ADCPy

Sep 21, 2021

Contents:

CHAPTER 1

Purpose

This code prepares large amounts of single ping ADCP data from the raw binary for use with xarray by converting it to netCDF.

CHAPTER 2

Motivation

The code was written for the TRDI ADCP when I discovered theat TRDI's Velocity software could not easily export single ping data. While there are other packages out there, as the time of writing this code, I had yet to find one that saved the data in netCDF format (so it can be accessed with xarray and dask), could be run on linux, windows and mac, and did not load it into memory (the files I have are > 2GB)

The code is written as a module of functions, rather than classes, ensemble information is stored as nested dicts, in order to be more readable and to make the structure of the raw data (particularly the TRDI instruments) understandable.

CHAPTER 3

Status

As the code stands now, a 3.5 GB, single ping Workhorse ADCP .pd0 file with 3 Million ensembles will take 4-5 hours to convert. I live with this, because I can just let the conversion happen overnight on such large data sets, and once my data is in netCDF, everything else is convenient and fast. I suspect that more speed might be acheived by making use of xarray and dask to write the netCDF output, and I may do this if time allows, and I invite an enterprising soul to beat me to it. I use this code myself on a routine basis in my work, and continue to make it better as I learn more about python.

At USGS Coastal and Marine Geology we use the PMEL EPIC convention for netCDF as we started doing this back in the early 1990's. Downstream we do convert to more current CF conventions, however our diagnostic and other legacy code for processing instrument data from binary and other raw formats depends on the EPIC convention for time, so you will see a time (Time (UTC) in True Julian Days: 2440000 = 0000 h on May 23, 1968) and time2 (msec since 0:00 GMT) variable created as default. This may confuse your code. If you want the more python friendly CF time (seconds since 1970-01-01T00:00:00 UTC) set timetype to CF.

Use at your own risk - this is a work in progress and a python learning project.

Enjoy,

Marinna

3.1 adcpy.EPICstuff package

3.1.1 Submodules

3.1.2 adcpy.EPICstuff.ADCPcdf2ncEPIC module

This code takes a raw netcdf file containing data from any 4 beam Janus acoustic doppler profiler, with or without a center beam, and transforms the data into Earth coordinates. Data are output to netCDF using controlled vocabulary for the variable names, following the EPIC convention wherever possible.

ADCPcdf2ncEPIC.doEPIC_ADCPfile(cdfFile, ncFile, attFile, settings)

cdfFile = path to a USGS raw netCDF ADCP data file

ncFile = a netcdf file structured according to PMEL EPIC conventions

attFile = a file containing global attributes (metadata) for the data. See below

settings = a dictionary of preferences for the processing:

```
'good_ensembles': [0, np.inf] # starting and ending indices of the input file. For␣
↪all data use [0,np.inf]
'orientation': 'UP' # uplooking ADCP, for downlooking, use DOWN
'transducer_offset_from_bottom': 1.0 # a float in meters
'transformation': 'EARTH' # | BEAM | INST
'adjust_to_UTC': 5 # for EST to UTC, if no adjustment, set to 0 or omit
```

Depth dependent attributes are compute from the mean Pressure found in the raw data file. So it is best to have the time series trimmed to the in water time or to provide the good ensemble indices for in water time

Note that file names and paths may not include spaces

Example contents of a Global Attribute file:

```
SciPi; J.Q. Scientist
PROJECT; USGS Coastal Marine Geology Program
EXPERIMENT; MVCO 2015 Stress Comparison
DESCRIPTION; Quadpod 13.9m
DATA_SUBTYPE; MOORED
COORD_SYSTEM; GEOGRAPHIC + SAMPLE
Conventions; PMEL/EPIC
MOORING; 1057
WATER_DEPTH; 13.9
WATER_DEPTH_NOTE; (meters), nominal
WATER_DEPTH_source; ship fathometer
latitude; 41.3336633
longitude; -70.565877
magnetic_variation; -14.7
Deployment_date; 17-Nov-2015
Recovery_date; 14-Dec-2015
DATA_CMNT;
platform_type; USGS aluminum T14 quadpod
DRIFTER; 0
POS_CONST; 0
DEPTH_CONST; 0
Conventions; PMEL/EPIC
institution; United States Geological Survey, Woods Hole Coastal and Marine Science␣
↪Center
institution_url; http://woodshole.er.usgs.gov
```

