# Echotools

***Release 0.2***

**Feb 15, 2022**

# CONTENTS:

CHAPTER

ONE

# INTRODUCTION

This document contains information on automatic post processing of near real time echometric sampling from a glider. This toolset is required since the pseudogram is embedded in the echometrics messages sent by the glider. The *dbd2asc* binary cannot be directly used as it truncates the least significant bits used to encode the pseudogram. The *dbd2asc* is used to obtain time and depth information to depth correct the pseudogram plot.

CHAPTER

TWO

# DOCUMENTATION

## 2.1 Near Real Time Post Processing of Pseudograms

The pseudograms are read from the near real time information sent over iridium. This information is embedded into the echometrics information provided by the Teledyne Webb TBD files. Sometimes the companion SBD file is required for additional time and depth information. A cache file is typically needed to decode the binary structure of the tbd and sbd file using this toolset or `dbd2asc`.

### 2.1.1 Output

The python program `decodePseudogram.py` can decode the tbd file and produce two types of output:

- CSV (optionally with header)
- image (png, jpg, pdf)

The information can be saved to a file or sent to standard output (stdout).

#### Metadata

When saving the CSV to a file or stdout, three columns of information are provided.

- Column 1: Time; seconds since 01-01-1970; timezone GMT/UTC
- Column 2: Depth; meters
- Column 3: Sv (dB)

NOTE: Unique keys are formed from the first two columns giving this dataset a 3D like structure.

### 2.1.2 Program

For more information on the operation details of this program and library, please refer to *Programs and Modules*.

### Syntax

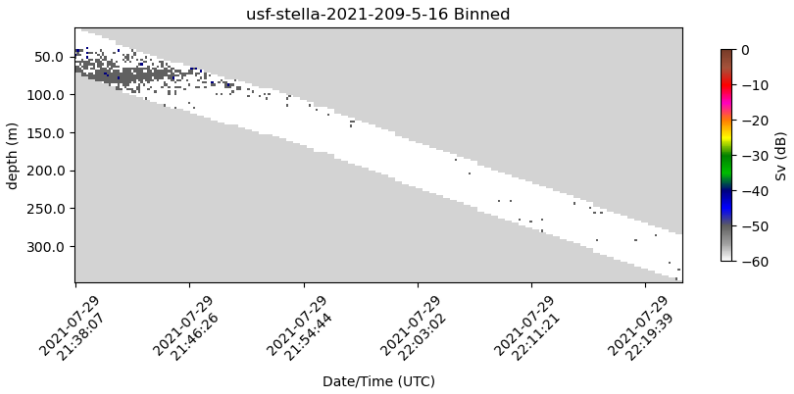
```
decodePseudogram.py:

usage: decodePseudogram.py [-h] [--tbdFile TBDFILE] [--sbdFile SBDFILE]
                           [--cacheDir CACHEDIR]
                           [--dbd2asc DBD2ASC]
                           [--csvOut CSVOUT] [--csvHeader]
                           [--imageOut IMAGEOUT] [--debug]
                           [--echosounderRange ECHOSOUNDERRANGE] [--useScatterPlot]
                           [--binnedDepthLabels] [--title TITLE]

Pseudogram decoder: This reads a Teledyne Web glider tbd file that has encoded
pseudograms. In some cases, additional metadata is required from the sbd
glider file. To generate output, --csvOut and/or --imageOut must specify full
or relative path with a filename or stdout. The arguments --csvOut and --imageOut
cannot both be stdout. The image format is controlled by file extention
(png, jpg, pdf, gif). The default echosounderRange is -60.0 meters for an echosounder
instrument points towards the surface. If the instrument is pointing down, use a
positive value (range) in meters.

optional arguments:
  -h, --help            show this help message and exit
  --tbdFile TBDFILE     full or relative path with filename to glider tbd
                        binary input file
  --sbdFile SBDFILE     full or relative path with filename to glider sbd
                        binary input file
  --cacheDir CACHEDIR   Directory with glider cache files; default current directory
  --dbd2asc DBD2ASC     full or relative path with filename to glider dbd2asc binary
  --csvOut CSVOUT       full or realative path with filename to write CSV output or
                        'stdout';
                        default None
  --csvHeader           (flag) include header with CSV output; default False
  --imageOut IMAGEOUT   filename to write image or stdout; default None
  --debug               (flag) show extra debugging for this python script;
                        default False
  --echosounderRange ECHOSOUNDERRANGE
                        Echosounder range; default -60.0 (meters) instrument
                        facing up; positive values instrument facing down
  --useScatterPlot      (flag) Plot using a scatterplot instead of a time/depth
                        binned (raster image) plot; default False
  --binnedDepthLabels   (flag) Use the original depth bin labels instead of the
                        adjusted depth labels for the time/depth binned plot;
                        default False
  --title TITLE         optional figure title; default None
```

