax2 = axes('Position',get(ax1,'Position'),...
          'XAxisLocation','top',...
          'YAxisLocation','right',...
          'Color','none',...
          'XColor','g','YColor','g');
h12 = line(x2,y2,'Color','g','Parent',ax2);

ylabel('Salinity (psu)')

print -depsc OR1-810_CT07_dist_depth-sal
print -dpdf OR1-810_CT07_dist_depth-sal

```

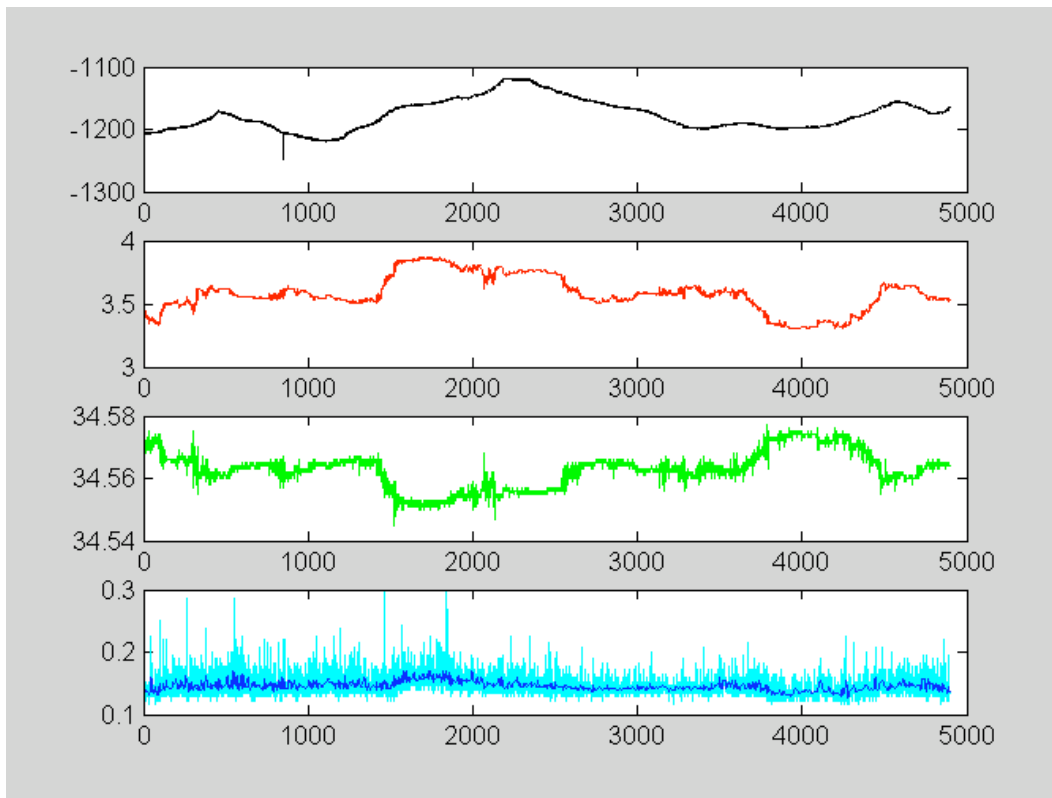
Depth, temperature, salinity, turbidity

```

figure
subplot(4,1,1)
plot(CTDdist,CTDwdepC,'k')
subplot(4,1,2)
plot(CTDdist,CTDtempC,'r')
subplot(4,1,3)
plot(CTDdist,CTDsalC,'g')
subplot(4,1,4)
plot(CTDdist,CTDturbC,'c')
CTDturbF = lowfilt(CTDturbC);
hold on
plot(CTDdist,CTDturbF,'b')
ylim([0.1 0.3])

print -depsc OR1-810_CT07_depth-temp-sal-turb
print -dpdf OR1-810_CT07_depth-temp-sal-turb

```



Making a sample file

```

outfile1 = 'OR1-810_CT07_samples.txt';

fid=fopen(outfile1, 'w');
fprintf(fid, '%s\n', 'SAMPLE,DATE,TIME,LAT, LON, UTMX, UTMY, WDEP, ID');

for bb = 1:length(SAMtime)
    sampN = ['OR1-810_CT07_0' num2str(BTLnum(bb))];
    gg = SAMtime{bb};
    sampT = [gg(1:2) ' ' gg(4:5) ' ' gg(7:8)];
    fprintf(fid, '%s,%s,%s,%9.8f,%10.7f,%8.2f,%8.2f,%6.2f \n', ...
        sampN, SAMdate{bb}, sampT, SAMlat{bb}, SAMlon{bb}, SAMx{bb}, ...
        SAMy{bb}, SAMwdep{bb});
end

```

Making new flash file

```

rec = zeros(1,length(FLAsodC));

for bb = 1:length(FLAsodC)
    rec(bb) = find(CTDsodC == FLAsodC(bb));
end

FLAxlb = LBxi(rec);
FLAYlb = LByi(rec);
FLAlonlb = LBloni(rec);
FLAlatlb = LBlati(rec);
FLAshlat = SNAVlati(rec);
FLAshlon = SNAVloni(rec);

display('Making output file...')
outfile = 'OR1-810_CT07_arc.txt';

fid=fopen(outfile, 'w');
fprintf(fid, '%s\n', 'DATE,TIME,LAT, LON, UTMx, UTMY, SHIPLAT, SHIPLON, DEPTH, ALT, WDEP, TEMP, TURB, SAL, HLINK');
for i = 1:length(FLAtimeC)
    temp = FLAtimeC{i};
    % image_name=[temp(1:2) '_' temp(4:5) '_' temp(7:8)];
    temp = PIXtimeC{i};
    image_name=temp;
    temp3 = FLAtimeC{i};
    image_time=temp3;
    temp2 = FLAdateC{i};
    image_date=[temp2(8:11) '_08_' temp2(1:2)];
end

```

```
path=['path:\' image_name];
fprintf(fid, '%s, %s, %9.8f, %10.7f, %8.2f, %8.2f, %9.8f, %10.7f, %6.2f, %4.2f, %6.2f, %7.4f, %7.4f, %7.4f, %s\n', ...
        FLAdateC(i), image_time, FLAlatlb(i), FLAlonlb(i), FLAxl(b(i), FLAylb(i), FLAshlat(i), FLAshlon(i), FLAdepC(i), FLAaltC(i), ..
        FLAwdpC(i), FLAtempC(i), FLAturbC(i), FLAsalC(i), path);
end
```

Making output file...

Saving stuff

```
eval('save OR1-810_CT07.mat');
fclose('all');
display('Jia shun!!')
```

Jia shun!!

Published with MATLAB® 7.2