# An Overview of Zhongxiang Zhou's Toolbox as modified by B. Bloss for TTIDE, r1.0.0

August 11, 2015

# Processing Script Overview

## Update History

0.9 Initial Document
1.0.0 Initial release for use processing TTIDE moorings.

## 1 Setup

### Functional Summary

Before invoking the toolbox functions to work through a mooring, the toolbox conventional workflow uses a few top level namespace variables:

- **savemmp** - Usually: [1] A flag to save data to output files
- **label** - Usually: ['Interp'] Do you want to interpolate gaps in data?
- **intwhere** - Usually: ['top-bottom-middle'] Where do we want to fill in gaps?
- **UID** - Similar to: ['TestRun-TTIDE-T7-' label] A descriptive ID that is used internally and in some filenames by the toolbox

### Called by

Nothing.

### Inside The Function

Nothing

# 2 Loading In Instruments

## Functional Summary

Next, processed instrument files are loaded in. Each instrument is loaded into a uniquely named structure. The actual variable names are used by the toolbox to determine what 'kind' of instrument is being processed. Notes below. Additionally, some fields within the structure may need renamed to feed the toolbox.

## Called by

Nothing

## Inside The Function

As noted above, the structures' actual variable names are inspected by the toolbox to determine what an instrument is (and thus the right routine to feed it to). The structure names simply need to start with these. The existing instrument types are: ADCP, MMP, SBE, and TLOG. Often loading in an instrument may result in a structure that is an extra layer deep. This is fixed in the example below. Additionally, the toolbox works with yearday fields instead of Matlab datenums. This is also shown below:

```
%An example for loading in an ADCP
ADCP_TOP=load('SN8122_TTIDE_AVE_10min.mat');
ADCP_TOP=ADCP_TOP.Vel;
ADCP_TOP.yday(find(isnan(ADCP_TOP.yday)))=datenum2yday(...
     ADCP_TOP.dtnum(find(isnan(ADCP_TOP.yday))));
```

# 3 Populate Mooring Info (Forward Parameters)

## Functional Summary

Here some of the overall parameters common to the entire mooring are set (location, depth, names) in the `FP` structure.

## Called by

Nothing

## Inside The Function

Specific variables for project name, project year (important as yearday is used), station number (SN), depth, and location are set up:

COMMENT HERE ABOUT BOTDEPTH

```
clear FP %in case it exists
FP.Project = 'T7_TestRUN-TTIDE';
FP.year=2015;
FP.SN='T7'; %Station Number
FP.depth=nanmean(ADCP_BOT.botdepth);
FP.lon = nanmean(ADCP_BOT.lon);
FP.lat = nanmean(ADCP_BOT.lat);
```

# 4 mkMooring

## Functional Summary

This generates the mooring (Raw, non-gridded) structure. It takes in the FP structure created above and incorporates that information into the mooring structure itself (Project, UID, Station Name, lat, lon, year, depth). It then sets up the instrument list.

## Called by

Top level mooring script

## Inside The Function

This prepares a the mooring's instrument list and data list. The InstrumentList allows types 'MP', 'SBE', 'ADCP' and the InstrumentNumber simply keeps tally of the number of each. The data list is what will actually track the corresponding instruments' data.
To add processing for new kinds of instruments, the shorthand name for the instrument will need added to InstrumentList.
NOTE: After creating the mooring using this function, you will want to set the Mooring.Figure_dir and Mooring.Data_dir variables. I use the present directory ('cd') for portability and example, but one could hardcode paths.

# 5 AddInstrumentToMooring

## Functional Summary

This function simply acts to determine what kind of instrument is being added, and set it up within the mooring structure appropriately. No actual processing occurs here.

## Called by

Top level mooring script, repeatedly, once for each instrument to be added.

### Inside The Function

The instrument type is determined by looking for the shorthand name (eg. 'MP' or 'SBE' etc) within the actual variable name the user assigned the instrument. Checks are provided to ensure that the instrument is a valid type and does not belong to more than one type. Then the provided instrument structure is linked to the mooring structure and the InstrumentNumber and DataNumber fields are updated. (Note: The data itself is NOT yet copied in).

## 6 PlotMooring

### Functional Summary

Conventionally, if savemmp (above) is set to 1, PlotMooring is run instead of ShowMooring (below). Simply saves a PDF of the plots produced in ShowMooring.

### Called by

Top level mooring script if savemmp==1.

### Inside The Function

This simply saves a pdf of the ShowMooring plot

## 7 ShowMooring

### Functional Summary

This is run directly without saving the figures if savemmp is set to zero. This produces initial plots for the un-gridded, raw mooring data.

### Called by

Top level mooring script if savemmp==0. Also called nested within PlotMooring.

### Inside The Function

This function will plot all the data it has for each of the default variables (which can be edited within this function). They are u,v,t, and s. The data are plotted and the common time is determined and boxed, highlighting when the intstruments overlap, appropriate for gridding. Note however, that this overlap is not saved as a suggestion yet, it is merely displayed. The toolbox will recalculate this during later processing and make a recommendation based on that later time overlap calculation.