### Examples

Produce time/depth binned (raster image) plot of University of South Florida glider with instrument facing downwards with a range of 60.0 meters:

```
./decodePseudogram.py --cacheDir examples/cache --dbd2asc bin/dbd2asc \
    --echosounderRange 60.0 --tbdFile examples/data/usf-stella-2021-209-5-16.tbd \
    --imageOut examples/output/usf-stella-2021-209-5-16_binned.png \
    --title "usf-stella-2021-209-5-16 Binned"
```

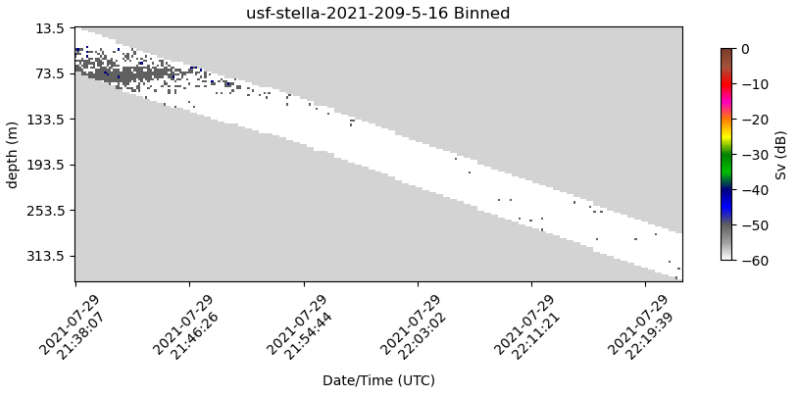
The resulting figure:




Produce time/depth binned (raster image) plot of University of South Florida glider with instrument facing downwards with a range of 60.0 meters (y-axis depth intervals centered on depth bins):

```
./decodePseudogram.py --cacheDir examples/cache --dbd2asc bin/dbd2asc \
    --echosounderRange 60.0 --tbdFile examples/data/usf-stella-2021-209-5-16.tbd \
    --imageOut examples/output/usf-stella-2021-209-5-16_binned_yaxis.png \
    --title "usf-stella-2021-209-5-16 Binned" --binnedDepthLabels
```

The resulting figure:




Produce scatter plot of University of South Florida glider with instrument facing downwards with a range of 60.0 meters:

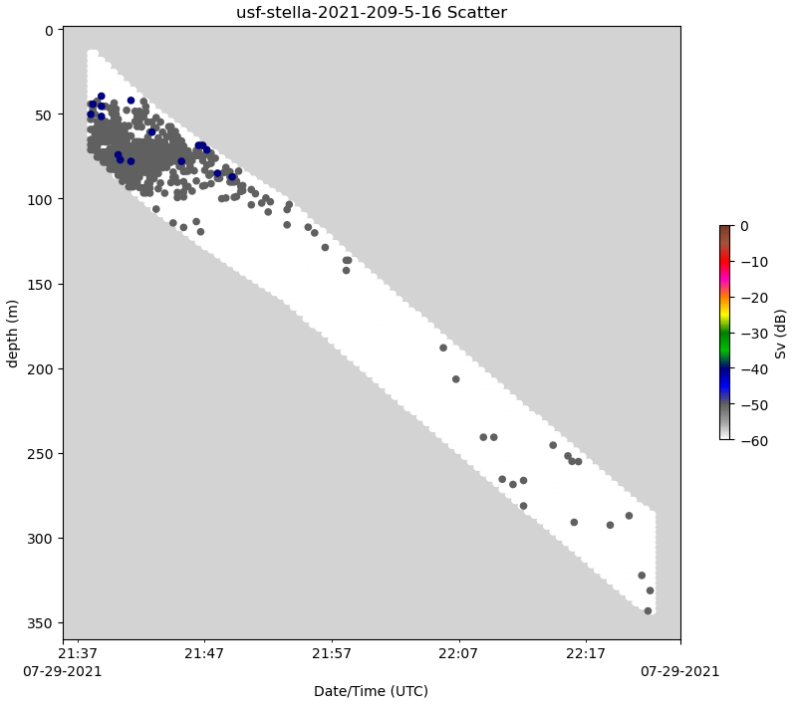
```
./decodePseudogram.py --cacheDir examples/cache --dbd2asc bin/dbd2asc \
    --echosounderRange 60.0 --tbdFile examples/data/usf-stella-2021-209-5-16.tbd \
```

(continues on next page)

(continued from previous page)

```
    --imageOut examples/output/usf-stella-2021-209-5-16_scatter.png \
    --title "usf-stella-2021-209-5-16 Scatter" --useScatterPlot
```

The resulting figure:




Produce time/depth binned (raster image) plot of University of Alaska Fairbanks glider with instrument facing upwards with a range of 60.0 meters (echosounderRange=-60.0) and output to CSV:

```
./decodePseudogram.py --cacheDir examples/cache --dbd2asc bin/dbd2asc \
    --echosounderRange -60.0 --tbdFile examples/data/uaf-gretel-2022-019-8-0.tbd \
    --imageOut examples/output/uaf-gretel-2022-019-8-0_binned.png \
    --title "uaf-gretel-2022-019-8-0 Binned" \
    --csvOut examples/output/uaf-gretel-2022-019-8-0.csv
```

The resulting output should look very similar to the standard output shown below.

Same example, but sending the CSV to stdout:

```
./decodePseudogram.py --cacheDir examples/cache --dbd2asc bin/dbd2asc \
    --echosounderRange -60.0 --tbdFile examples/data/uaf-gretel-2022-019-8-0.tbd \
    --csvOut stdout
```

Output printed to standard output:

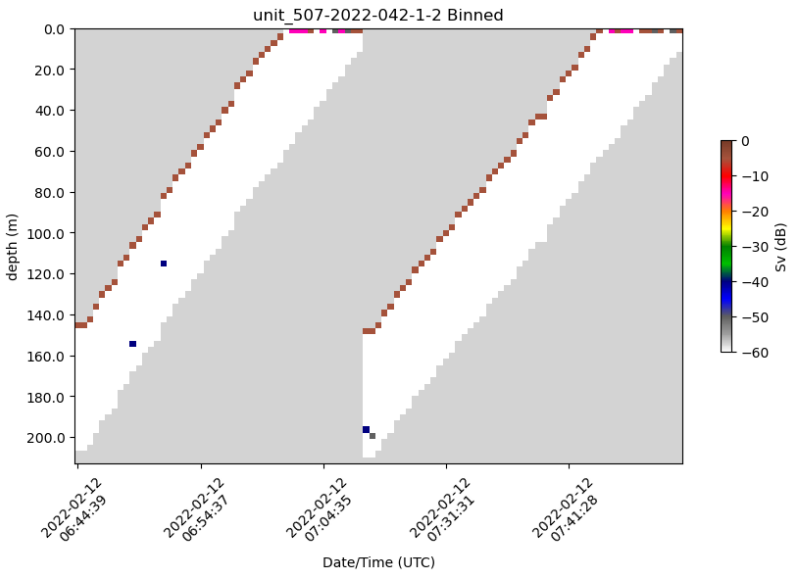
```
1642721076.190063, 13.320000, -60.000000
1642721076.190063, 10.320000, -60.000000
1642721076.190063, 7.320000, -60.000000
....
```

Information sent to standard output can be captured or piped into another process for upstream processing or storage by another program.

