Foreword

This User’s Guide describes the Advanced Research WRF (ARW) Version 3.8 modeling system, released in April 2016. As the ARW is developed further, this document will be continuously enhanced and updated. Please send feedback to wrfhelp@ucar.edu.

This document is complementary to the ARW Tech Note (http://www2.mmm.ucar.edu/wrf/users/docs/arw_v3.pdf), which describes the equations, numerics, boundary conditions, and nesting etc. in greater detail.

Highlights of updates to WRFV3.8 include:

- **WRF model:**
  - *Physics:*
    - New Kain-Fritsch cumulus potential scheme (contributed by PNNL);
    - A new option for aerosol-radiation interaction (aer_opt = 3) (contributed by G. Thompson and T. Eidhammer of RAL/NCAR);
    - Improvement to various physics options;
    - Updated cloud fraction scheme (icloud =3);
    - WRF-Hydro 4.0 (contributed by the WRF-Hydro group of RAL/NCAR)
  - *Others:*
    - A flux-adjusting surface data nudging option;
    - Improved vertical refinement (contributed K. Lundquist of LLNL);
    - 30-seconds GMTED2010 topography dataset.

- **WRF-DA updates:**
  - New observation type: AMSR-2 radiance (contributed by Xin Li, Jie Ming, Ming Xue, Yuan Wang, and Kun Zhao of Nanjing University, China)
  - New dynamic constraint for 3DVAR and Hybrid DA (contributed by Syed Rizvi of NCAR and Chun Yang of Nanjing University of Information Science & Technology, China)

- **WRF-Chemistry**
  - Dust scheme improvement (contributed by Martina Klose of Univ. of Cologne and New Mexico State Univ.).

For the latest version of this document, please visit the ARW Users’ Web site at http://www2.mmm.ucar.edu/wrf/users/.

---

**Contributors to this guide:**
Wei Wang, Cindy Bruyère, Michael Duda, Jimy Dudhia, Dave Gill, Michael Kavulich, Kelly Keene, Ming Chen, Hui-Chuan Lin, John Michalakes, Syed Rizvi, Xin Zhang, Judith Berner, Soyoung Ha and Kate Fossell

**Contributors to WRF-Fire chapter:**
Jonathan D. Beezley, Janice L. Coen, and Jan Mandel

**Contributors to UPP section:**
Hui-Ya Chuang, Nicole McKee, Tricia Slovacek, Jamie Wolff, and Kate Fossell
**Special Acknowledgment:**
We gratefully acknowledge the late Dr. Thomas T. Warner for his years of research, development, instruction, leadership, and motivation in the field of NWP modeling.
1. Overview
- Introduction ............................................................................................................. 1-1
- The WRF Modeling System Program Components ........................... 1-2

2. Software Installation
- Introduction ............................................................................................................. 2-1
- Required Compilers and Scripting Languages ................................................. 2-2
- Required/Optional Libraries to Download ......................................................... 2-3
- Post-Processing Utilities .................................................................................... 2-4
- UNIX Environment Settings .............................................................................. 2-4
- Building the WRF Code ................................................................................... 2-5
- Building the WPS Code ................................................................................... 2-6
- Building the WRFDA Code (for 3DVAR) ....................................................... 2-7
- Building the WRFDA Code (for 4DVAR) ....................................................... 2-9

3. The WRF Preprocessing System (WPS)
- Introduction ............................................................................................................. 3-1
- Function of Each WPS Program ........................................................................ 3-2
- Installing the WPS .............................................................................................. 3-4
- Running the WPS ............................................................................................... 3-8
- Creating Nested Domains with the WPS .......................................................... 3-20
- Selecting Between USGS and MODIS-based Land Use Data ........................................ 3-22
- Selecting Static Data for the Gravity Wave Drag Scheme .................................. 3-23
- Using Multiple Meteorological Data Sources .................................................. 3-23
- Using Non-isobaric Meteorological Datasets .................................................... 3-26
- Alternative Initialization of Lake SSTs ............................................................... 3-27
- Parallelism in the WPS ....................................................................................... 3-28
- Checking WPS Output ......................................................................................... 3-29
- WPS Utility Programs ....................................................................................... 3-30
- Writing Meteorological Data to the Intermediate Format ................................ 3-34
- Required Meteorological Fields for Running WRF ......................................... 3-36
- Creating and Editing Vtables ............................................................................. 3-37
- Writing Static Data to the Geogrid Binary Format ............................................ 3-39
- Creating an Urban Fraction Field from NLCD Data .......................................... 3-42
- Description of Namelist Variables .................................................................... 3-44
- Description of GEOGRID.TBL Options ............................................................. 3-49
- Description of index Options ............................................................................ 3-52
- Description of METGRID.TBL Options ............................................................. 3-55
- Available Interpolation Options in Geogrid and Metgrid ................................ 3-58
- Land Use and Soil Categories in the Static Data .............................................. 3-61
- WPS Output Fields ............................................................................................. 3-63
4. WRF Initialization
   - Introduction ........................................................................... 4-1
   - Initialization for Ideal Cases ............................................. 4-3
   - Initialization for Real Data Cases ..................................... 4-6

