Using Python to Read HDF5 files from the Global Precipitation Measurement (GPM) mission

The NASA Goddard Precipitation Processing System, http://pps.gsfc.nasa.gov
Written by Owen.Kelley@nasa.gov
Updated: 22 June 2015

1. Purpose

The Global Precipitation Measurement (GPM) core satellite carries a radar and a passive microwave instrument that has been in earth orbit since February 2014. In addition to generating data products from these two instruments, the Precipitation Processing System (PPS) at NASA Goddard generates multi-satellite precipitation data products. The GPM standard products are written the HDF5 file format, and to quote the Preface of Collette (2013), "Over the past several years, Python has emerged as a credible alternative to scientific analysis environments like IDL or Matlab." Python is a programming language that can access HDF5 files if the optional h5py module is installed, as described in Collette's 2013 book, *Python and HDF5*. This document describes a short Python program developed at PPS to help researchers take a first look at the GPM mission's HDF5 files.

The sample Python program described in this document can be downloaded from the PPS website. This sample program illustrates how to read GPM HDF5 data into Python and a simple way to display these data after they are read. This Python program is intended to work on Linux, Mac OS X, and Windows 7 systems that have Python version 2.7 install on them. The numpy, matplotlib, and h5py modules must be installed in your copy of Python for the sample Python program to work.

Before proceeding with this document, you may wish to consider if Python is the best method for you to read and display GPM HDF5 files. Three of the ways to access GPM HDF5 files include the following: writing a program in a low-level language (C or FORTRAN), write a program in a high-level language (IDL, Matlab, or Python), or use the point-and-click graphical interface of a Geographic Information System (GIS). The advantage of a high-level language is that you don't have to write as many lines of code as is required with a low-level language. Among the high-level languages, the amount of code that must be written is comparable, but ease of installation varies. High-level languages for which you purchase a license (i.e., IDL and Matlab) tend to be easy to install and have tech support if you run into problems. Open-source languages (i.e., Python) can be difficult to install and have limited tech support. This document can only provide a few hints about installing Python on various platforms and some of these hints may soon become out-of-date, due to the shifting nature of Python and Python modules.

In particular, many online sources warn against trying to install Python under various version of Microsoft Windows because so many installation problems crop up. Even if you are able to figure out how to install the Python modules you need on a Windows system after a few days of work, the next time you need to install Python, you may need to work through a new set of problems. Just because you find a way to install Python and the required modules under

Windows that doesn't mean that your colleagues, with slightly different Windows systems or a different set of already-installed applications will also be able to discover a way to install Python and those modules on their Windows system.

In other words, if you are already have a working copy of Python installed on your computer and some Python programming experience, then this document may be useful to you. For more information about GPM, please visit the science team's website (http://pmm.nasa.gov) or read Hou et al. (2014).

2. Installation

For Linux or Mac users, determine if you have Python 2.7 installed on your machine by typing "python" on the command line. For Windows users, type "python" in the search field of your start menu. If you do not have Python installed, then would need to install it plus three Python modules before you would be able to run the sample Python program for reading GPM HDF5 files. These three Python modules are numpy, matplotlib, and h5py. The rest of this section provides a few tips about installing Python on Windows 7 systems or on Linux and Mac systems. There is no guarantee that these tips will enable you to install Python or these Python modules. In fact, you could harm your system by trying to install Python and these modules. Please consult your system administrator before following any of the suggestions below.

2a. Windows 7: Installing Python language and required Python modules

To give a flavor of what you are up against if you want to try to access HDF5 files with Python under Windows, here is what occurred on our test machine in June of 2015. None of the following four Python bundles could be installed on our test machine: Anaconda, Pythonxy, winpython, or enthought. Even installing pure Python failed, unless the following things were done: (1) Python had to be installed in the default directory of C:\Python27 and (2) the files that ESRI's ArcGIS had placed in C:\Python27 had to be deleted before doing a fresh Python install from http://www.python.org.

After trial and error on each stage, the following steps worked on our Windows 7 machine in June of 2015. The steps are listed here in case they might be at least partially relevant to some other user's machines. The following list of steps evolved from the list in Glipin (2013).

(1) Install pure Python, version 2.7. From http://www.python.org, download the python2.7.10.msi Microsoft Software Installer (MSI) file for 64-bit window. Run that executable by double click in it. Install it in the default location presented, which is C:\Python27. After installation completes, looking in the C:\Python27\Scripts folder to verify that this folder includes a file called easy_install.exe because this executable is essential for subsequent steps.