Created on Tue May 16 13:33:31 2017

@author: mmartini

EPICstuff.ADCPcdf2ncEPIC.**add_VAR_DESC**(*cdf*)
add the VAR_DESC global attribute constructed from variable names found in the file

Parameters **cdf** (*object*) – netCDF file object

EPICstuff.ADCPcdf2ncEPIC.**beam2inst**(*adcpo*, *reverse=False*, *force=False*)
Rotate velocities from beam to instrument coordinates.

Parameters

- **adcpo** (*dict*) – containing the beam velocity data.

- **reverse** (*bool*) – If True, this function performs the inverse rotation (inst->beam).
- **force** (*bool*) – When true do not check which coordinate system the data is in prior to performing this rotation.

EPICstuff.ADCPcdf2ncEPIC.**cal_earth_rotmatrix**(*heading=0*, *pitch=0*, *roll=0*, *declination=0*)

this transformation matrix is from the R.D. Instruments Coordinate Transformation booklet. It presumes the beams are in the same position as RDI Workhorse ADCP beams, where, when looking down on the transducers:

```
Beam 3 is in the direction of the compass' zero reference
Beam 1 is to the right
Beam 2 is to the left
Beam 4 is opposite beam 3
Pitch is about the beam 2-1 axis and is positive when beam 3 is raised
Roll is about the beam 3-4 axis and is positive when beam 2 is raised
Heading increases when beam 3 is rotated towards beam 1
```

Nortek Signature differs in these ways:

```
TRDI beam 3 = Nortek beam 1
TRDI beam 1 = Nortek beam 2
TRDI beam 4 = Nortek beam 3
TRDI beam 2 = Nortek beam 4
Heading, pitch and roll behave the same as TRDI
```

Parameters

- **heading** (*float*) – ADCP heading in degrees
- **pitch** (*float*) – ADCP pitch in degrees
- **roll** (*float*) – ADCP roll in degrees
- **declination** (*float*) – heading offset from true, Westerly is negative

Returns

EPICstuff.ADCPcdf2ncEPIC.**calc_beam_rotmatrix**(*theta=20*, *convex=True*, *degrees=True*)

Calculate the rotation matrix from beam coordinates to instrument head coordinates. per dolfyn rotate.py code here: https://github.com/lkilcher/dolfyn

Parameters

- **theta** (*float*) – is the angle of the heads (usually 20 or 30 degrees)
- **convex** (*int*) – is a flag for convex or concave head configuration.
- **degrees** (*bool*) – is a flag which specifies whether theta is in degrees or radians (default: degrees=True)

EPICstuff.ADCPcdf2ncEPIC.**doEPIC_ADCPfile**(*cdfFile*, *ncFile*, *attFile*, *settings*)

Convert a raw netcdf file containing data from any 4 beam Janus acoustic doppler profiler, with or without a center beam, and transforms the data into Earth coordinates. Data are output to netCDF using controlled vocabulary for the variable names, following the EPIC convention wherever possible.

Parameters

- **cdfFile** (*str*) – raw netCDF input data file name
- **ncFile** (*str*) – output file name
- **attFile** (*str*) – text file containing metadata

- **settings** (*dict*) – a dict of settings as follows:

```
'good_ensembles': [] # starting and ending indices of the input␣
↪file. For all data use [0,np.inf]
'orientation': 'UP' # uplooking ADCP, for downlooking, use DOWN
'transducer_offset_from_bottom': 2.02 # in meters
'transformation': 'EARTH' # | BEAM | INST
'adjust_to_UTC': 5 # for EST to UTC, if no adjustment, set to 0 or␣
↪omit
```

EPICstuff.ADCPcdf2ncEPIC.**floor**(*dec*)

convenience function to round down provided to avoid loading the math package and because np.floor was causing unexpected behavior w.r.t ints

Parameters **dec** (*float*) –

Returns rounded number

EPICstuff.ADCPcdf2ncEPIC.**inst2earth**(*adcpo*, *reverse=False*, *fixed_orientation=False*, *force=False*)

Rotate velocities from the instrument to earth coordinates.