5. WRF Model
   - Introduction ........................................................................... 5-1
   - Installing WRF ...................................................................... 5-2
   - Running WRF ......................................................................... 5-8
   - Examples of namelists for various applications .................. 5-35
   - Check Output .......................................................................... 5-37
   - Trouble Shooting .................................................................... 5-38
   - Physics and Dynamics Options .......................................... 5-38
   - Summary of PBL Physics Options ....................................... 5-52
   - Summary of Microphysics Options ...................................... 5-54
   - Summary of Cumulus Parameterization Options .................. 5-56
   - Summary of Radiation Physics Options ................................. 5-57
   - Description of Namelist Variables ........................................ 5-60
   - WRF Output Fields .................................................................... 5-115
   - Special WRF Output Variables ............................................ 5-122

6. WRF Data Assimilation
   - Introduction ........................................................................... 6-1
   - Installing WRFDA for 3D-Var Run ....................................... 6-4
   - Installing WRFPLUS and WRFDA for 4D-Var Run ............... 6-8
   - Running Observation Preprocessor (OBSPROC) ..................... 6-9
   - Running WRFDA ...................................................................... 6-14
   - Radiance Data Assimilations in WRFDA .............................. 6-22
   - Radar Data Assimilation ..................................................... 6-32
   - Precipitation Data Assimilation in WRFDA 4D-Var ............... 6-34
   - Updating WRF boundary conditions .................................... 6-37
   - Background Error and Running GEN_BE ............................. 6-39
   - Additional WRFDA Exercises .............................................. 6-45
   - WRFDA Diagnostics ............................................................. 6-49
   - Generating Ensembles with RANDOMCV ............................ 6-53
   - Hybrid Data Assimilation .................................................... 6-54
   - ETKF Data Assimilation ....................................................... 6-58
   - Description of Namelist Variables ........................................ 6-63

7. Objective Analysis (OBSGRID)
   - Introduction .......................................................................... 7-1
   - Program Flow .......................................................................... 7-2
   - Source of Observations ....................................................... 7-3
8. WRF Software
- WRF Build Mechanism ........................................ 8-1
- Registry .......................................................... 8-5
- I/O Applications Program Interface (I/O API) ............... 8-14
- Timekeeping .................................................... 8-14
- Software Documentation .................................. 8-15
- Performance .................................................. 8-15

9. Post-Processing Programs
- Introduction ...................................................... 9-1
- NCL .............................................................. 9-2
- RIP4 ............................................................ 9-20
- ARWpost ........................................................ 9-29
- UPP .............................................................. 9-36
- VAPOR .......................................................... 9-38

10. Utilities and Tools
- Introduction ...................................................... 10-1
- read_wrf_nc ..................................................... 10-1
- iowrf ............................................................ 10-5
- p_interp .......................................................... 10-6
- TC Bogus Scheme ............................................ 10-9
- v_interp .......................................................... 10-11
- proc_oml.f ...................................................... 10-13
- Tools ............................................................ 10-14

Appendix A: WRF-Fire
- Introduction ...................................................... A-1
- WRF_Fire in idealized cases ................................ A-3
- Fire variables in namelist.input .............................. A-4
- namelist.fire .................................................. A-5
- Running WRF_Fire on real data ............................. A-6
CONTENTS

