RAPIDpy Documentation

Release 2.6.0

Alan D. Snow

May 09, 2019

Contents

RAPIDpy is a python interface for RAPID that assists to prepare inputs, runs the RAPID program, and provides post-processing utilities. More information about installation and the input parameters for RAPID can be found at http://rapid-hub.org. The source code for RAPID is located at https://github.com/c-h-david/rapid.

Contents:

CHAPTER 1

Installation

1.1 Step 1: Install RAPID

1.1.1 Before Installation Steps:

Ubuntu:

```
$ sudo apt-get install gfortran g++
```

RedHat/CentOS:

```
$ sudo yum install gcc-c++ gcc-gfortran
```

Windows with Cygwin:

Downloaded Cygwin (64-bit) (https://www.cygwin.com/) with these dependencies:

- gcc-core
- gcc-fortran
- gcc-g++
- gdb
- git
- make
- time
- wget

1.1.2 Installation Steps:

Manual:

- See: http://rapid-hub.org

Bash:

1. Clone RAPID repository:

```
$ git clone https://github.com/c-h-david/rapid.git
```

2. Install Prereqs:

```
$ cd rapid
$ chmod u+x rapid_install_prereqs.sh
$ ./rapid_install_prereqs.sh
```

3. Append *source rapid_specify_varpath.sh* to the ~/.bashrc or ~/.bash_profile:

```
source /path/to/cloned/rapid/rapid_specify_varpath.sh
```

4. Restart Terminal
5. Build RAPID:

```
$ cd rapid/src
$ make rapid
```

1.2 Step 2: Install RAPIDpy

Due to the dependencies required, we recommend using Anaconda or Miniconda. They can be downloaded from https://www.continuum.io/downloads or from https://conda.io/miniconda.html.

After installing Anaconda or Miniconda:

```
$ conda install -c conda-forge rapidpy
```

1.2.1 Developer Installation

This is how you get the most up-to-date version of the code.

See: https://github.com/erdc-cm/RAPIDpy/blob/master/.travis.yml for a more detailed list of installation steps.

Note: If you don't have git, you can download the code from https://github.com/erdc-cm/RAPIDpy

```
$ git clone https://github.com/erdc-cm/RAPIDpy.git
$ cd RAPIDpy
$ python setup.py install
```

To develop on the latest version:

```
$ git clone https://github.com/erdc-cm/RAPIDpy.git
$ cd RAPIDpy
$ python setup.py develop
```

CHAPTER 2

Generating Stream Network

2.1 Using ArcHydro to Generate Stream Network

See:

- https://github.com/Esri/python-toolbox-for-rapid
- https://github.com/erdc-cm/python-toolbox-for-rapid

2.2 Using TauDEM to Generate Stream Network

For more information about taudem, see: http://hydrology.usu.edu/taudem/taudem5/index.html

2.3 Installation

2.3.1 Step 1: Install TauDEM

```
$ cd /path/to/scripts
$ git clone https://github.com/dtarb/TauDEM.git
$ cd TauDEM/src
$ make
```

2.3.2 Step 2: Install RAPIDpy with GIS Dependencies

See: *Installation*

2.4 How To Use

2.4.1 Initialize TauDEM Manager

2.4.2 Generate network from DEM

2.4.3 Add Length in meters attribute

2.4.4 Extract Sub Network

STEP 1: Extract sub network from stream network

There are two options to do this.

1. Choose your own outlet point: `extractSubNetwork()`
2. Or let the code find the larges network: `extractLargestSubNetwork()`.

STEP 2: Extract sub network from catchments

CHAPTER 3

RAPID GIS Tools

These tools generate the RAPID input files and weight table files from the GIS stream networks.

Note: To generate your own network from a DEM see *Generating Stream Network*

Note: For these tools to work, you need GIS dependencies installed (See *Installation*).

3.1 Workflows

3.1.1 Static RAPID Files

3.1.2 Weight Table Files

3.1.3 Static RAPID Files and Weight Table Files

3.2 Individual Tools

3.2.1 Static RAPID Files

3.2.2 Weight Tables

3.3 Utilities

CHAPTER 4

Running RAPID

4.1 Tutorial

4.1.1 Step 1: Initialize the RAPID manager class.

- First, add the path to you rapid executable location.
- Next, you need to either tell it how many processors to use using the *num_processors* input variable or to use all available processors set *use_all_processors* to true.
- After that, add any other parameters you would like to use that would normally be in the rapid namelist file (this is case sensitive).

4.1.2 Step 2 (optional): Add/update additional namelist parameters later

4.1.3 Step 3 (optional): Update reach number data

4.1.4 Step 4 (optional): Update simulation runtime data

4.1.5 Step 5: Run RAPID

4.1.6 Step 6 (optional): Convert RAPID output to be CF Compliant

4.2 Full API Description

CHAPTER 5

Inflow from Land Surface Models

Code to use to prepare input data for RAPID from Land Surface Models (LSM) such as:

- ECMWF's ERA Interim Data
- NASA's GLDAS/NLDAS/LIS Data
- CMIP5 Data (daily VIC data available from 1950 to 2099)
- Joules
- WRF
- FLDAS (http://disc.sci.gsfc.nasa.gov/uui/datasets?keywords=FLDAS)
- ERA 20CM

5.1 Step 1: Retrieve Land Surface Model Runoff Output

Download the data into a local directory.