(2) Add Python to your system's "Path" environment variable. To do this under Windows 7, click on Start and then right click on "Computer". Select Properties from the window that pops up. Click on Advanced on the left hand column. Click on the Environment Variables button. Edit the system "Path" to begin with the phrase "C:\Python27;C:\Python27\Scripts;".

(3) Verify that python itself is working. To do this, open a Microsoft Windows "Command" window, which is a window that has a command line that allows you to execute DOS commands. To open a command window, go to the Start menu and type "command" or "cmd.exe" in the search field. In the list of search results, click on "Command Prompt". In any directory, you should be able to type "python" or "python.exe" in the DOS prompt and Python 2.7 should start up. "python.exe" it located in C:\Python27\python.exe", but you added that directory to your Path environment variable in the previous step.

(4) Install Microsoft Visual C++ 9.0 from http://aka.ms.vcpython27 . This means downloading VCForPython27.msi and double clicking on it to install. Visual C++ is necessary in order for the Python installer to combine C code as part of the installation of the matplotlib Python module. The Python installer is unable to install Visual C++, and that is why you must do so yourself.

(5a) Install the Python math module, known as numpy, using in Windows "Command" window. Use "pip install numpy" on the DOS command line. Unfortunately, one of the steps of installing numpy can take an hour, during which time you will get no feedback prior to the installation completing successfully. Verify that numpy installed successfully by typing "import numpy" in Python.

(5b) Install the Python plotting module, known as matplotlib. Use "pip install matplotlib". This will take under one minute. Verify that the matplotlib Python module installed successfully by typing "import matplotlib" in Python.

(5c) Install the Python HDF5 module, known as h5py. You may want to type "easy_install h5py" because "pip install h5py" fails in our tests. Verify that the h5py Python module installed successfully by typing "import h5py" in Python.

A few notes about working in Windows: To edit a text file in Windows 7 command prompt, use "start notepad filename" for a file created on a windows system or "start wordpad filename" for a file created on a Linux system. To get a directory listing use "dir" instead of Unix "ls".

2b. Linux and Mac OS X: Installing Python language and required Python modules

On Linux and Mac systems, Python 2.6 or Python 2.7 may already be installed on your system. Verify that you have the numpy, matplotlib, and h5py modules installed, and if not, install them.

2b. Obtain the Python program and sample HDF5 files

To obtain the example program and sample HDF5 files, visit the PPS FTP site: ftp://gpmweb2.pps.eosdis.nasa.gov/pub/THOR/python and download gpmPython.zip . On a Linux or Mac system type "unzip –qq gpmPython.zip" to extract the files. On a Windows system, right click on the *.zip file and select "extract files". Below are the contents of the gpmPython.zip file:

```
2A.GPM.DPR.V5.20140409-S190631-E191227.000640.V03B.subset.HDF5
2A.GPM.GMI.GPROF.20140409-S190623-E191330.000640.V03C.subset.HDF5
3B-HHR.MS.MRG.3IMERG.20140409-S190000-E192959.1140.V03D.HDF5
gpm.py
gpmPythonNotes.pdf
```

The files in the *.zip file may be described as follows. The sample Python program is called gpm.py. Two of the HDF5 files are a single-instrument rainfall estimate from the main GPM instruments: the Dual-frequency Precipitation Radar (DPR) and the passive microwave radiometer called the GPM Microwave Imager (GMI) (Hou et al. 2014). To keep the *.zip file from becoming too large, the DPR and GMI files included in it are subsets of the original HDF5 files in the GPM archive. These subsets are geographically smaller (they contain less than a full-orbit of data) and they contain only a sample of the variables in the archived files. Incidentally, these two subsets were generated using the publically available subset feature of the PPS data ordering system called STORM (http://storm.pps.eosdis.nasa.gov). The third HDF5 file included in the *.zip file is an example of the IMERG multi-satellite precipitation data product. The IMERG product may be the most widely used GPM data product. For a 30-minute period, it provides average rainfall rates estimates that cover the globe with grid boxes 0.1 by 0.1 degree in size. The IMERG HDF5 file and the two single-instrument HDF5 files were chosen because they contain observations of Tropical Cyclone Ita approaching Australia at 1908 UTC on 9 April 2014.

Once you have finished running the example program on the sample HDF5 files, you may wish to download other GPM HDF5 files from the PPS archive. Before doing so, register your email address with PPS by going to this URL: http://registration.pps.eosdis.nasa.gov/ . To download Realtime GPM HDF5 files, go the ftp://jsimpson.pps.eosdis.nasa.gov/data/ using a web browser or using some other FTP client such as the Linux "ftp" command. Type in your just-registered email address when prompted for a username and password. To download Production GPM HDF5 files, go to ftp://arthurhou.pps.eosdis.nasa.gov/gpmdata/ . As with Realtime files, type in your just-registered email address when prompted for a username and password.