- Fire state variables ................................................................. A-12
- WRF-Fire software ................................................................. A-12
Chapter 1: Overview

Table of Contents

- Introduction
- The WRF ARW Modeling System Program Components

Introduction

The Advanced Research WRF (ARW) modeling system has been in development for the past few years. The current release is Version 3, available since April 2008. The ARW is designed to be a flexible, state-of-the-art atmospheric simulation system that is portable and efficient on available parallel computing platforms. The ARW is suitable for use in a broad range of applications across scales ranging from meters to thousands of kilometers, including:

- Idealized simulations (e.g. LES, convection, baroclinic waves)
- Parameterization research
- Data assimilation research
- Forecast research
- Real-time NWP
- Hurricane research
- Regional climate research
- Coupled-model applications
- Teaching

The Mesoscale and Microscale Meteorology Division of NCAR is currently maintaining and supporting a subset of the overall WRF code (Version 3) that includes:

- WRF Software Framework (WSF)
- Advanced Research WRF (ARW) dynamic solver, including one-way, two-way nesting and moving nest.
- The WRF Preprocessing System (WPS)
- WRF Data Assimilation (WRF-DA) system which currently supports 3DVAR 4DVAR, and hybrid data assimilation capabilities
- Numerous physics packages contributed by WRF partners and the research community
- Several graphics programs and conversion programs for other graphics tools

And these are the subjects of this document.
The WRF modeling system software is in the public domain and is freely available for community use.

**The WRF Modeling System Program Components**

The following figure shows the flowchart for the WRF Modeling System Version 3.

As shown in the diagram, the WRF Modeling System consists of these major programs:

- The WRF Preprocessing System (WPS)
- WRF-DA
- ARW solver
- Post-processing & Visualization tools

**WPS**

This program is used primarily for real-data simulations. Its functions include 1) defining simulation domains; 2) interpolating terrestrial data (such as terrain, landuse, and soil
types) to the simulation domain; and 3) degribbing and interpolating meteorological data from another model to this simulation domain. Its main features include:

- GRIB 1/2 meteorological data from various centers around the world
- USGS 24 category and MODIS 20 category land datasets; USGS GTOPO30 elevation dataset; Global 5-minutes United Nation FAO, and North-America STATSGO 30 sec soil category dataset; 10-min greenness fraction data based on AVHRR and 30-sec greenness fraction data based on 10 years MODIS; MODIS-based leaf-area index; 0.15 degree monthly albedo and snow albedo data; and 1-degree deep soil temperature data; plus a few specialized datasets
- Map projections for 1) polar stereographic, 2) Lambert-Conformal, 3) Mercator and 4) latitude-longitude
- Nesting
- User-interfaces to input other static data as well as met data

**WRF-DA**

This program is optional, but can be used to ingest observations into the interpolated analyses created by WPS. It can also be used to update WRF model's initial conditions when the WRF model is run in cycling mode. Its main features are as follows:

- It is based on an incremental variational data assimilation technique, and has both 3D-Var and 4D-Var capabilities
- It also includes the capability of hybrid data assimilation (Variational + Ensemble)
- The conjugate gradient method is utilized to minimize the cost function in the analysis control variable space
- Analysis is performed on an un-staggered Arakawa A-grid
- Analysis increments are interpolated to staggered Arakawa C-grid and it gets added to the background (first guess) to get the final analysis of the WRF-model grid
- Conventional observation data input may be supplied either in ASCII format via the “obsproc” utility or “PREPBUFR” format.
- Multiple satellite observation data input may be supplied in BUFR format
- Multiple radar data (reflectivity & radial velocity) input is supplied through ASCII format
- Multiple outer loop to address the nonlinearity
- Capability to compute adjoint sensitivity
- Horizontal component of the background (first guess) error is represented via a recursive filter (for regional) or power spectrum (for global). The vertical component is applied through projections on climatologically generated averaged eigenvectors and its corresponding Eigen values
- Horizontal and vertical background errors are non-separable. Each eigenvector has its own horizontal climatologically-determined length scale
- Preconditioning of the background part of the cost function is done via the control variable transform U defined as $B = UU^T$
It includes the “gen_be” utility to generate the climatological background error covariance estimate via the NMC-method or ensemble perturbations
• A utility program to update WRF boundary condition file after WRF-DA

