WRF with Lightning Assimilation

7 February 2017

Please direct any issues/questions to Nick Heath:
Heath.Nicholas@epa.gov
or
nkh09@my.fsu.edu

Overview

This document describes how to run WRF with the lightning assimilation technique described in Heath et al. (2016). The assimilation method uses gridded lightning data to trigger and suppress sub-grid deep convection in Kain-Fritsch. The gridded lightning data (variable name is 'LNT') is read in through auxinput8. The lightning data is grouped into 30-min intervals and treated as simple zeros (no lightning) or ones (lightning) for the assimilation method. All of the necessary code modifications and data are described below.

NOTE: This current assimilation code works in WRF version 3.8. We are currently trying to get the code into the official WRF release, but until that happens each new version of the Kain-Fritsch code will need to be adapted manually to include the assimilation.

The Code

The latest version of the code is contained in:

WRF_LTGDA_7Feb2017.tar.gz

This file contains several files for the assimilation:

Registry.EM
Registry
Registry.EM_COMMON
module_cumulus_driver.F
module_cu_kfeta.F
module_first_rk_step_part1.F
ltng_regrid_WRF.py
namelist.adds

The Registry and FORTRAN files contain all the updates needed to run WRF with lightning assimilation. Within the Registry and FORTRAN files, search for '!LTNG' to find all of the modifications made for the lightning assimilation. There is also a generic Python script ('ltng_regrid_WRF.py') that can be modified to grid lightning data to your WRF domain using the wrfinput_d01 file for your simulation. The 'namelist.adds' file shows the additional WRF namelist options that need to be added to run with assimilation.

Compiling WRF with Lightning Assimilation

Copy module_cu_kfeta.F and module_cumulus_driver.F into WRFV3/phys

Copy module_first_rk_step_part1.F into WRFV3/dyn_em

Copy all of the Registry files into WRFV3/Registry

If this is a fresh version of WRF, then you can compile using the standard procedure. If you are adding this to an existing WRF setup, you will need to issue a "clean –a" because of the updated Registry files. Then, re-configure and re-compile as usual.

Preparing Lightning Data for WRF

After compiling, the next step is to grid your lightning data into your WRF domain. This can be done using the 'ltng_regrid_WRF.py' script. Note that this is rather slow at the moment, so feel free to use your own method of gridding. If you do use this script, it expects the lightning data in the following format:

YYYY-MM-DD HH:MM:SS.sss LAT LON

Here is an example of a few lines of data:
2011-07-01 00:00:00.168 42.435 -102.795
2011-07-01 00:00:00.811 32.267 -94.215
2011-07-01 00:00:01.169 22.751 -81.083

If you choose to use the regrid script provided, further instructions are provided within the code. If you grid the data yourself, just make sure that the lightning data variable is 'LNT' and that the data are in 30-min intervals with 0 = no lightning and 1 = lightning. Further, the lightning assimilation expects the data to be held in the following format:

The 00:00 UTC time would hold data from 00:00 – 00:30 UTC
The 00:30 UTC time would hold data from 00:30 – 01:00 UTC
etc.

Lastly, be sure to propagate whatever naming convention you use for your lightning data file into the WRF namelist (described below).

Running WRF

Now you are ready to run WRF. Move the lightning data file into the directory you will be running WRF from. Add the following namelist options (which are also in the 'namelist.adds' file):

&time_control

```
io_form_auxinput8           = 2,
auxinput8_inname            = 'NLDN_LTNG_<year>_<month>.nc',
frames_per_auxinput8        = 1600,
auxinput8_interval_m        = 30,
auxinput8_end_h             = 9999,
```

The user has flexibility with the auxinput8_inname, as mentioned previously. If you want WRF to search for the correct file based on the current time, you can include the date in the file name, e.g., NLDN_LTNG_<year>_<month>.nc, or NLDN_LTNG_<year>_<month>_<day>.nc, etc. If you only have one file you are using, you can explicitly state the input file name: auxinput8_inname = 'NLDN_LTNG.nc'

The 'auxinput8_interval_m = 30' should not be changed. The other timing options, 'frames_per_auxinput8' and 'auxinput8_end_h', need to be modified based on your particular case. For example, if we were running all of July 2011, the file created would have 1488 time steps. So, we would change 'frames_per_auxinput8 = 1488'. The end hour can be set to an arbitrarily large number to ensure the assimilation runs continuously throughout the entire run. If you are only using assimilation for a portion of the run, e.g., during the spin-up of a real-time forecast, the end hour will need to be modified accordingly.

In the &physics portion of the namelist, make the following changes and additions:

```
&physics
 cu_physics                 = 1,
 cudt                       = 10,
 kfeta_trigger              = 1,
 ltg_assim                  = .true.,
 suppress_opt               = 2,
```

The suppress_opt = 2 option uses the "ShallOnly" technique described in Heath et al. (2016), which is the recommended approach for most applications. There are two other options, 0 (no suppression) and 1 (full suppression), which are described in Heath et al. (2016).

NOTE: Because the technique was developed for NLDN data, and they only cover the CONUS, the assimilation automatically switches to suppress_opt = 0 in grid cells outside of the NLDN range (-130, -60 W), (20, 53 N). This is hard-coded into the Kain-Fritsch scheme. If you need to modify this, search for 'nldn_llon' within 'module_cu_kfeta.F' and make the necessary changes to the NLDN bounds parameters.

Once these options are set, you are now ready to run WRF with assimilation.

References

Heath, N. K., J. E. Pleim, R. C. Gilliam, and D. Kang (2016), A simple lightning assimilation technique for improving retrospective WRF simulations, *J. Adv. Model. Earth Syst.*, *8*, 1806 – 1824, doi:*10.1002/2016MS000735*.