HOW TO INSTALL THE TOPOFLOW 3.5 PYTHON PACKAGE AND DEPENDENCIES (FEB. 12, 2017)

SCOTT D. PECKHAM

Step 1. Install Python 2.7 and commonly-used Python packages. One of the easiest ways to do this is to install *Anaconda*, a complete, open-source Python platform. Anaconda supports MacOS, Linux and Windows. You can download the installers from: *https://www.continuum.io/downloads*. The installation includes over 100 Python packages that support scientific work. This includes all but one of the packages needed by TopoFlow, including: *NumPy, SciPy, setuptools, pip and h5py*. It also includes *Matplotlib, Jupyter, Pandas, curl, wheel* and many others. Anaconda also includes a package and dependency manager called *conda*, which makes it easy to install any of 620 other Python packages (e.g. *netCDF4*).

Step 2. Install the netCDF4 module. TopoFlow uses this module to write model output to standardized netCDF files. The netCDF4 module relies on the *h5py* package that is included with Anaconda.

```
$ conda install netCDF4
```

You can check whether the package was installed correctly by typing *python* in a terminal window (to start a Python session) and then typing *import netCDF4* at the Python prompt.

Step 3. Download the TopoFlow Python package (v. 3.5) from GitHub.
Download it from: *https://github.com/peckhams/topoflow* as a zip file and unzip it.

Step 4. Install the TopoFlow Python package. There are many advantages to installing TopoFlow as a Python package, but it is also helpful to retain the option of making changes to the TopoFlow source code without rebuilding the package. It is therefore recommended to install TopoFlow as an "editable install". This is done by copying the entire TopoFlow package folder (TopoFlow_3.5) someplace convenient (e.g. your home directory or Dropbox folder). This folder contains a file called *setup.py* used for installation. Then, in a terminal window, type the commands

```
$ cd TopoFlow_3.5
$ pip install --editable ./
```

A folder with extension *.egg-info* will be created in the TopoFlow_3.5 folder that allows it to be recognized as a Python package. (Note: A similar but slightly different method is to use the command: *python setup.py develop* instead of *python setup.py install* .)

Step 5. Perform a test run of TopoFlow with the default data set *Treynor*. TopoFlow allows you to specify different directories for model input and output files in the CFG files. The input files for the Treynor data set are in the *examples/Treynor_Iowa_30m* folder of the TopoFlow package. However, output files are written to a directory in your home directory called *TF_Output/Treynor*. So first, create this directory with the commands

```
$ cd;  mkdir TF_Output;  mkdir TF_Output/Treynor
```

Next, open a terminal window and type:

```
$ python -m topoflow
```

You can edit the EMELI *provider file* (with extension *providers.txt*) to specify different components to use for the various hydrologic processes. Each process component is configured with its own *configuration file*, or CFG file, which are text files with extension *.cfg*.

Step 6. Run TopoFlow with your own data sets. Acquire a DEM for your study site and create necessary input files as explained in Appendix B and C. You need a CFG file for every component you want to use in the CFG directory. You also need an *outlet file* (extension *outlets.txt* and an EMELI *provider file* (extension *providers.txt*). You should start with copies from the Treynor example and edit them as needed, making sure their *comp_status* has been set to *Enabled*. All of these files have filenames that begin with the *cfg_prefix*, which is typically the *case_prefix* associated with a particular modeling scenario. To run TopoFlow for your own data set, open a terminal window and type:

```
$ python -m topoflow --cfg_prefix PREFIX --cfg_directory DIRECTORY
         --driver_comp_name DRIVER
```

With your own data set, you may need to use smaller time steps in the CFG files to achieve a numerically stable model run (i.e. that doesn't crash). Also, you should use the same time step in the CFG files for the meteorology and infiltration components. Be aware that grid stack files can be large (i.e. those ending with *.rts* or *2D-*.nc*) and can accumulate over multiple model runs.