Examples that utilize tbd and sbd files:

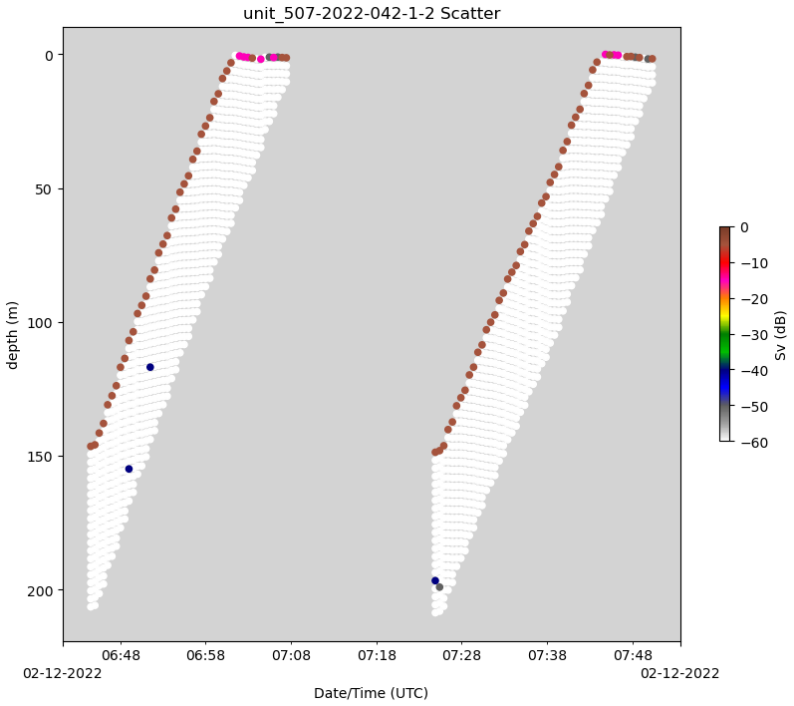
```
./decodePseudogram.py --cacheDir examples/cache --dbd2asc bin/dbd2asc \
    --echosounderRange -60.0 \
    --tbdFile examples/data/unit_507-2022-042-1-2.tbd \
    --sbdFile examples/data/unit_507-2022-042-1-2.sbd \
    --imageOut examples/output/unit_507-2022-042-1-2_binned.png \
    --title "unit_507-2022-042-1-2 Binned"
```

The resulting figure:




```
./decodePseudogram.py --cacheDir examples/cache --dbd2asc bin/dbd2asc \
    --echosounderRange -60.0 \
    --tbdFile examples/data/unit_507-2022-042-1-2.tbd \
    --sbdFile examples/data/unit_507-2022-042-1-2.sbd \
    --imageOut examples/output/unit_507-2022-042-1-2_scatter.png \
    --title "unit_507-2022-042-1-2 Scatter" --useScatterPlot
```

The resulting figure:




### Requirements

This python script requires:

- python3
- decodePsudogram.py
- teledyne.py (python class for handling teledyne glider files)
- Teledyne Webb linux binary (dbd2asc)
  - The ability to execute these binaries
- Input files: tbd and or sbd files and associated cache (cac) files

## 2.2 Teledyne Webb file types

### 2.2.1 Flight data extensions

- dbd - full resolution data
- sbd - shortened flight data that is sent back via iridium
- mlg - message log files from the flight computer

### 2.2.2 Science data extensions

- ebd - full resolution data
- tbd - shortened science data that is sent back via iridium
- nlg - message log files from the science computer. We are running the echodroid in verbose more so all of the back and forth between the Science persistor, the Odroid and the mini are logged in this file.