3. Running the Python program

Start a Python 2.7 session.

Load the sample program as a Python module typing "import gpm". The Python source file for this program is the gpm.py text file that you extracted from the gpmPython.zip file that you downloaded from the PPS website, as described in the previous section. Since this is Python, the gpm.py source file is text. On a Linux or Mac system, you can use the "vi" text editor to examine the contents of gpm.py and on a Windows system, you can use WordPad.

Once you have loaded the "gpm" module, you can use the Python help( ) function to obtain information about methods available to you from that module. Allowing for variations that may occur as gpm.py continues to be developed at PPS, the output of "help(gpm)" will look something like the following:

```
NAME
    gpm

FILE
    /.../python/gpm.py
```

```
DESCRIPTION
    gpm.py is a Python program written by PPS to provide an introduction
    to reading the HDF5 files from the Global Precipitation Mission(GPM).
    Documentation for this Python program is available one the PPS
    homepage, http://pps.gsfc.nasa.gov .

FUNCTIONS
    listContents(fileName, outputFile=None)
        Obtain a text list of the variables in an HDF5 file.

    plotGrid(grid)

    plotSwath(swath, swathRange)

    readGrid(fileName, varName, grid)

    readSwath(fileName, varName, swath)

    testRun()
        Run this function to test some of the read and display functions
        in this Python module.
```

The Python interrogation function dir( ) provides similar, but more concise information about the gpm module when you type "dir(gpm)" at the Python prompt.

Using a text editor, examine the contents of the testRun( ) function near the bottom of the gpm.py Python source file. You can pick and choose commands from testRun( ) remembering to prefix "gpm" on any function name. For example, to run the readSwath( ) function, you would type "gpm.readSwath(fileName,varName,swath)" on the Python command line after defining the required input variables fileName and varName. Alternatively, one can just run the entire suite of tests by typing "gpm.testRun( )" on the Python command line. When a python plot window pops up, dismiss it in order to pass on to the next test. At this time, three display windows will pop-up when you type gpm.testRun( ) on the Python command line.

If you want to modify lines of code in gpm.py, you can do so with a text editor. Then to run the modified version of gpm.py, type "reload(gpm)" on the Python command line.

The sample program contains Python functions to read and display swath and grid files containing GPM HDF5 data. The sample program also contains a function for creating a list of all variables inside of an HDF5 file. We describe these in turn.

A swath file generally contains data that was collected during a single orbit of a satellite. Each pixel observed by the satellite is associated with a latitude and longitude value. For this reason, one generally reads a data variable plus the associated latitude and longitude variables. The gpm.readSwath( ) function in the sample program reads these three variables and returns them in a Python dictionary called "swath". The swath dictionary is then passed to the gpm.plotSwath( ) function. The images generated from the GMI and DPR swath files are shows in Fig. 1.

