WRF-NMM: Initialization with Real Data

Matthew Pyle

8 August 2006

Overview

- Where does “real” fit into the WRF-NMM system?
- Brief code description
- Running “real”: The namelist, input and output files

Where does “real” fit?

- The SI by itself **DOES NOT** generate the files needed to run the WRF model.
- The “real” program reads the SI output, and creates WRF dynamical core specific variables that are output to an initial condition (*wrfinput_d01*) and lateral boundary condition (*wrfbdy_d01*) file.
- Built alongside the WRF forecast model, it utilizes the WRF software infrastructure.

Code description

- The unique “real” functionality is largely contained in ./main/real_nmm.F and ./dyn_nmm/module_initialize_real.F.

- However, the program utilizes code shared with the forecast model for common tasks such as reading and writing data.

Code description (cont.)

- In broad terms, the code (main program and subroutines) within real_nmm.F:
  - Does time looping and makes the calls that read in the SI output files.
  - Calls init_domain (next slide)
  - Calls start_domain, which initializes variables, allocates arrays, and performs other assorted tasks.
  - Computes boundary tendencies.
  - Makes calls to write the output *wrfinput_d01* and *wrfbdy_d01* files.

Code description (cont.)

- The module_initialize_real code (init_domain):
  - Interpolates input soil data in the vertical, and generates the appropriate surface fields for the selected physics options.
  - Strives to maintain consistency between various land surface characteristics (land/sea mask, sea ice, veg/soil classes, skin temperature, albedo, …)
  - Generates a variety of fields used by the model dynamics – many depend on the model timestep, and thus must be defined within real.

Namelist items

- The same *namelist.input* file used to run the WRF model is also used when running “real”. It is NOT the same as the namelist used with the SI code.

- Will attempt to limit the namelist discussion here to items directly relevant to “real”.

&time_control

```
run_days                            = 2,
run_hours                           = 0,
run_minutes                         = 0,
run_seconds                         = 0,
```

Specifies the length of the model forecast. Alternately, these “run_” specifications can be set to zero, allowing the “start_” and “end_” definitions (next slide) to control forecast length.

&time_control (cont.)

```
start_year                          = 2005,
start_month                         = 08,
start_day                           = 24,
start_hour                          = 12,
start_minute                = 00,
start_second                = 00,

end_year                            = 2005,
end_month                           = 08,
end_day                             = 26,
end_hour                            = 12,
end_minute                  = 00,
end_second                  = 00,

interval_seconds                    = 10800,
```

The “real” job will utilize SI output available between the start and end times when generating the lateral boundary condition (LBC) file.

interval_seconds: specifies frequency of LBC updates, and should match frequency at which SI output was produced.

&domains

```
time_step                  = 26,
time_step_fract_num        = 2,
time_step_fract_den        = 3,

max_dom                    = 1,
```

Defines model time step in seconds (including a fractional component). Here defined as 26 2/3 s.

WRF-NMM model time step is generally about 2.25 x (grid spacing in km), or about 330 x (angular grid spacing) and is selected to obtain an integer number of time steps per hour

max_dom should remain one (no nesting here yet).

&domains (cont.)

```
s_we                = 1,
e_we                = 361,
s_sn                = 1,
e_sn                = 576,
s_vert              = 1,
e_vert              = 61,


dx                  = .087603,
dy                  = .075047,
```

"**we**": west-east dimension

"**sn**": north-south dimension

"**vert**": vertical dimension

All start (s_) at 1, and the end (e_) dimensions for "we" and "sn" are ONE GREATER than their specification in the WRF-NMM SI namelist.

As in the SI namelist, the dx and dy specifications are in fractions of a degree. Be consistent between the SI and model namelists!

Running real_nmm.exe

Assuming that:

- The WRF source code has been successfully downloaded and compiled.

- WRF-NMM SI output exists for the domain and forecast period of interest.

- The WRF namelist file has been properly edited.

Running real_nmm.exe (cont.)

- Combine the contents of ./WRFV2/test/nmm_real/ or ./WRFV2/run/ (including *real_nmm.exe* and the *namelist.input* file) and the WRF-NMM SI output files (*wrf_real_input*…) into a single run directory.
- *real_nmm.exe* can be run serially or using distributed memory parallelism (choice determined by how the source code is compiled).
- "real" is memory intensive, particularly for large domains. Running on multiple CPUs may provide access to more memory, which enables runs over larger-dimension domains.

Running real_nmm.exe (cont.)

- About how much memory is used? A non-representative sample of domains:
  - 360 x 575 x 60 (~ 4.8 GB on single proc, ~510 MB/proc when distributed over 12 processors)
  - 530 x 851 x 35 (~ 6.4 GB)
  - 110 x 181 x 35 (~ 314 MB)

Running real_nmm.exe (cont.)

- Run as a parallel job:
  - mpirun –np # *real_nmm.exe*
  - poe *real_nmm.exe*
  - Log file(s) in rsl.out.* and rsl.error.* (one pair of rsl.* files for each procesor)
- Run serially:
  - ./*real_nmm.exe* > real.log 2>&1 (sh/ksh)
  - ./*real_nmm.exe* >& real.log (csh)

Running real_nmm.exe (cont.)

- If all goes well, will end up with non-zero length *wrfinput_d01* and *wrfbdy_d01* output files.
- Can examine these files (assuming they are netCDF) with ncdump, ncBrowse, or a similar netCDF tool.
- If *real_nmm.exe* runs to completion, the line "real_nmm: SUCCESS COMPLETE REAL_NMM INIT" will be near the end of the log file.

The WPS future:

- The “real” job isn’t immune from the preprocessing changes coming due to the advent of the so-called WPS.

- The vertical interpolation functionality will become part of “real”; this change is the primary piece of work that needs to be done for the WRF-NMM to become ready for the WPS.

Acknowledgements

- Some ideas for this talk came from Dave Gill's ARW Tutorial slides.

- Dave also provided guidance in the initial development/port of a "real" capability into the WRF-NMM.