WRF Code Submissions— Coding Standards and Testing Requirements

Coding Standards

Code written for inclusion in WRF must following these rules and coding practices. Some pertain specifically to physics routines.

– Use one self–contained module
– Standard Fortran 90 free format (!comments, &continuation lines)
– Use IMPLICIT NONE (declare all variables)
– Use INTENT IN/OUT/INOUT at least at top level
– No prints (call *wrf_message*)
– No read/writes in run–time code (use WRF I/O streams)
– No stops (call *wrf_error_fatal*)
– No EQUIVALENCE
– No COMMON blocks
– Use top of module for module-wide parameters, data/table constant arrays
– Pass variable arrays via argument lists
– Include an init routine to do start-time initialization of data, etc.
– Init routine to be called from relevant routine of physics_init (CASE SELECT)
– Reads of constant data/table arrays can be done in init code (bcast needed)
– Main run-time routine to be called from the relevant driver (CASE SELECT)
– Distinguish indices:
  – – its, ite, etc: Loop limits and local array dimensions
  – – ims, ime, etc: Dimensions of WRF (registry) arrays passed in
  – – ids, ide, etc: Only for domain-boundary tests

Required Testing

Code offered to the WRF repository must be tested by the submitting developer. The purposes include improving the level of code submitted to the repository and making the integration of code into WRF more efficient. The testing is to check that the software behaves as desired, is robust, and yields reasonable results. It is also to have the developers themselves address issues or bug fixes upfront, in order to reduce problem code and to avoid time loss of the Developers Committee, Release Committee, and others involved in the process. The testing areas are: (a) software testing, (b) robustness testing, and (c) reasonableness testing. These are summarized below. The testing information and the testing materials will be provided at http://www.wrf-model.org/users/testing.php .

(a) Software Testing

Before code is accepted, developers must run their codes through the WRF Software Test Suite (WSTS) in order to ensure that the new code compiles, runs, and satisfies bit-for-bit comparison results. The WSTS has basic tests as well as tests of advanced features in WRF.

These are short tests that don't have large memory or time requirements. The areas to be checked through use of the WSTS are: (i) compilation, (ii) parallel operation (iii) nesting, (iv) compatibility with other physics parameterization schemes, and (v) restarts.

Regarding the ARW and NMM solvers, code or modifications can be written for either one of the solvers. While development may be targeted for one specific solver (core), the testing must confirm that compilation and operation with the other solver are not adversely affected.

(b) Case Testing— Robustness and Reasonableness Testing

Case simulations are to be done to assess the code's robustness and reasonableness. Modestly-sized domains and a relatively short forecast length are employed for these tests. The simulations should employ parallel processing. Data for initializing the cases is provided at the code testing web site listed above.

The robustness testing requires doing runs on five different days in each of two seasons, summer and winter. These runs are used to identify problems that running a single event or scenario might not reveal. The run lengths should be 24 hours. To pass these tests, the simulations should run to completion, and the output should be checked for a lack of NaN's ("not a number" designations) and a lack of fatal CFL violations.

The reasonableness testing requires running two cases and examining basic forecast output to ensure that the results are not unphysical. Both a summer and a winter case should be more extensively examined. The forecast lengths for both cases should be 24 hours. Note that these cases may also be counted towards the five total robustness runs.

Some standard option output and plots of certain forecast fields will be provided for developers to check their output against. The goal is to see general consistency of forecast results and to check if there are any gross discrepancies from a standard baseline simulation. NCL scripts are provided for the tester to make some basic plots, which can be compared with the provided plots.