ARW Solver

This is the key component of the modeling system, which is composed of several initialization programs for idealized, and real-data simulations, and the numerical integration program. The key features of the WRF model include:

• Fully compressible nonhydrostatic equations with hydrostatic option
• Regional and global applications
• Complete Coriolis and curvature terms
• Two-way nesting with multiple nests and nest levels
• Concurrent one-way nesting with multiple nests and nest levels
• Offline one-way nesting with vertical nesting
• Moving nests (prescribed moves and vortex tracking)
• Mass-based terrain-following coordinate
• Vertical grid-spacing can vary with height
• Map-scale factors for these projections:
  o polar stereographic (conformal)
  o Lambert-conformal
  o Mercator (conformal)
  o Latitude and longitude, which can be rotated
• Arakawa C-grid staggering
• Runge-Kutta 2nd and 3rd order time integration options
• Scalar-conserving flux form for prognostic variables
• 2nd to 6th order advection options (horizontal and vertical)
• Monotonic transport and positive-definite advection option for moisture, scalar, tracer, and TKE
• Weighted Essentially Non-Oscillatory (WENO) advection option
• Time-split small step for acoustic and gravity-wave modes:
  o small step horizontally explicit, vertically implicit
  o divergence damping option and vertical time off-centering
  o external-mode filtering option
• Upper boundary absorption and Rayleigh damping
• Lateral boundary conditions
  o idealized cases: periodic, symmetric, and open radiative
  o real cases: specified with relaxation zone
• Full physics options for land-surface, planetary boundary layer, atmospheric and surface radiation, microphysics and cumulus convection
• Ocean models
• Grid analysis nudging using separate upper-air and surface data, and observation nudging
• Spectral nudging
• Digital filter initialization
OVERVIEW

- Adaptive time stepping
- Orographic gravity wave drag
- Stochastic kinetic-energy backscatter scheme
- A number of idealized examples

Graphics and Verification Tools

Several programs are supported, including RIP4 (based on NCAR Graphics), NCAR Graphics Command Language (NCL), and conversion programs for other readily available graphics packages like GrADS.

Program VAPOR, Visualization and Analysis Platform for Ocean, Atmosphere, and Solar Researchers (http://www.vapor.ucar.edu/), is a 3-dimensional data visualization tool, and it is developed and supported by the VAPOR team at NCAR (vapor@ucar.edu).

Program MET, Model Evaluation Tools (http://www.dtcenter.org/met/users/), is developed and supported by the Developmental Testbed Center at NCAR (met_help@ucar.edu).

The details of these programs (with the exception of the MET program) are described more in the later chapters of this user's guide. See the above link for information about MET.
Chapter 2: Software Installation

Table of Contents

- Introduction
- Required Compilers and Scripting Languages
- Required/Optional Libraries to Download
- Post-Processing Utilities
- UNIX Environment Settings
- Building the WRF Code
- Building the WPS Code
- Building the WRFDA Code (for 3DVAR)
- Building the WRFDA Code (for 4DVAR)

Introduction

The WRF modeling system software installation is fairly straightforward on the ported platforms listed below. The model-component portion of the package is mostly self-contained. The WRF model does contain the source code to a Fortran interface to ESMF and the source to FFTPACK. Contained within the WRF system is the WRFDA component, which has several external libraries that the user must install (for various observation types and linear algebra solvers). Similarly, the WPS package, separate from the WRF source code, has additional external libraries that must be built (in support of Grib2 processing). The one external package that all of the systems require is the netCDF library, which is one of the supported I/O API packages. The netCDF libraries and source code are available from the Unidata homepage at http://www.unidata.ucar.edu (select DOWNLOADS, registration required).

There are three tar files for the WRF code. The first is the WRF model (including the real and ideal pre-processors). The second is the WRFDA code. The third tar file is for WRF chemistry. In order to run the WRF chemistry code, both the WRF model and the chemistry tar file must be combined.

The WRF model has been successfully ported to a number of Unix-based machines. We do not have access to all of them and must rely on outside users and vendors to supply the required configuration information for the compiler and loader options. Below is a list of the supported combinations of hardware and software for WRF.
The WRF model may be built to run on a single-processor machine, a shared-memory machine (that uses the OpenMP API), a distributed memory machine (with the appropriate MPI libraries), or on a distributed cluster (utilizing both OpenMP and MPI). The WRFDA and WPS packages run on the above-listed systems.

