Python Tools to Create Data Files

A Python object is available to create and manipulate data files which are required by the DOE open access requirements. The files created are in the HDF5 format, with a structure for the data and metadata corresponding to our recommended specs – available in "Recommendations for format of data associated with publications". This should allow any user to readily use the information contained. Questions or suggestions should go to Josh Stillerman (jas@mit.edu)

The python module is available for import on the C-Mod cluster

```
$ python
Python 2.7.11 (default, Sep 29 2016, 13:33:00)
[GCC 5.3.1 20160406 (Red Hat 5.3.1-6)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from h5_data import h5_data
```

The module defines a constructor for the object:

```
>>> help(h5_data)

|
| __init__(self, filename,
|              user_fullname=None, fig_description=None,
|              fig_source=None, comment=None)
|     Create an h5_data object.
|
|     If the file exists, the file level metadata will be
|       updated with any specified arguments.
|     If the file does not exist, it will be created with all specified
|       file level metadata.
|
|     @param file-name: Name of the file to create or open
|     @type file-name: str
|     @param user_fullname: Optional user_fullname, name of the user
|       creating this file
|     @type user_fullname: str
|     @param fig_description: Optional description of this figure.
|     @type fig_description: str
|     @param source: Optional source for this file
|       (e.g. 'Phys. Plasmas 17, 1234 2010')
|     @type sourcet: str
|     @param comment: Optional file level comment for this figure
|     @type comment: str
|
```

The which in turn has method add_dataset:

```
| add_dataset(self, name, legend=None, plot_info=None,
|               x_data=None, x_units=None, x_label=None,
|               y_data=None, y_units=None, y_label=None,
|               z_data=None, z_units=None, z_label=None)
|     Add a data_set (trace) to an existing Figure file
|
```

```
|     @param name: Name of this data set
|     @type name: str
|     @param legend: Optional legend for this trace
|     @type legend: str
|     @param plot_info: Optional information on plot representation
|                       (e.g. 'blue squares').
|     @type plot_info: str
|
|     @param x_data: Data for the X axis
|     @type x_data: numeric
|     @param x_units: Optional units for X axis
|     @type x_units: str
|     @param x_label: Optional label for X axis
|     @type x_label: str
|
|     @param y_data: Data for the Y axis
|     @type y_data: numeric
|     @param y_units: Optional units for Y axis
|     @type y_units: str
|     @param y_label: Optional label for Y axis
|     @type y_label: str
|
|     @param z_data: Data for the Z axis
|     @type z_data: numeric
|     @param z_units: Optional units for Z axis
|     @type z_units: str
|     @param z_label: Optional label for Z axis
|     @type z_label: str
```

To use the h5_data object:

- Import the module
  >>> from h5_data import h5_data
- Create the object
  hdf_file = h5_data("figure_1.hdf5",
  fig_description = “A description of the figure”,
  fig_source = “Source of the figure”,
  comment = “Comment about the figure”,
  user_fullname = “Your full name”)
- Add a data set to the file
  x_units = “s”
  x_label = “time (s)”
  y_units = “m”
  y_label = “height (m)”
  hdf_file.add_dataset('J0',
  legend=None, plot_info='Blue Line',
  x_data=x, x_units=x_units, x_label=x_label,
  y_data=y0, y_units=y0_units, y_label=y0_label)

For each data file, corresponding to each figure, the user would first construct an h5_data object by calling h5_data(arguents). Then, each time a new trace or group of data points is added, using the add_dataset method. The arguments to the add_dataset method provide the x, y (and z) data. For both procedures, keywords containing the metadata are used. This method can be used as many times as needed, the file is opened and closed automatically each time it is called.

Full listing of metadata names used in the object and method:

file level metadata (with the constructor)

- file-name: Name of the file to create or open
- user_fullname: Optional user_fullname
- name of the user creating this file
- fig_description: Optional description of this figure.
- source: Optional source for this file
  (e.g. 'Phys. Plasmas 17, 1234 2010')
- comment: Optional file level comment for this figure

Group/dataset level metadata

- group_name = string - name of data group – first argument (not a keyword)
- legend = string description of this data group that distinguishes it
  from other data groups in same plot
- plot_info = string - optional description of plot style for this data group
- example 'red filled circles' or 'blue solid line'
- similar to what might appear in the figure caption

Trace level metadata

- x = float or integer array
- x_units = string
- x_axis = string - label for x axis
- x_name = string - optional longer description of x data

- y = float or integer array - should be same size as x in 2d plot
- y_units = string
- y_axis = string - label for y axis
- y_name = string - optional longer description of y data
- y_type = string - optional data type

- optionally for 3D plot
- z = float or integer array - should be same size and shape as x#y
- z_units = string
- z_axis = string - label for z axis
- z_name = string - optional longer description of z data

Complete example program

```
$ cat example.py
#!/usr/bin/python

from scipy import linspace
from scipy.special import jv
from h5_data import h5_data
import matplotlib.pyplot as plt
"""
  Define some variables for file level metadata
"""
file_name = 'Fig_3'
fig_description = 'Besel Functions J0, J1 and J2'
fig_source = 'Phys. Plasmas 17, 1234 2010'
comment = 'This is the way the ball bounces'
user_fullname = 'John Doe'

"""
  Draw the first trace.
"""
x = linspace(0, 20)
y0 = jv(0,x)
plt.plot(x,y0, '-b', label='J0')
x_units='s'
x_label='time (s)'
y0_units='m'
y0_label='height (m)'

"""
  Draw the 2nd trace.
"""
y1 = jv(1,x)
plt.plot(x, y1, '-g', label='J1')
y1_units='m'
y1_label='height (m)'

"""
  Draw the third trace.
"""
y2 = jv(2,x)
plt.plot(x, y2, '-r', label='J2')
y2_units='m'
y2_label='height (m)'

"""
  Add Labels and legend
"""
plt.title(fig_description)
plt.xlabel(x_label)
plt.ylabel(y0_label)
plt.legend(loc='upper right')
plt.show()
"""
  Create an hdf5 file to hold the data for this figure.
```

```
  Annotate it with the file-level metadata.
"""
hdf_file = h5_data("%s.hdf5"%(file_name,),
                fig_description = fig_description,
                fig_source = fig_source,
                comment = comment,
                user_fullname = user_fullname)

"""
  Add the first data set, with its metadata.
"""
hdf_file.add_dataset('J0',
                 legend=None, plot_info='Blue Line',
                 x_data=x, x_units=x_units, x_label=x_label,
                 y_data=y0, y_units=y0_units, y_label=y0_label )

"""
  Add the second data set, with its metadata.
"""
hdf_file.add_dataset('J1',
                 legend=None, plot_info='Green Line',
                 x_data=x, x_units=x_units, x_label=x_label,
                 y_data=y1, y_units=y1_units, y_label=y1_label )

"""
  Add the third data set, with its metadata.
"""
hdf_file.add_dataset('J2',
                 legend=None, plot_info='Red Line',
                 x_data=x, x_units=x_units, x_label=x_label,
                 y_data=y2, y_units=y2_units, y_label=y2_label )
```