Interfacing WRF

1) Check out the OOPS code:

git clone https://github.com/UCAR/oops.git OOPS_wrf

2) Go the newly created directory:

cd OOPS_wrf

3) Set up to track the develop head:

git checkout --track origin/develop

4) Move to the WRF branch:

git checkout feature/wrf

5) Compile with the provided script:

./build_oops_cheyenne.csh

6) The script build_oops_cheyenne.csh always recompiles from scratch, to restart a compilation:

Set-up the environment:

unsetenv LD_LIBRARY_PATH

module purge

module load gnu cmake/3.7.2 netcdf

setenv BOOST_ROOT /glade/p/ral/nsap/jcsda/code/boost_1_64_0

setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${BOOST_ROOT}/stage/lib"

Go to the build directory and use the make command:

cd build/oops/wrf/test; make

7) To run the test in serial mode, set-up first the run-time environment (if not already done in 6):

unsetenv LD_LIBRARY_PATH

module purge

module load gnu cmake/3.7.2 netcdf

setenv BOOST_ROOT /glade/p/ral/nsap/jcsda/code/boost_1_64_0

setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${BOOST_ROOT}/stage/lib"

8) Run the dirac test:

cd build/oops/wrf/test; ../../../bin/wrf_dirac.x testinput/dirac.nicas.json;

9) Look for files:

dirac_d01_2017-07-28_06:00:00 Input perturbation

bens_d01_2017-07-28_06:00:00 Response to the perturbation

10) Plot the results (B*X') of the test with ncview:

module load ncview;

ncview ./bens_d01_2017-07-28_06:00:00

11) Select temperature U, the plot should look like this one below:

blocked URL

12) To run the test in MPI mode, edit script run_oops_cheyenne.ksh and select the wrf model:

model=wrf

13) Execute the script:

```
./run_oops_cheyenne.ksh
```

Notes:

1) The input data consist of a 16 member time lagged ensemble and can be found at:

```
/glade/p/ral/nsap/jcsda/data/wrf/data/2017072800
```

2) The MPI version has not been yet fully implemented.