**Required Compilers and Scripting Languages**

The majority of the WRF model, WPS, and WRFDA codes are written in Fortran (what many refer to as Fortran 90). The software layer, RSL, which sits between WRF and WRFDA, and the MPI interface is written in C. WPS makes direct calls to the MPI libraries for distributed memory message passing. There are also ancillary programs that are written in C to perform file parsing and file construction, which are required for default building of the WRF modeling code. Additionally, the WRF build mechanism uses several scripting languages: including perl, Cshell and Bourne shell. The traditional UNIX text/file processing utilities are used: make, m4, sed, and awk. See Chapter 8: WRF Software (Required Software) for a more detailed listing of the necessary pieces for the WRF build.

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Hardware</th>
<th>OS</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray</td>
<td>XC30 Intel</td>
<td>Linux</td>
<td>Intel</td>
</tr>
<tr>
<td>Cray</td>
<td>XE AMD</td>
<td>Linux</td>
<td>Intel</td>
</tr>
<tr>
<td>IBM</td>
<td>Power Series</td>
<td>AIX</td>
<td>vendor</td>
</tr>
<tr>
<td>IBM</td>
<td>Intel</td>
<td>Linux</td>
<td>Intel / PGI / gfortran</td>
</tr>
<tr>
<td>SGI</td>
<td>IA64 / Opteron</td>
<td>Linux</td>
<td>Intel</td>
</tr>
<tr>
<td>COTS*</td>
<td>IA32</td>
<td>Linux</td>
<td>Intel / PGI / gfortran / g95 / PathScale</td>
</tr>
<tr>
<td>COTS</td>
<td>IA64 / Opteron</td>
<td>Linux</td>
<td>Intel / PGI / gfortran / PathScale</td>
</tr>
<tr>
<td>Mac</td>
<td>Power Series</td>
<td>Darwin</td>
<td>xlf / g95 / PGI / Intel</td>
</tr>
<tr>
<td>Mac</td>
<td>Intel</td>
<td>Darwin</td>
<td>gfortran / PGI / Intel</td>
</tr>
<tr>
<td>NEC</td>
<td>NEC</td>
<td>Linux</td>
<td>vendor</td>
</tr>
<tr>
<td>Fujitsu</td>
<td>FX10 Intel</td>
<td>Linux</td>
<td>vendor</td>
</tr>
</tbody>
</table>

* Commercial Off-The-Shelf systems
Required/Optional Libraries to Download

The only library that is *always* required is the netCDF package from Unidata (login > Downloads > NetCDF). Most of the WRF post-processing packages assume that the data from the WRF model, the WPS package, or the WRFDA program are using the netCDF libraries. One may also need to add ‘/path-to-netcdf/netcdf/bin’ to their path so that they may execute netCDF utility commands, such as `ncdump`. Use a netCDF version that is 3.6.1 or later. To utilize the compression capabilities, use netCDF 4.0 or later. Note that compression will require the use of HDF5.

*Note 1:* If one wants to compile WRF system components on a Linux or Darwin system that has access to multiple compilers, link the correct external libraries. For example, do not link the libraries built with PathScale when compiling the WRF components with gfortran. Even more, the same options when building the netCDF libraries must be used when building the WRF code (32 vs 64 bit, assumptions about underscores in the symbol names, etc.).

*Note 2:* If netCDF-4 is used, be sure that it is installed without activating parallel I/O based on HDF5. The WRF modeling system is able to use either the classic data model from netCDF-3 or the compression options supported in netCDF-4.

If you are going to be running distributed memory WRF jobs, you need a version of MPI. You can pick up a version of mpich, but you might want your system group to install the code. A working installation of MPI is required prior to a build of WRF using distributed memory. Either MPI-1 or MPI-2 are acceptable. Do you already have an MPI lying around? Try

```
which mpif90
which mpicc
which mpirun
```

If these are all defined executables in your path, you are probably OK. Make sure your paths are set up to point to the MPI `lib`, `include`, and `bin` directories. As with the netCDF libraries, you must build MPI consistently with the WRF source code.