#### Cache files

Cache files (cac) files help describe the binary structure of the tbd and sbd data files.

## 2.3 Auxillary programs

### 2.3.1 Catalog

#### processTbd.py

This program can process one or more `tbd` files in a given directory. The program will write out corresponding `dat` files which are the decoded versions of the `tbd` file. A error will result if the corresponding cache file is not found.

```
./processTbd.py --binDir bin --dataDir examples/data \
  --cacheDir examples/cache --rename --file *.tbd
```

#### catalogTbd.py

This program scans the decoded `dat` files and creates a catalog file. The catalog file contains a summary of files with start and end date and time to quickly identify which `tbd` files are needed for post processing.

```
./catalogTbd.py --dataDir examples/data --file *.dat \
  --catalog examples/data/catalog.dat
```

### 2.3.2 Originally developed programs

For the original toolset provided by Alex, see the `alex/original` directory. Within that directory run the `echoGen.sh` script with appropriate symbolic links to data and cache files.

NOTES:

- The Teledyne Webb binary `dba_sensor_filter` is required for this toolset.
- A symbolic link from `examples` needs to point to the directory with `data` and `cache` directory.
- A symbolic link from `bin` needs to point to the directory with the Teledyne Webb binaries.

Here is an example call to these scripts from the main shell program:

```
. echoGen.sh uaf-gretel-2022-019-6-0.tbd
```

The shell script performs the following steps:

- Decode (tmpSSV.txt)

- Reorder (tmpEcho.txt)
- Extract time/depth from tbd (tmpBar.txt)
- Create images from decoded temporary files
  - Raw image
  - Depth (bin) corrected image

This toolset of scripts formed the basis for creating the `decodePseudogram.py` script and `teledyne.py` python class/library.

## 2.4 Teledyne linux binaries

The zip file is stored within the repository. Extract only the binaries needed to perform post processing after downloading the repository.

Source of these binaries require registration with Teledyne Marine Webb Reserches forum:

https://datahost.webbresearch.com/

## 2.5 Example data

### 2.5.1 USF: Stella

#### July Pt Sur cruise

- Mode: echo
- Instrument is pointed down (+60.0 meters)

| tbd | cache | description |
|---|---|---|
| usf-stella-2021-209-5-16.tbd | d0f88888.cac | Day 5 Segment 16 |

### 2.5.2 UAF: Gretel

#### Seward, AK sea trial: January 19-20, 2022

- Mode: combo
- Instrument is pointed up (-60.0 meters)

| tbd | cache | description |
|---|---|---|
| uaf-gretel-2022-019-0-0.tbd | 6ce3fe1f.cac | Dive 1 |
| uaf-gretel-2022-019-6-0.tbd | 6ce3fe1f.cac | Dive 2 |
| uaf-gretel-2022-019-8-0.tbd | 6ce3fe1f.cac | Dive 6 |
| uaf-gretel-2022-020-0-0.tbd | 6ce3fe1f.cac | Dive 8 |

#### Echodroid deployment: February 12-, 2022

- Mode: combo
- Instrument is pointed up (-60.0 meters)

| files | cache | description |
|---|---|---|
| unit_507-2022-042-1-2.tbd | 6ce3fe1f.cac | Day 1 Segment 2 |
| unit_507-2022-042-1-2.sbd | 28b7e1b2.cac | Day 1 Segment 2 |

## 2.6 Teledyne files

### 2.6.1 File extensions

Flight data:

- dbd: full resolution data
- sbd: shortened flight data that is sent back via iridium
- mlg: message log files from flight computer

Science data:

- ebd: full resolution data
- tbd: shortened science data that is sent back via iriduim
- nlg: message log files from the science computer

Processed data:

- dat: decoded data using `dbd2asc`

## 2.7 Programs and Modules

### 2.7.1 Programs

#### decodePseudogram.py

This script contains functions that carries out the process of obtaining the pseudogram data array in either CSV (ASCII, spreadsheet) or image(PNG) form. The output can be directed to a file or standard output (stdout).

