EGADS Documentation

EUFAR FP7

N6SP - Standards and Protocols

Last Updated: November 10, 2010

Contents

Chapter 1

Introduction

The motivation behind EGADS (EUFAR General Airborne Data-processing Software) was to create a general method for processing airborne atmospheric sciences data from all...

The current EGADS implementation allows users to interact with data processing algorithms using the Python command-line or via Python scripts. Python was chosen due to its straightforward syntax and flexibility between multiple systems. Central to EGADS is a data type that encapsulates data and metadata into one object. This simplifies the housekeeping of such data, and allows it to be easily passed between algorithms and data files.

Chapter 2

Installation

The latest version of EGADS can be obtained from http://eufar-egads.googlecode.com

2.1 Prerequisites

Building EGADS requires the following software to be installed:

Python 2.5 or newer. Available at http://www.python.org/.

numpy 1.3.0 or newer. Available at http://numpy.scipy.org/.

scipy 0.6.0 or newer. Available at http://www.scipy.org/.

Python netCDF4 libraries 0.8.2. Available at http://code.google.com/p/netcdf4-python/.

2.2 Optional Packages

The following are useful when using or compiling EGADS :

IPython An optional package which simplifies Python command line usage (http://ipython.scipy.org). IPython is an enhanced interactive Python shell which supports tab-completion, debugging, history, etc.

2.3 Installation

To install EGADS , type `python setup.py install` from the command line. To install to a user-specified location, type `python setup.py install --prefix=$MYDIR`.

2.4 Testing

To test EGADS after it is installed, run the following commands in Python:

```
>>> import egads
>>> egads.test()
```

Chapter 3

Tutorial

3.1 Command-line usage

The simplest way to start working with EGADS is to run it from the Python command line. To load egads into the Python namespace, simply import it:

```
>>> import egads
```

You may then begin working with any of the algorithms contained in EGADS .

3.1.1 Exploring egads

There are several useful methods to explore routines contained in EGADS . The first is using the Python built-in `dir()` command:

```
>>> dir(egads)
```

returns all the classes and subpackages contained in EGADS . EGADS follows the naming conventions from the Python Style Guide (http://www.python.org/dev/peps/pep-0008), so classes are always `MixedCase`, functions and modules are generally `lowercase` or `lowercase_with_underscores`. As a further example,

```
>>> dir(egads.input)
```

would returns all the classes and subpackages of the `egads.input` module.

Another way to explore EGADS is by using tab completion, if supported by your Python installation. Typing

```
>>> egads.
```

then hitting TAB will return a list of all available options.

Python has built-in methods to display documentation on any function known as docstrings. The easiest way to access them is using the `help()` function:

```
>>> help(egads.input.NetCdf)
```

will return all methods and their associated documentation for the NetCdf class.

3.1.2 Working with NetCDF files

EGADS provides two classes to work with NetCDF files. The simplest, `egads.input.NetCdf()`, allows simple read/write operations to NetCDF files. The other, `egads.input.EgadsNetCdf()`, is designed to interface with NetCDF files conforming to the N6SP data and metadata regulations. This class directly reads or writes NetCDF data using instances of the `EgadsData` class.

Opening

To open a NetCDF file, simply create a `egads.input.NetCdf()` instance and then use the `open(pathname, permissions)` command:

```
>>> import egads
>>> f = egads.input.NetCdf()
>>> f.open('/pathname/filename','r')
```

Valid values for permissions are:

- `r` – Read: opens file for reading only. Default value if nothing is provided.
- `w` – Write: opens file for writing, and overwrites data in file.
- `a` – Append: opens file for appending data.
- `r+` – Same as 'a'.

Getting info

- `f.get_dimensions()` – returns list of all dimensions and their sizes
- `f.get_dimensions(var_name)` – returns list of all dimensions for `var_name`
- `f.get_attributes()` – returns a list of all top-level attributes
- `f.get_attributes(var_name)` – returns list of all attributes attached to `var_name`
- `f.get_variables()` – returns list of all variables
- `f.get_filename()` – returns filename for currently opened file

Reading data

To read data from a file, use the `read_variable()` function:

```
>>> data = f.read_variable(var_name, input_range)
```

where `var_name` is the name of the variable to read in, and `input_range` (optional) is a XXXXX

If using the `egads.input.NetCdf()` class, an array of values contained in `var_name` will be returned. IF using the `egads.input.EgadsNetCdf()` class, an instance of the `EgadsData()` class will be returned containing the values and attributes of `var_name`.

Writing data

The following describe how to add dimenisons or attributes to a file.

- `f.add_dim(dim_name, dim_size)` – add dimension to file
- `f.add_attribute(attr_name, attr_value)` – add attribute to file
- `f.add_attribute(attr_name, attr_value, var_name)` – add attribute to `var_name`

Data can be output to variables using the `write_variable()` function as follows:

```
>>> f.write_variable(data, var_name, dims, type)
```

where `var_name` is a string for the variable name to output, `dims` is a tuple of dimension names (not needed if the variable already exists), and `type` is the data type of the variable. The default value is *double*, other valid options are *float*, *int*, *short*, *char* and *byte*.

If using `egads.input.NetCdf()`, values for `data` passed into `write_variable` must be scalar or array. Otherwise, if using `egads.input.EgadsNetCdf()`, an instance of `EgadsData()` must be passed into `write_variable`. In this case, any attributes that are contained within the `EgadsData()` instance are applied to the NetCDF variable as well.

Closing

To close a file, simply use the `close() command`:

```
>>> f.close()
```

3.2 Scripting

Another option for using EGADS is to create script files, which are useful for common or repetitive tasks. This can be done using any available text editor. The example script belows shows the calculation of density for all NetCDF files in a directory.

```
#!/usr/bin/env python

import egads
import egads.algorithms.thermodynamics as thermo

filenames = egads.get_file_list('data/*.nc')

f = egads.input.EgadsNetCdf()   # create EgadsNetCdf instance

for name in filenames:          # loop through files

    f.open(name, 'a')            # open NetCdf file

    T_s = f.read_variable('T_t') # read in static temperature
    P_s = f.read_variable('P_s') # read in static pressure from file

    rho = thermo.density_dry_air_cnrm(P_s, T_s)  # calculate density

    f.write_variable(rho, 'rho', ('Time',))      # output variable

    f.close()                                    # close file
```

Appendix A

Class reference