Note that to output WRF model data in Grib1 format, Todd Hutchinson (WSI) has provided a complete source library that is included with the software release. However, when trying to link the WPS, the WRF model, and the WRFDA data streams together, always use the netCDF format.

*Note 3:* The entire step-by-step recipe for building the WRF and WPS packages is available at: http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php This page includes complete turn-key directions, from tests of your machines’s utilities all the way up through where to download real-time data.
Post-Processing Utilities

The more widely used *(and therefore supported)* WRF post-processing utilities are:

- **NCL** ([homepage](http://www.ncl.ucar.edu) and [WRF download](http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_examples.htm))
  - NCAR Command Language written by NCAR’s Computer Information Systems Laboratory (formerly the Scientific Computing Division)
  - NCL scripts written and maintained by WRF support
  - many template scripts are provided that are tailored for specific real-data and ideal-data cases
  - raw WRF output can be input with the NCL scripts
  - interactive or command-file driven

- **GrADS** ([homepage](http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/GrADS/GrADS.htm) and [WRF download](http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/GrADS/GrADS.htm))
  - download GrADS executable, build format converter
  - programs *(ARWpost)* are available to convert the WRF output into an input format suitable for GrADS
  - simple to generate publication quality
  - interactive or command-file driven

- **RIP4** ([homepage](http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/RIP4/RIP4.htm) and [WRF download](http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/RIP4/RIP4.htm))
  - RIP4 written and maintained by Mark Stoelinga, UW
  - interpolation to various surfaces, trajectories, hundreds of diagnostic calculations
  - Fortran source provided
  - based on the NCAR Graphics package
  - pre-processor converts WRF, WPS, and WRFDA data to RIP input format
  - table driven

UNIX Environment Settings

There are only a few environmental settings that are WRF system related. Most of these are not required, but when things start acting badly, test some out. In Cshell syntax:

- `setenv WRF_EM_CORE 1`
  - explicitly defines which model core to build
- `setenv WRF_NMM_CORE 0`
  - explicitly defines which model core NOT to build
- `setenv WRF_DA_CORE 0`
  - explicitly defines no data assimilation
- `setenv NETCDF /usr/local/netcdf` (or wherever you have it stored)
  - all of the WRF components want both the lib and the include directories
- `setenv OMP_NUM_THREADS n` (where *n* is the number of procs to use)
if you have OpenMP on your system, this is how to specify the number of threads

- `setenv MP_STACK_SIZE 64000000`
  - OpenMP blows through the stack size, set it large
  - However, if the model still crashes, it may be a problem of over-specifying stack size. Set stack size sufficiently large, but not unlimited.
  - On some systems, the equivalent parameter could be KMP_STACKSIZE, or OMP_STACKSIZE

- `unlimit`
  - especially if you are on a small system

Building the WRF Code

The WRF code has a fairly complicated build mechanism. It tries to determine the architecture that you are on, and then presents you with options to allow you to select the preferred build method. For example, if you are on a Linux machine, it determines whether this is a 32 or 64 bit machine, and then prompts you for the desired usage of processors (such as serial, shared memory, or distributed memory). You select from among the available compiling options in the build mechanism. For example, do not choose a PGI build if you do not have PGI compilers installed on your system.

An instructional web site describes the sequence of steps required to build the WRF and WPS codes (though the instructions are specifically given for tcsh and GNU compilers).


- Get the WRF zipped tar file for WRFV3 from
  - [http://www2.mmm.ucar.edu/wrf/users/download/get_source.html](http://www2.mmm.ucar.edu/wrf/users/download/get_source.html)
  - Always get the latest version if you are not trying to continue a long project, or duplicate previous work

- unzip and untar the file
  - `gzip -cd WRFV3.TAR.gz | tar -xf -`
  - Alternatively `tar -xzW WRFV3.TAR.gz` on some systems

- cd WRFV3
- `./configure`
  - **serial** means single processor
  - **smpar** means Symmetric Multi-Processing/Shared Memory Parallel (OpenMP) – this does not reliably work on most non-IBM machines
  - **dmpar** means Distributed Memory Parallel (MPI)
  - **dm+sm** means Distributed Memory with Shared Memory (for example, MPI across nodes with OpenMP within a node) – usually better performance is through **dmpar** only
  - The second option is for nesting: 0 = no nesting, 1 = standard static nesting, 2 = nesting with a prescribed set of moves, 3 = nesting that allows
a domain to follow a vortex (typhoon tracking)

- A typical option that may be included on the ./configure command is the flag “-d” (for debug). This option removes optimization, which is useful when running a debugger (such as gdb or dbx)
- For bounds checking and some additional exception handling, the debugging flag “-D” may be selected. Only PGI, Intel, and gfortran have been set up to use this option.