- http://apps.ecmwf.int/datasets
- http://ldas.gsfc.nasa.gov/index.php
- ftp://gdo-dcp.ucllnl.org/pub/dcp/archive/cmip5/hydro/BCSD_daily_VIC_nc/

5.2 Step 2: Create folders for RAPID input and output

In this instance:

```
$ cd $HOME
$ mkdir -p rapid-io/input rapid-io/output
```

5.3 Step 3: Create script using LSM process

Here is the API for the function to use. Follow the example to create a script to use the code such as in ~/run_lsm.py.

5.4 Step 4: Add RAPID files to the rapid-io/input directory

Make sure the directory is in the format [watershed name]-[subbasin name] with lowercase letters, numbers, and underscores only. No spaces!

Example:

```
$ ls /rapid-io/input
nfie_texas_gulf_region-huc_2_12

$ ls /rapid-io/input/nfie_texas_gulf_region-huc_2_12
comid_lat_lon_z.csv
k.csv
rapid_connect.csv
riv_bas_id.csv
weight_era_t511.csv
weight_nldas.csv
weight_gldas.csv
weight_lis.csv
weight_wrf.csv
weight_cmip5.csv
x.csv
```

If you have not generated these files yet, see *RAPID GIS Tools*

5.5 Step 5: Run the code

```
$ python ~/run_lsm.py
```

CHAPTER 6

RAPIDDataset

This is a wrapper for the RAPID Qout netCDF file. Here are some basic examples for useage.

CHAPTER 7

Postprocessing

7.1 Merge RAPID Output

7.2 Generate qinit from past qout

RAPIDpy also creates a qinit file from a RAPID qout file. This example shows how.

7.3 Generate seasonal qinit from past qout

7.4 Goodness of Fit

To check how well your simulation performed versus observations, these functions can help you.

CHAPTER 8

RAPID to GSSHA

It is possible to use RAPID streamflow as an overland flow boundary condition to the Gridded Surface Subsurface Hydrologic Analysis (GSSHA) model.

8.1 What is GSSHA?

GSSHA is a physically-based, distributed hydrologic model. GSSHA is developed and maintained by Coastal and Hydraulics Laboratory (CHL) which is a member of the Engineer Research & Development Center of the United States Army Corps of Engineers (USACE).

Note: For more information about GSSHA please visit the the gsshawiki .

8.2 Tutorial

There are two ways to input RAPID as a boundary condition for GSSHA. One is to connect the GSSHA stream network link and node to the RAPID river ID and generate the IHG file. The other is to generate an XYS timeseries file and add it to the netork using WMS.

8.2.1 Method 1: Generate IHG File

Step 1.1: Look at Stream Network in WMS to find Link & Node

1. Open GSSHA project in WMS
    - Switch to 2-D Grid Module
    - In the top menu: *GSSHA -> Open Project File*
2. Turn on *Stream Link Numbers* Display

- In the top menu: *Display -> Display Options*
- Select *Map Data* in top left box
- In the center box, make sure *Stream Link Numbers* is checked under the Arcs subsection.

3. Determine the Link ID by looking on model.

Step 1.2: Connect RAPID river ID to GSSHA Link & Node and Generate IHG

8.2.2 Method 2: Generate XYS File

Step 2.1: Generate XYS File

Step 2.2: Add XYS File in WMS

In the Distributed Hydrology section, go to *Overland Flow Boundary Conditions in GSSHA (Tutorial 55)* at http://www.aquaveo.com/software/wms-learning-tutorials

Here is a direct link to the document: http://wmstutorials-10.1.aquaveo.com/55%20Gssha-Applications-OverlandBoundaryConditions.pdf

CHAPTER 9

Credit

- The development of RAPIDpy was funded by ERDC.
- The pre-processing GIS tools and the inflow tools in RAPIDpy are based on the ESRI RAPID Toolbox (https://github.com/Esri/python-toolbox-for-rapid).

CHAPTER 10

How the Inflow and GIS tools work:

Snow, Alan D., Scott D. Christensen, Nathan R. Swain, E. James Nelson, Daniel P. Ames, Norman L. Jones, Deng Ding, Nawajish S. Noman, Cedric H. David, Florian Pappenberger, and Ervin Zsoter, 2016. A High-Resolution National-Scale Hydrologic Forecast System from a Global Ensemble Land Surface Model. *Journal of the American Water Resources Association (JAWRA)* 1-15, DOI: 10.1111/1752-1688.12434 https://onlinelibrary.wiley.com/doi/full/10.1111/1752-1688.12434

Snow, Alan Dee, "A New Global Forecasting Model to Produce High-Resolution Stream Forecasts" (2015). All Theses and Dissertations. Paper 5272. http://scholarsarchive.byu.edu/etd/5272

CHAPTER 11

Publications using RAPIDpy and RAPID:

Tavakoly, A. A., A. D. Snow, C. H. David, M. L. Follum, D. R. Maidment, and Z.-L. Yang, (2016) "Continental-Scale River Flow Modeling of the Mississippi River Basin Using High-Resolution NHDPlus Dataset", Journal of the American Water Resources Association (JAWRA) 1-22. DOI: 10.1111/1752-1688.12456

CHAPTER 12

Datasets produced using RAPIDpy and RAPID:

Ahmad A Tavakoly. (2017). RAPID input files corresponding to the Mississippi River Basin using the NHDPlus v2 Dataset [Data set]. Zenodo. http://doi.org/10.5281/zenodo.322886

CHAPTER 13

Other tools to prepare input for RAPID

- For ESRI users: https://github.com/Esri/python-toolbox-for-rapid
- Modified version of the ESRI RAPID Toolbox: https://github.com/erdc-cm/python-toolbox-for-rapid
- For the NHDPlus dataset: https://github.com/c-h-david/RRR

CHAPTER 14

Indices and tables

- genindex
- modindex
- search