Parameters

- **adcpo** (*dict*) – containing the data in instrument coordinates
- **reverse** (*bool*) – If True, this function performs the inverse rotation (earth->inst).
- **fixed_orientation** (*bool*) – When true, take the average orientation and apply it over the whole record.
- **force** (*bool*) – When true do not check which coordinate system the data is in prior to performing this rotation.

Notes

The rotation matrix is taken from the Teledyne RDI ADCP Coordinate Transformation manual January 2008

When performing the forward rotation, this function sets the 'inst2earth:fixed' flag to the value of *fixed_orientation. When performing the reverse rotation, that value is 'popped' from the props dict and the input value to this function'fixed_orientation* has no effect. If *'inst2earth:fixed'* is not in the props dict then the input value *is* used.

EPICstuff.ADCPcdf2ncEPIC.**read_globalatts**(*fname*)

read_globalatts: read in file of metadata for a tripod or mooring

reads global attributes for an experiment from a text file (fname) called by all data processing programs to get uniform metadata input one argument is required- the name of the file to read- it should have this form:

```
SciPi; J.Q. Scientist
PROJECT; USGS Coastal Marine Geology Program
EXPERIMENT; MVCO 2015 Stress Comparison
DESCRIPTION; Quadpod 13.9m
DATA_SUBTYPE; MOORED
COORD_SYSTEM; GEOGRAPHIC + SAMPLE
Conventions; PMEL/EPIC
MOORING; 1057
WATER_DEPTH; 13.9
WATER_DEPTH_NOTE; (meters), nominal
WATER_DEPTH_source; ship fathometer
```

(continues on next page)

(continued from previous page)

```
latitude; 41.3336633
longitude; -70.565877
magnetic_variation; -14.7
Deployment_date; 17-Nov-2015
Recovery_date; 14-Dec-2015
DATA_CMNT;
platform_type; USGS aluminum T14 quadpod
DRIFTER; 0
POS_CONST; 0
DEPTH_CONST; 0
Conventions; PMEL/EPIC
institution; United States Geological Survey, Woods Hole Coastal and Marine␣
↪Science Center
institution_url; http://woodshole.er.usgs.gov
```

Parameters **fname** (*str*) – input file name

Returns dict of metadata

EPICstuff.ADCPcdf2ncEPIC.**setupEPICnc**(*fname*, *rawcdf*, *attfile*, *settings*)

Construct an empty netCDF output file to EPIC conventions

Parameters

- **fname** (*str*) – output netCDF file name
- **rawcdf** (*Dataset*) – input netCDF raw data file object
- **attfile** (*str*) – metadata text file
- **settings** (*dict*) – settings as follows:

```
'good_ensembles': [] # starting and ending indices of the input␣
↪file. For all data use [0,np.inf]
'orientation': 'UP' # uplooking ADCP, for downlooking, use DOWN
'transducer_offset_from_bottom': 2.02 # in meters
'transformation': 'EARTH' # | BEAM | INST
'adjust_to_UTC': 5 # for EST to UTC, if no adjustment, set to 0 or␣
↪omit
```

Returns netCDF file object

EPICstuff.ADCPcdf2ncEPIC.**writeDict2atts**(*cdfobj*, *d*, *tag*)

write a dictionary to netCDF attributes

Parameters

- **cdfobj** (*object*) – netcdf file object
- **d** (*dict*) – metadata
- **tag** (*str*) – tag to add before each atrribute name

Returns dict of metadata as written to file

3.1.3 adcpy.EPICstuff.EPICmisc module

Helper functions, mostly EPIC specific

EPICstuff.EPICmisc.**EPICtime2datetime**(*time*, *time2*)
convert EPIC time and time2 to python datetime object

Parameters

- **array time** (*numpy*) –
- **array time2** (*numpy*) –

Returns gregorian time as a list of int, datetime object

EPICstuff.EPICmisc.**ajd**(*dto*)
Given datetime object returns Astronomical Julian Day. Day is from midnight 00:00:00+00:00 with day fractional value added.