- ./compile em_real (or any of the directory names in ./WRFV3/test directory)
- ls -ls main/*.exe
  - If you built a real-data case, you should see ndown.exe, real.exe, and wrf.exe
  - If you built an ideal-data case, you should see ideal.exe and wrf.exe

The WRF code supports a parallel build option, an option that compiles separate source code files in the WRF directories at the same time on separate processors (though those processors need to share memory) via a parallel make. The purpose of the parallel build option is to be able to speed-up the time required to construct executables. In practice, users typically see approximately a 2x speed-up, a limit imposed by the various dependencies in the code due to modules and USE association. To enable the parallel build option, the user sets an environment variable, J. In csh, to utilize two processors, before the ./compile command, issue the following:

```
setenv J "-j 2"
```

Users may wish to only use a single processor for the build. In which case:

```
setenv J "-j 1"
```

Users wishing to run the WRF chemistry code must first download the WRF model tar file, and untar it. Then the chemistry code is untar’ed in the WRFV3 directory (this is the chem directory structure). Once the source code from the tar files is combined, then users may proceed with the WRF chemistry build.

**Building the WPS Code**

**Building WPS requires that WRFV3 be already built.**

**If you plan to use Grib2 data, additional libraries for zlib, png, and jasper are required. Please see details in Chapter 3.**

- Get the WPS zipped tar file WPSV3.TAR.gz from
  - http://www2.mmm.ucar.edu/wrf/users/download/get_source.html
- Also download the geographical datasets from the same page. There are new data sets for land cover for North America (NLCD), and high-resolution urban data sets for select North American cities.
- Unzip and untar the source code file
  - gzip -cd WPSV3.TAR.gz | tar -xf -
• **cd WPS**
• **./configure**  
  o Choose one of the options  
  o Usually, serial builds are the best for an initial test. Most large domains work with a single processor for WPS  
  o WPS requires that you build for the appropriate Grib decoding. Select an option that is suitable for the data you will use with the ungrib program (the Grib2 option will work for either Grib1 or Grib2 data)  
  o If you select a Grib2 option, you must have those libraries prepared and built in advance (see the chapter on WPS for the location of these compression libraries). Add the paths to these libraries and include files using variables COMPRESSION_LIBS and COMPRESSION_INC in `configure.wps`. Also inside the `configure.wps` file is the location of the built WRFV3 directory, which needs to be modified. This is how the WPS picks up all of the required IO pieces to build the geogrid.exe and metgrid.exe files.
• **./compile**
• **ls -ls *.exe**  
  o You should see `geogrid.exe`, `ungrib.exe`, and `metgrid.exe` (if you are missing both `geogrid.exe` and `metgrid.exe`, you probably need to fix where the path to WRF is pointing in the `configure.wps` file; if you are missing `ungrib.exe`, try a Grib1-only build to further isolate the problem)
• **ls -ls util/*.exe**  
  o You should see a number of utility executables: `avg_tsfc.exe`, `calc_ecmwf_p.exe`, `glprint.exe`, `g2print.exe`, `height_ukmo.exe`, `mod_levs.exe`, `plotfmt.exe`, `plotgrids.exe`, and `rd_intermediate.exe` (files requiring NCAR Graphics are `plotfmt.exe` and `plotgrids.exe`)
  
  • If `geogrid.exe` and `metgrid.exe` executables are missing, the path to the built WRFV3 directory structure is probably incorrect (found inside the `configure.wps` file)
  • If the `ungrib.exe` is missing, the Grib2 libraries are probably not linked or built correctly
  • If the `plotfmt.exe` or the `plotgrids.exe` programs is missing, the NCAR Graphics path is probably set incorrectly