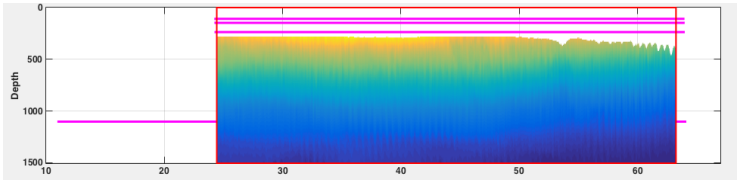


Figure 1: ShowMooring Output for one variable (of several). Note the pink lines showing single depth instruments and red box bounding the automatically identified common (overlapping) time frame.

# 8 saveDATA

## Functional Summary

This function saves the input argument (structure) to a .mat file with the name of the argument.

## Called by

Top level mooring script (conventionally if savemmp is set to one).

## Inside The Function

This is fairly straight forward. it saves the data passed in as the single argument under the literal name of the argument passed in. If Data_dir or Data_name do not exist, the present working directory and DATA.UID are used respectively.

# 9 CheckMooring

## Functional Summary

This function provides a documented (text file) evaluation of the instruments comprising the mooring.

## Called by

Top level mooring script.

## Inside The Function

This function creates or overwrites a text file with the Mooring.UID name. In that file it journals an instrument by instrument evaluation as it determines the dt and dz for each. It additionally finds the common yearday range. After evaluating all the instruments on the mooring, it then suggests and sets a: Mooring.dt, Mooring.yday_grid, Mooring.dz, and Mooring.z_grid .

NOTE: after this function has run, you may choose to override these values! Simply edit them afterward. That allows custom gridding to be set.

# 10 Set Custom dt, yday_grid, dz, and z_grid

## What to Do Here

If you wish to set custom gridding, overwrite these variables here. They may be set in the structure FP. If they do not exist in FP, then the equivalent fields in the mooring structure will then be used.
NOTE: The variables below (in mkMMP) use different names depending if they were from FP or the Mooring structure.

- Mooring.z_grid ⇒ FP.z
- Mooring.yday_grid ⇒ FP.yday

Notice that the Mooring.dz and Mooring.dt values aren't actually used here! It appears that they were only used in constructing the above grids.

# 11 mkMMP

## Functional Summary

This creates a basic MMP structure. The name is a relic. This will be the structure which holds the gridded, data.

## Called by

Top level mooring script.

## Inside The Function

This creates the shell of the gridded mooring structure, into which the data will be put later. The shell includes: Project, SN, UID,lon, lat, year, depth, all copied from the Mooring structure if the exist (set to NaN if they DNE). The z grid (MMP.z) is realigned if necessary (so it becomes a single column). Additionally NaN populated 2-D matrices are created for each variable hardcoded in this function (u,v,t,s,sgth). Additionally, MMP.dz and MMP.dyday are created from median and mean of the respective grids, NOT from the variables assigned in the Mooring.

Finally, each instrument's data is added to the MMP structure, as this function iteratively calls AddDATAToMMP (see next).

# 12 AddDATAToMMP

## Functional Summary

This function goes through a single instrument (DATA source) and calls the apropriate gridding function then incorporates the results into the overall MMP gridded structure.

## Called by

mkMMP.m

## Inside The Function

This function has allowances to add new variables to the MMP structure if they weren't in the list above (in mkMMP) but are found in an instrument's variable list (VarNameList). Such variables are created as above as 2-D NaN filled matrices.

Then, one by one, each variable is gridded. The toolbox contains two functions for gridding data, one for ADCPs, and the other for (MMPs and all other instruments). The toolbox may benefit from writing additional routines for other instruments. The determination of which instrument type is being worked with, is made again by the actual variable name assigned by the user at the top of the mooring script.

Then, the data which results from the gridding is copied into the MMP gridded variables. However, NOTE: the data is only copied in if none exists already! That is, the toolbox makes no judgements on which data is superior if there is a conflict, instead simply the first data gridded, is put in the overall grid and is not overwritten.

# 13 GriddingMMP

## Functional Summary

This function grids a single variable from an instrument. It first does time (the second matrix dimension, across), then the depth gridding (the first matrix dimension, down).

## Called by

AddDataToMMP.m

## Inside The Function

Simple linear interpolation is used to fit the raw data first onto the yearday grid then, column by column, onto the depth grid. If the gridded data is too

far away from the raw data, it is set to NaN, ensuring the interpolation doesn't go too far. Finally, if requested, columns and rows containing all NaNs can be trimmed off. In conventional usage the trim capability isn't used (not used ⇒ FP.trim==0).

# 14 GriddingADCP

## Functional Summary

This is a simpler version of the GriddingMMP function which deals specifically with ADCPs.