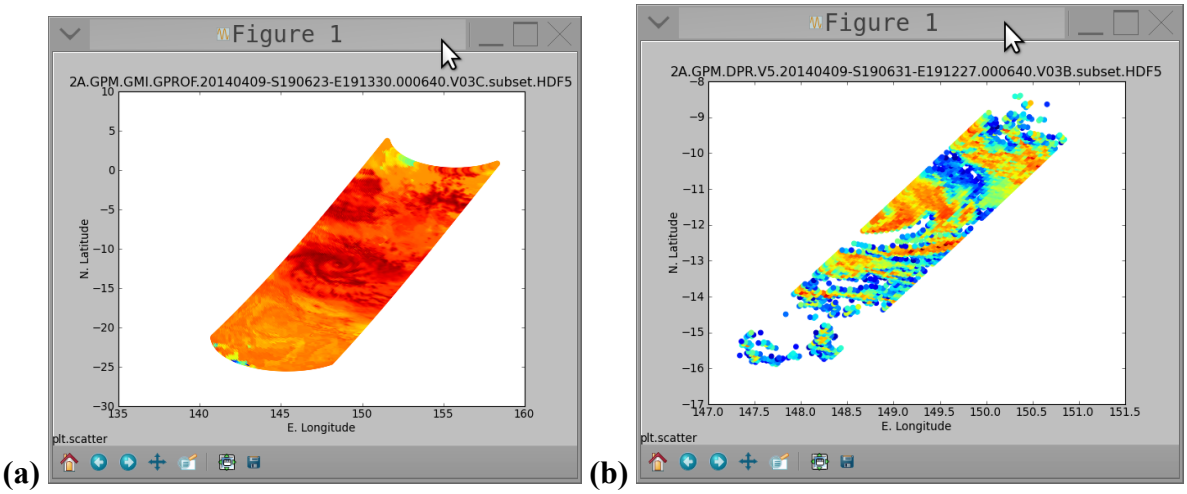

Fig. 1. Python displays of GPM single-instrument HDF5 files created by the gpm.plotSwath( ) function in the sample program gpm.py. (a) GMI surface precipitation rate (millimeters per hour). (b) DPR estimated-surface precipitation rate (mm h^{-1}).

A grid file generally contains rectangular grids that cover the whole globe's longitude (180°E to 180°W) and covers all or part of the globe's latitude (i.e., 70°S to 70°N or 90°S to 90°N) with each grid cell having a fixed size in degrees latitude and longitude. The gpm.readGrid( ) function reads a specified grid variable and gpm.plotGrid( ) displays it graphically.

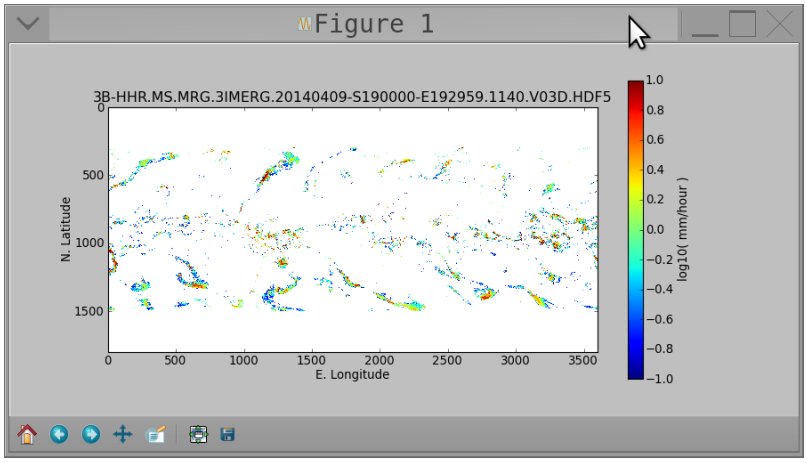

Fig. 2. Python display of the GPM multi-satellite product called IMERG, created by the gpm.plotGrid( ) function in the sample program gpm.py. The display is the base-10 logarithm of 30-minute averaged rainfall rate.

The GPM file specifications give you the names of variables in GPM HDF5 files. To access them in Python (or in IDL or Matlab for that matter), it can be helpful to know the

absolute path to the variable. The GPM file specification is available as a PDF file from the PPS website, http://pps.gsfc.nasa.gov. For example in a 1C-GMI HDF5 file, latitude is stored in a variable whose absolute path within the HDF5 file is "/S1/Latitude". To provide a list of the absolute path for all datasets in an HDF5 file, you can use the gpm.listContents( ) function in the sample Python program gpm.py. If you provide gpm.listContents( ) with only one argument (the name of the HDF5 file to be examined), then the list of variables will be printed to the screen. If you provide gpm.listContents( ) with two arguments (the HDF5 file to be examined and a name for an output text file), then the list of variables will be printed to the output filename provided. Below is what is the list of contents that is printed to the output file (or to the screen) when you give gpm.listContents( ) the name of an IMERG HDF5 grid file.

```
file: 3B-HHR.MS.MRG.3IMERG.20140409-S190000-E192959.1140.V03D.HDF5

/Grid GridHeader attribute:
BinMethod=ARITHMETIC_MEAN;
Registration=CENTER;
LatitudeResolution=0.1;
LongitudeResolution=0.1;
NorthBoundingCoordinate=90;
SouthBoundingCoordinate=-90;
EastBoundingCoordinate=180;
WestBoundingCoordinate=-180;
Origin=SOUTHWEST;

/Grid/IRkalmanFilterWeight  int16(3600, 1800)
/Grid/HQprecipSource  int16(3600, 1800)
/Grid/lon  float32(3600,)
/Grid/precipitationCal  float32(3600, 1800)
/Grid/precipitationUncal  float32(3600, 1800)
/Grid/lat  float32(1800,)
/Grid/HQprecipitation  float32(3600, 1800)
/Grid/probabilityLiquidPrecipitation  int16(3600, 1800)
/Grid/HQobservationTime  int16(3600, 1800)
/Grid/randomError  float32(3600, 1800)
/Grid/IRprecipitation  float32(3600, 1800)
```

Fig. 3. The list of variables in the IMERG HDF5 file as shown in the text file output generated by the gpm.listContents( ) function of the gpm.py Python program.

