Data associated with Hughes et al. (in prep)

An ocean turbulence data reduction scheme for autonomous, vertically profiling floats

To be submitted to *J. Atmos. Oceanic Tech.*

During field tests in 2019, we deployed two FCS floats (units 4002 and 4003). Between the two, approximately 650 profiles were taken. For each profile, we are archiving four configurations of the data. Each one is a struct within the same Matlab `.mat` file. (An example of how to read in this file in Python is given at the end of this document.)

Struct name	Type of data
`data`	The raw data in voltage units
`cal`	The raw data after calibration (with shear despiked)
`avg`, `avgr`	The binned data including derived turbulence quantities

`cal`, `avg`, and `avgr` contain readmes with full details about the individual quantities that were measured or derived.

The structs `avg` and `avgr` are similar, but `avg` was derived with a standard processing approach, whereas `avgr` was derived using the data reduction scheme described in the paper. The number of depths bins may differ between the two due to differences in the calculation of W_{min} and hence the identification of the end of the profile. Further, `avgr` (but not `avg`) includes bad values of ϵ and χ in the lowest tens of meters of upward profiles when the ballast pump is running.[1]

Each file also includes the struct `head`, which is a header file that includes calibration coefficients.

Each filename is of the form 'DIR_YYYYMMDDhhmmss_00X.mat' where `DIR` is

[1]The data reduction scheme does not include a method to identify when the ballast pump is on within individual profiles. Therefore, we have left the bad values in 'avgr'. However, the bad values are easily identified in post-processing by comparing successive casts.

`UP` or `DN`, `YYYYMMDDhhmmss` is the time stamp when the raw file was first created, and `.00X` is `.001` for unit 4002 and `.002` for unit 4003.

Some files, especially early ones, may not contain any values in `avg` or `avg`. These can be ignored.

To open a given file in Python and load the data into dicts, use the following:

```
from numpy import squeeze
from hdf5storage import loadmat

f = loadmat(<filename.mat>)

def fields(struct):
    return struct.dtype.names

def read_array(quantity, struct):
    return squeeze(f[struct][quantity][0][0])

data = dict()
for quantity in fields(f['data']):
    data[quantity] = read_array(quantity, 'data')

cal = dict()
for quantity in fields(f['cal']):
    cal[quantity] = read_array(quantity, 'cal')

avg = dict()
for quantity in fields(f['avg']):
    avg[quantity] = read_array(quantity, 'avg')

avgr = dict()
for quantity in fields(f['avgr']):
    avgr[quantity] = read_array(quantity, 'avgr')
```