Welcome to the WRF-Fire Tutorial Workshop!

In this tutorial series, we will cover the basics of preparing and running coupled fire-atmosphere simulations with WRF-Fire. This tutorial includes (1) recorded video lectures on different topics, (2) reference input and output files for several idealized fire simulations and a real fire simulation, and (3) multiple office hours and a Slack channel to answer your questions. In the video series, we will cover (1) the basic options available within WRF-Fire as well as how to set up a model simulation, (2) how to download and process the required input data for real fire simulations, and (3) how to start the simulation from an observed fire perimeter.

As the focus of this workshop is on fire simulation with WRF-Fire, we will not cover the basics of the WRF numerical weather prediction model. Ideally, you should have preliminary working knowledge of WRF to be able to readily follow this tutorial series. Additionally, we will provide pre-compiled WRF-Fire executables to be used on the Derecho HPC system for this tutorial series (described below). **We will not cover in detail the compilation process as it is highly system dependent and out of the scope of this tutorial.** However, below are the general steps to compile WRF, in addition to other high-level information that is important for you to read.

Compiling WRF

When we refer to "WRF-Fire", we are talking about a specific physics component of WRF. Therefore, when one compiles the WRF model, the modules specific to WRF-Fire are included by default. In general, compiling WRF consists of three main steps:

1. Install the required compilers
    - The required compilers are Fortran and C compilers, such as GCC and GFORTRAN, or Intel compilers
2. Compile the required libraries
    - The required libraries are:
        1. NetCDF C and NetCDF Fortran
        2. A MPI library (e.g., OpenMPI): this library is only required if you would like to run simulations using parallel processing (multiple CPUs)
        3. Jasper
        4. Libpng
        5. Zlib
3. First compile WRF and then compile WPS
    - Three main commands are involved in compiling WRF and WPS:
        1. *./clean -a*: removes any old compilation files – needs to be run prior to each compilation
        2. *./configure*: configure the compilation setup based on the environment in which you are compiling WRF and WPS
        3. *./compile*: compile WRF and WPS. For WRF, you need to identify which executables you wish to compile. For example *./compile em_real* will compile WRF-Fire for a real-case fire simulation (and therefore populate

the */test/em_real* folder with necessary ancillary files and executables), and *./compile em_fire* will compile WRF-Fire for idealized fire simulations (and therefore populate the */test/em_fire* folder with necessary ancillary files and executables).

Some guides are available that provide a tutorial for compiling WRF:

- WRF Official Compilation Guide: https://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php
- WRF-Fire Wiki Website Developed by Us: https://unr-wrf-fire.readthedocs.io/en/latest/DCR.html

Note: for an HPC environment, one should use the libraries and compilers available on the target HPC based on its documentation. For instance, on the NSF NCAR-Wyoming Supercomputing Center's Derecho system, WRF can be compiled by loading the following modules, using "module load (name of the module)":

1. ncarenv
2. gcc (or Intel compilers)
3. ncarcompilers
4. cray-mpich
5. netcdf

Note: for running models, the job management of the target HPC should be considered.

To help you get started, we have created an example environment for the Derecho system. These files can be found in the GitHub repository here and here, for WPS and WRF, respectively. To set your environmental modules, simply type: "*source <name of file>*".

Note: the compilation environments for WPS and WRF are the same, we simply provide a file in each directory for convenience.

Running WPS and WRF executables

To simplify the process for those who are less familiar with configuring and compiling WPS/WRF, we provide pre-compiled executables for WPS and WRF in the GitHub repository. These executables were created from the above environment on the Derecho system, using configure option 34 for WRF and configure option 3 for WPS.

Example portable batch system (PBS) job submission scripts, specific to running the WPS and WRF executables on Derecho, are also available on the GitHub repository. On Derecho, the job script may be submitted to the queue via: "*qsub <name of submission script>*". **Before you submit your PBS job script, please double check the directives.** More details about PBS may be found here.

Real-world simulation

The real case that we are using for the WRF-Fire Tutorial is the C34 Fire (also called the C-34 Fire or the CR34 Fire) from February 2019. This was a rather small, wind-driven fire that burned in short grass and brush fuels in southeastern Colorado, U.S. Some details about the event may be found here.

Reference datasets

One of the main goals of this tutorial is for YOU (the participant) to learn and become familiar with processing LANDFIRE fuel and topographic datasets and running WRF-Fire. Along the way, you are likely to run into snags and unexpected issues. Don't worry! To help you understand whether your results are reasonable, we provide a few reference datasets.

Note: we don't necessarily expect identical results between participants due to differences in compilation environments, namelist options, etc.

The reference datasets, which are specific to the domain covering the C34 Fire, can be found in two places:

1. If you have access to Derecho, then they can be accessed directly at the head directory: "*/glade/work/tjuliano/fire/unr/WPS_GEOG/*", with subdirectories named:

    sb40_c34_2014: Scott and Burgan 40 fuel categories from 2014 LANDFIRE
    sb40_c34_2016: Scott and Burgan 40 fuel categories from 2016 LANDFIRE
    sb40_c34_2020: Scott and Burgan 40 fuel categories from 2020 LANDFIRE
    topo_30m_c34: High-resolution (30 m grid spacing) SRTM topography

2. In the GitHub repository, under: "*c34/fuel_and_topo*"