4. Some useful Python commands

Once you know the full path to the HDF5 variable that you want to read, actually reading it in Python take just two lines of code. For example, to read the GMI surface precipitation estimate from an HDF5 file, you would open the file and read the data like this, if the name of the HDF5 file was stored in the Python string called **`fileName`**:

```
fileHandle = h5py.File( fileName, 'r' )
data = fileHandle[ '/S1/surfacePrecipitation' ]
```

It is temping to think of the objects, like the **`data`** object just read, as arrays of the sort created by Python's numpy module. This analogy, however, is imperfect as Collette (2013, Chap. 3) explains. It is true that many of the standard numpy functions will work on the objects read from HDF5 files. For example, you can show the **`data`** object's dimensions or print out the maximum and minimum values using numpy functions:

```
data.shape
numpy.amin( data )
numpy.amax( data )
numpy.median( data )
numpy.mean( data )
```

However, if you type just the object name on the interactive Python prompt, you will see that it is really an HDF object, not a numpy array...

```
<HDF5 dataset "surfacePrecipitation": shape (229, 221), type "<f4">
```

...or if you type "type(data)" it's type will be <class 'h5py.highlevel.Dataset'>. If you are doing simple operations, you usually don't need to worry about this numpy-array vs. HDF5-dataset distinction. If you subset an HDF5 dataset, you obtain a pure numpy array, but you might not notice the change. A subset operation of taking every tenth element of an HDF5 dataset will result in an object with the type of a numpy array. In other words, the Python command "type(data[::10, ::10])" returns <type 'numpy.ndarray'>.

To display a swath on a map, one needs to know the latitude and longitude values as well as the data values. For this reason, the gpm.readSwath( ) function of the gpm.py Python program stores these three variables in a dictionary. If one had read in data, lat, and lon, one could create such a dictionary oneself by typing the following Python command:

```
swath = { 'lat': lat, 'lon': lon, 'data': data }
```

You would access one of the element of the **`swath`** dictionary, such as the **`data`** element, with the following syntax:

```
swath['data']
```

Another common operation that researchers perform on data read from GPM HDF5 files is to see what fraction of a variable exceeds a particular threshold. For example, you might wonder what fraction of observations were determined to have zero rain rates and what fraction of observations had heavy rain of ≥10 millimeters an hour. The following two Python statements calculate such fractions.

```
print 'fraction of elements equal to 0 ' \
  1.0* numpy.sum( numpy.equal( data, 0 ) ) / numpy.size(data)

print 'fraction of elements greater than or equal to 10 ' \
  1.0* numpy.sum( numpy.greater_equal( data, 10 ) ) / numpy.size(data)
```

References

Collette, A., 2013: *Python and HDF5*. O'Reilly Media, Inc. [Available in hardcopy or through http://safaribooksonline.com]

Gilpin, A., 7 July 2013: Setting up Python and easy_install on Windows 7, blog post on the ADEsquared blog. [Accessed online June 2015 at https://adesquared.wordpress.com]

Hou, A. Y., R. K. Kakar, S. Neeck, A. A. Azarbarzin, C. D. Kummerow, M. Kojima, R. Oki, K. Nakamura, and T. Iguchi, 2014: The Global Precipitation Measuring Mission. *Bulletin American Meteorological Society*, May 2014, 701−722.

Huffman, G. J., Bolvin, D. T., D. Braithwaite, K. Hsu, R. Joyce, and P. Xie, 2014: NASA Global Precipitation Measurement (GPM) Integrated Multi-satellitE Retrievals for GPM (IMERG), Algorithm Theoretical Basis Document (ATBD). https://storm-pps.gsfc.nasa.gov/storm/IMERG_ATBD_V4.pdf .

Lutz, M., 2013: *Learning Python*, 5th edition. O'Reilly Media, Inc. [Available in hardcopy or through http://safaribooksonline.com]

PPS, 2015: "GPM File Specification", NASA. [Available online at ftp://gpmweb2.pps.eosdis.nasa.gov/pub/GPMfilespec/filespec.GPM.V1.pdf]

Related Websites

http://matplotlib.org

http://pmm.nasa.gov

http://pps.gsfc.nasa.gov

http://storm.pps.eosdis.nasa.gov

http://www.h5py.org

http://www.numpy.org

http://www.python.org