Parameters **dto** (*object*) – datetime

Returns int Astronomical Julian Day

EPICstuff.EPICmisc.**apply_timezone**(*cf_units*)

In xarray, the presence of time zone information in the units was causing decode_cf to ignore the hour, minute and second information. This function applys the time zone information and removes it from the units

Parameters **cf_units** (*str*) –

Returns str

EPICstuff.EPICmisc.**catEPIC**(*datafiles*, *outfile*)

EPICstuff.EPICmisc.**cftime2EPICtime**(*timecount*, *timeunits*)

EPICstuff.EPICmisc.**check_fill_value_encoding**(*ds*)

restore encoding to what it needs to be for EPIC and CF compliance variables' encoding will be examined for the correct _FillValue

Parameters **ds** – xarray Dataset

Returns xarray Dataset with corrected encoding, dict with encoding that can be used with xarray.to_netcdf

EPICstuff.EPICmisc.**fix_missing_time**(*ds*, *delta_t*)
fix missing time values change any NaT values in 'time' to a time value based on the last known good time, iterating to cover larger gaps by constructing time as we go along. xarray.DataArray.dropna is one way to do this, automated and convenient, and will leave an uneven time series, so if you don't mind time gaps, that is a better tool.

Parameters

- **ds** – xarray Dataset, time units are in seconds
- **deltat** – inter-burst time, sec, for the experiment's sampling scheme

Returns

EPICstuff.EPICmisc.**jdn**(*dto*)
convert datetime object to Julian Day Number

Parameters **dto** (*object*) – datetime

Returns int Julian Day Number

EPICstuff.EPICmisc.**make_encoding_dict**(*ds*)
prepare encoding dictionary for writing a netCDF file later using xarray.to_netcdf

Parameters **ds** – xarray Dataset

Returns dict with encoding prepared for xarray.to_netcdf to EPIC/CF conventions

EPICstuff.EPICmisc.**resample_cleanup**(*datafiles*)

EPICstuff.EPICmisc.**s2hms**(*secs*)
convert seconds to hours, minutes and seconds

Parameters **secs** (*int*) –

Returns hours, minutes and seconds

3.1.4 adcpy.EPICstuff.repopulateEPIC module

repopulateEPIC

Distribute burst data output from reshapeEPIC along the sample dimension

a burst shaped file output from reshapeEPIC will have two issues that need to be addressed before the data can be used with xarray:

```
-- reshape time to be one dimension
-- make sure the samples within each burst are index according to their
time stamps.  Within burst time stamps will not be preserved
```

Usage: python repopulateEPIC.py shaped_file new_file sample_rate [start='left'] [drop = None]

param str shaped_file output from reshapeEPIC, the expected shape of the data is one of:: [time, sample] [time, sample, depth] [time, sample, depth, lat, lon]

param str new_file a new file with the adjusted time, this file, if it exists, will be overwritten

param int sample_rate the sample rate the instrument was intended to use during each burst, in seconds

param str start what the time stamp should be for each burst:: left = beginning of the burst based on the first sample time center = middle of the burst based on first sample and last sample times right = end of the burst based on the last sample time

param list drop variable names to omit from the output file

Created on Wed Oct 3 15:21:53 2018 @author: mmartini

EPICstuff.repopulateEPIC.**repopulateEPIC**(**args*, ***kwargs*)

3.1.5 adcpy.EPICstuff.reshapeEPIC module

reshapeEPIC

apportion a continuous time series file into bursts (e.g. reshape)

Notes

- the expected dimensions are [time, depth, lat, lon] for continuous data in EPIC
- we are reshaping to [time, sample, depth, lat, lon]
- for ADCP files, beams are expressed as variables vel1, vel2, ... veln
- if there is a shape problem then the data file might not be properly understood.

It might be that this code won't work, this problem will become evident if an error is produced and operation returns to the keyboard in debug mode. If this happens, check the shapes of the variables.