Data written as output is in three columns (comma separated values): Timestamp, Depth, Density. A metadata description is given below.

Image output is controlled by the filename extension provided. A typical format is PNG. Use ".png" in the filename to produce a PNG formatted image.

### Metadata

Here is the metadata description for each of the coloums of this dataset (ASCII, spreadsheet):

| Column | Description |
|---|---|
| Timestamp | seconds since 01-01-1970 epoch; timezone GMT/UTC |
| Depth | depth (meters) |
| Density | Sv (dB) |

### Command line arguments

```
-h, --help            show this help message and exit
--tbdFile TBDFILE     full or relative path with filename
                      to Teledyne glider tbd binary input file
--sbdFile SBDFILE     full or relative path with filename
                      to Teledyne glider sbd binary input file
--cacheDir CACHEDIR   Directory with glider cache files;
                      default current directory
--dbd2asc DBD2ASC     full or relative path with filename
                      to glider dbd2asc binary
--csvOut CSVOUT       full or realative path with filename
                      to write CSV output or 'stdout'; default None
--csvHeader           (flag) include header with CSV output; default False
--imageOut IMAGEOUT   filename to write image or stdout; default None
--debug               (flag) show extra debugging for this
                      python script; default False
--echosounderRange ECHOSOUNDERRANGE
                      Echosounder range; default -60.0 (meters)
                      instrument facing up; positive values
                      instrument facing down
--title               optional figure title; default None
--useScatterPlot      (flag) Plot using a scatterplot
                      instead of a time/depth binned
                      (raster image) plot; default False
--binnedDepthLabels   (flag) Use the original depth bin
                      labels instead of the adjusted depth
                      labels for the time/depth binned plot;
                      default False
```

decodePseudogram.**printAttributes**(*myObj*)

This is a convienence function for printing object attribute information for the supplied object. This function writes to standard output and does not print any objects with an underscore prefix. This was mainly used for debugging during development.

**Parameters** **myObj** (any, required) –

**Return type** Prints information to standard output.

decodePseudogram.**showHelp**(*parser*)

This prints the program description and arguments to standard ouptut and exits.

#### catalogTbd.py

catalogTbd.**catalogFiles**(*dataDir*, *inputFile*, *catFile*, *showWarning*)

#### processTbd.py

processTbd.**processTbdFile**(*tbdFile*, *cacheDir*, *binDir*, *dataDir*, *renameFlag*)

### 2.7.2 Modules

#### teledyne

**class** teledyne.**Glider**(*tbdFile=None*, *sbdFile=None*, *cacheDir=None*, *dbd2asc=None*, *debugFlag=False*)

Bases: object

A container class for handling Teledyne Webb glider data.

Initialize a glider object.

**Parameters**

- **tbdFile** (str) – Full or relative path with filename to glider tbd file.
- **sbdFile** (str) – Full or relative path with filename to glider sbd file.
- **cacheDir** (str) – Full or relative path to directory with glider (sensor) cache files.
- **dbd2asc** (str) – Full or realtive path with filename to Teledyne Webb binary dbd2asc. NOTE: System must be able to execute the binary.
- **debugFlag** (bool) – Flag for extra debugging information printed to standard output. Default: False

**createPseudogramSpreadsheet**(*args*)

This function reads GLIDER.data['pseudogram'] and places it in a spreadsheet format in GLIDER.data['spreadsheet']. The function is expecting at least two fields to have been read from the provided tbd file.

**Parameters** **args** (argparse) – Parsed command line arguments.

##### Notes

- args.debugFlag: Boolean flag. If True, additional output is printed to standard output.
- args.useScatterPlot: Boolean flag. If True, a scatter plot is produced instead of the depth/time binned plot.

**extractColumns**(*source*, *columns=[]*, *ignoreNaNColumns=[]*, *asDict=False*)

This function extracts requested columns from the GLIDER.data[source] object. This function will also remove rows for specified columns that contain NaNs.