## Called by

AddDataToMMP.m

## Inside The Function

This operates similarly to the GriddingMMP function, however it focuses only on ADCP processing. It lacks a trim, and check if data is too far away. Instead a different means of handling the ADCP gridding is used, and the interpolation is done only over the valid region of time and depth covered by the ADCP.

# 15 GriddingAA

## Functional Summary

This is a simpler version of the GriddingMMP function which deals specifically with ADCPs.

## Called by

AddDataToMMP.m

## Inside The Function

This operates similarly to the GriddingMMP function, however it focuses only on Aanderaa processing. It assumes the AAxxx.z parameter contains a time-series of depth values representing the actual depths of the instrument. This can be created using a perturbation script as shown in the TTIDE T7 example.

# 16 GriddingSBE_Knockdown

## Functional Summary

This is a simpler version of the GriddingMMP function which deals specifically with SBEs that have been knocked down.

## Called by

AddDataToMMP.m

## Inside The Function

This operates similarly to the GriddingMMP function, however it focuses only on SBE processing. It assumes the SBExxx.z parameter contains a time-series of depth values representing the actual depths of the instrument. This can be created using a perturbation script as shown in the TTIDE T7 example.

# 17 PlotMMP

## Functional Summary

This is similar to the ShowMooring function above, plotting all the variables we choose to show.

## Called by

Top level mooring script.

## Inside The Function

Before executing this function, set up: MMP.VarNamesToShow with those variable names to be plotted as well as the MMP.Figure_dir and MMP.Data_Dir. Finally, you may specify an MMP.Figure_name .

# 18 AddDisplacementToMMP_iso

## Functional Summary

This function takes t or sgth (sgth is the default) and returns the vertical displacements of parcels.

## Called by

Top level mooring script.

### Inside The Function

This function requires two forward parameters to be set before running: FP.time='yday'; is common to specify the yearday grid variable, and FP.varname='sgth' or FP.varname='t'.

This function creates a list of isopycnals or isotherms and then interpolates to determine the depth of each iso-line, column by column. Then, smoothed iso-lines are calculated, either using a specified window (units: days), or the mean of the entire line (if FP.window is NaN). Eta is then calculated as the negative of the difference of the original lines and the smoothed lines. Finally, these resulting displacements are projected onto the grid. The function in its current state goes further and actually interpolates eta at the top, bottom, and middle. Andy noted that a good window is over 4x M2 tidal cycles (12.42/24*4)
NOTE TO SELF: It appears this was left in as an error? It also changes FP.where out of the user's control. REVISIT and REVISE?

## 19 ReplaceNanInMMP

### Functional Summary

This function replaces NaNs in the variables that are passed in. The user can select to fill in NaNs in the top, middle, and/or bottom as perscribed by FP.where.

### Called by

AddDisplacementToMMP, Top Level Mooring Script

### Inside The Function

String comparisons are done on a lowercase converted version of FP.where to determine where to replace NaNs. For NaNs in the middle, linear interpolation is used. For NaNs in the bottom, the method depends on the variable name. For: n2,u, and v, the bottom NaNs are replaced with a mean value of the last few good values (as determined by FP.BottomMean, or its default). If the variables are: eta or t, a polyfit is used. For NaNs at the top in: n2, eta, a linear interpolation between zero at the top of the grid and the first good value is used. For: u,v, the first good value is repeated.

## 20 AddN2ToMMP

### Functional Summary

This calculate the N2 profiles from salinity and temperature.

### Called by

Top level mooring script.

### Inside The Function

Three parameters are passed in. FP.window determines the smoothing window (or a mean is used if FP.window is longer than the record.) FP.vsmooth is the number of vertical points to use for smoothing (defaults to 3). Finally FP.bound is a flag that determines how many boundary layers to remove. This function calculates N2 then using the saltwater toolbox (the version called for is included in the toolbox zip file).

## 21 MakeCTD_fromMMP

### Functional Summary

This takes the gridded mooring and creates a CTD structure that is compatible with Matthew Alfords CTD class of functions.

### Called by

Top level script.

### Inside The Function

This is largely for creating a compatible structure. If FP.depth is provided, the depth gridding runs to that depth, otherwise MMP.depth is used and a warning is printed. Additionally N2 is interpolated onto this new CTD grid.

## 22 ...A few small utility functions remain

### AddBCVelToCTD, AddEnergyToIW, AddFluxToIW, ezFlux, etc

There is an ensemble of energy and flux functions that are available to run on MMP structures once created. These are beyond the scope of the gridding project, but are available for use during processing. They are designed for, and work best with, McLane moorings. Sparse moorings, like the TTIDE T-moorings do not work well with them. They are fairly straight forward to use. See the IWISE examples if you seek to use these.

## 23 Final Notes

- Watch FP! It is easy to forget values which have previously been set for other functions which might affect later functions. Consider trying to directly set all FP values before each function call or clearing variables in FP when they are not needed.