WARNING: time may not be monotonically increasing within bursts (e.g. along the sample dimension) this means that if the number of samples per burst is inconsistent, or if there are gaps in time, the end of a burst may be fill_value, including time values

Marinna Martini for the USGS in Woods Hole, 9/20/2018 originally coded for MATLAB as reshapeEPIC https://cmgsoft.repositoryhosting.com/trac/cmgsoft_m-cmg/browser/trunk/MMstuff/reshapeEPIC.m

Created on Thu Sep 20 14:52:42 2018

@author: mmartini https://github.com/mmartini-usgs

EPICstuff.reshapeEPIC.**find_boundaries**(*data*, *edges*)

using a list of start and end timestamps (edges) that delineate the beginning times and ending times of burts of measurements, find the indices into the data that correspond to these edges. The time base may be irregular, it does not matter.

Parameters

- **data** (*list*) – time stamps from the data
- **edges** (*list[tuple]*) – start and end times

Returns list of indices

EPICstuff.reshapeEPIC.**find_first_masked_value**(*x*)

helper function to find the first occurrence of a masked value in a numpy masked array returns None if no masked values are found :param numpy array x: :return: index

EPICstuff.reshapeEPIC.**generate_expected_start_times**(*cdffile*, *dim*, *burst_start_offset*, *burst_interval*, *burst_length*, *sample_rate*)

generate a regular and recurring set of start and end timestamps that delineate the beginning times and ending times of burts of measurements

Parameters

- **cdffile** (*str*) – name of a continuous time series data file
- **dim** (*str*) – the unlimited or time dimension which we will find the indices to reshape
- **burst_start_offset** (*int*) – when to start to make bursts in the continuous data, seconds
- **burst_interval** (*int*) – time between start of bursts, seconds
- **burst_length** (*int*) – number of samples in a burst
- **sample_rate** (*int*) – Hertz

Returns list of tuples of start and end times for each burst

EPICstuff.reshapeEPIC.**reshapeEPIC**(*cont_file*, *burst_file*, *burst_length*, *dim='time'*, *edges=None*, *drop=None*, *variable_attributes_to_omit=None*, *verbose=False*)

apportion a continuous time series file into bursts (e.g. reshape)

Usage issue_flags = reshapeEPIC(cont_file, burst_file, burst_length, dim=None, edges=None, drop=None)

Parameters

- **cont_file** (*str*) – name of netCDF file with continuous data
- **burst_file** (*str*) – name of file to store the reshaped data, attributes will be copied
- **burst_length** (*int*) – maximum number of samples in each burst
- **dim** (*str*) – name of dimension along which we will split the data, usually 'time' or 'Rec'
- **edges** (*list[tuple]*) – [(start0, end0), (start1, end1), ...] of edges defining the edges of each burst
- **drop** (*str*) – set of variable names to omit from the output file
- **variable_attributes_to_omit** (*str*) – variable attributes to omit from output file
- **verbose** (*bool*) – get lots of feedback to STDOUT

Returns dictionary of problem types and status

EPICstuff.reshapeEPIC.**save_indexes_to_file**(*cdffile*, *edge_tuples*, *index_file=None*)

write indexes to a file with the time stamps for QA/QC

Parameters

- **cdffile** (*str*) – the continuous time series netCDF file being operated upon
- **edge_tuples** (*list[tuple]*) – the bursts to output
- **index_file** (*str*) – a file to output a string listing of time stamps

3.1.6 Module contents

3.2 adcpy.Nortekstuff package

3.2.1 Submodules

3.2.2 adcpy.Nortekstuff.Norteknc2USGScdf module

3.2.3 Module contents

3.3 adcpy.TRDIstuff package

This part of ADCPy handles raw data from Teledyne RD Instruments Acoustic Doppler Profilers. Raw binary data are commonly in a format called pd0.

3.3.1 Submodules

3.3.2 adcpy.TRDIstuff.TRDIpd0tonetcdf module

3.3.3 adcpy.TRDIstuff.pd0 module

3.3.4 adcpy.TRDIstuff.pd0splitter module

3.3.5 Module contents

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

W