NOTE: This function replaces the need for dba_sensor_filter and subsequently ignoring output of NaNs.

**Parameters**

- **columns** (list) – Named columns to subset from GLIDER.data[source] object.
- **ignoreNaNColumns** (list) – One the data is collected by column, rows are eliminated in named columns where the values are NaN.

- **asDict** (bool) – This is a flag to change the return value as a python dict() object where the column names are the dictionary keys.

**Returns** This returns a subset of data stored in GLIDER.data[source]. This will either be another numpy array or a python dictionary.

**Return type** numpy array or dict()

**getDepthPixel**(*reqDepth*, *minDepth*, *maxDepth*, *depthBinSize*)

For the image (plotting) routine, depths are placed in discrete pixels by the depth bin size. The first pixel has the depth range of minimum depth to minimum depth plus depth bin size.

### Notes

For a depth bin size of 3.0 meters and a minimum depth of 2.0 meters. Bin zero (0) should be 2.0 to 5.0 meters, bin (1) will be from 5.0 meters to 8.0 meters and so forth.

**Parameters**

- **reqDepth** (float) – Requested depth (meters)
- **minDepth** (int) – Minimum depth bin
- **maxDepth** (int) – Maximum depth bin (not used)
- **depthBinSize** (float) – Actual depth size of each depth bin (meters)

**Returns** Returns the depth bin adjusted to the minimum depth bin

**Return type** int

**handleImage**(*args*)

This function handles writing out a graphical image. By default, the image rendering uses descrete pixels that have depth and time bins. If –useScatterPlot is set, a scatter plot is produced instead of a time/depth binned (raster image/imshow) plot. The raster plot coordinates are the depth and time bins which requires redefining x and y labels on the fly.

**Parameters** **args** (argparse) – Parsed command line arguments.

### Notes

- args.imageOut: May be a full or relative path with filename or *stdout*.
- args.debugFlag: Boolean flag. If True, additional output is printed to standard output.
- args.useScatterPlot: Boolean flag. If True, a scatter plot is produced instead of the depth/time binned plot.

**handleSpreadsheet**(*args*)

This function handles writing out the decoded tbd data in CSV format. Either a filename is provided or the spreadsheet is sent to 'stdout'. If the csvHeader flag is set to True, a header is also provided.

**Parameters** **args** (argparse) – Parsed command line arguments.

### Notes

- args.debugFlag: Boolean flag. If True, additional output is printed to standard output.
- args.csvOut: May be a full or relative path with filename or *stdout*.
- args.csvHeader: Boolean flag. If True, a header is included with CSV output to file or standard out.

**hatches_plot**(*ax*, *h*)
Create a default background for glider plots 'o'.

**nearest**(*array*, *val*)
Find the nearest value in a sorted numpy array and return the index for the nearest value.

**Parameters**

- **array** (numpy) – A value sorted numpy array that is to be searched.
- **val** (value) – The value to search for the closest matching element in the provided numpy array.

**Returns** The index of the closest matching element.

**Return type** int

**readPseudogram**()
This function reads the glider tbd file and extracts the embedded pseudogram from the echometrics data. The *dbd2asc* file cannot be used since it truncates the least significant bits in which the pseudogram is embedded.

NOTE: This function *readPsuedogram* should only be used for extracting encoded pseudogram information embedded in the echometrics data. All other glider files should be read using *dbd2asc*.

This function automatically determines if the glider was in "echo" or "combo" mode. Prior knowledge of the operational mode is not necessary.

**readSbd**()
This function reads a glider sbd file. This also reads the corresponding cache file for additional metadata.

**readTbd**()
This function reads a glider tbd file using the Teledyne Webb linux binary dbd2asc. This also reads the corresponding cache file for additional metadata.

**stopToDebug**()
Generic function to stop python in its debugger. When stopped by this function, it is necessary to go up one level in the execution stack to get to the exact location of the breakpoint. Use the *up* command to go up one level in the execution stack.

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

# INDEX