### Building the WRFDA Code (for 3DVAR)

WRFDA uses the same build mechanism as WRF; thus, this mechanism must be instructed to configure and build the code for WRFDA rather than WRF. Additionally, the paths to libraries needed by WRFDA code must be set, as described in the steps below.
• Get the WRFDA zipped tar file, WRFDA_V3.7.TAR.gz, from
  http://www2.mmm.ucar.edu/wrf/users/wrfda/download/get_source.html
• Unzip and untar the WRFDA code
  o \texttt{tar xf WRFDA\_V3.7.TAR.gz}
  o This will create a directory, WRFDA
• \texttt{cd WRFDA}
  o In addition to netCDF, set up environmental variables pointing to
    additional libraries required by WRFDA, such as RTTOV
  o Please note: only the netCDF library is mandatory to compile the basic
    WRFDA system; all other libraries are optional
  o If you intend to use satellite radiance data, an RTM (Radiative Transfer
    Model) is required. The current RTM versions that WRFDA uses are
    \texttt{CRTM} \texttt{v2.1.3} and \texttt{RTTOV} \texttt{v11}. WRFDA can compile with CRTM only,
    or RTTOV only, or both CRTM and RTTOV together

To compile WRFDA with CRTM: \texttt{setenv CRTM 1}

(Note: the latest available CRTM, version 2.1.3, is included in this release
version and it will be compiled automatically when the appropriate
environmental variable is set. Users do not need to download and install
CRTM).

To compile WRFDA with RTTOV: RTTOV still must be downloaded (https://nwpsaf.eu/deliverables/rtm/rtm_rttov11.html) and
installed using the same compiler that will be used to build WRFDA, since
the library produced by one compiler may not be compatible with code
compiled with another. Then, the necessary environment variable should
be set with

\texttt{setenv RTTOV ${path\_for\_RTTOV}}

• \texttt{./configure wrfda}
  o \texttt{serial} means single processor
  o \texttt{dmpar} means Distributed Memory Parallel (MPI)
  o \texttt{smpar} and \texttt{dm+sm} are not recommended for use with WRFDA
• WRFDA also supports \texttt{parallel build}.
• \texttt{./compile all\_wrfvar}
• \texttt{ls -ls var/build/*.exe}
  o If the compilation was successful, \texttt{da\_wrfvar.exe},
    \texttt{da\_update\_bc.exe}, and other executables should be found in the
    var/build directory. Their links are in the var/da directory;
    \texttt{obsproc.exe} should be found in the var/obsproc/src directory
Building the WRFDA Code (for 4DVAR)

Building WRFDA 4DVAR requires that WRFPLUSV3.7 be already built.

- Get the WRFPLUSV3.7 zipped tar file WRFPLUS_V3.7.tar.gz from
  - http://www2.mmm.ucar.edu/wrf/users/wrfda/download/wrfplus.html
- unzip and untar the source code file
  - tar -xf WRFPLUS_V3.7.tar.gz
- cd WRFPLUSV3
- ./configure wrfplus
  - serial means single processor
  - dmpar means Distributed Memory Parallel (MPI)
  - (Note: WRFPLUS does not support Shared Memory Parallel and WRFPLUS is compiled as realsize=8)
- WRFPLUS also supports parallel build.
- ./compile em_real
- ls -ls main/*.exe
  - you should see ndown.exe, real.exe, and wrf.exe

- Set up the environmental variable pointing to WRFPLUS_DIR.
  - setenv WRFPLUS_DIR ${path_of_wrfplusv3.7} (csh)
  - export WRFPLUS_DIR=${path_of_wrfplusv3.7} (bash)
- Please refer to above section “Building WRFDA code (for 3DVAR)” to download code and set up necessary environmental variables.
- ./configure 4dvar
  - serial means single processor
  - dmpar means Distributed Memory Parallel (MPI)
- ./compile all_wrfvar
- ls -ls var/build/*.exe
  - If the compilation was successful, da_wrfvar.exe, da_update_bc.exe, and other executables should be found in the var/build directory. Their links are in the var/da directory; obsproc.exe should be found in the var/obsproc/src directory
Chapter 3: WRF Preprocessing System (WPS)

Table of Contents

- Introduction
- Function of Each WPS Program
- Installing the WPS
- Running the WPS
- Creating Nested Domains with the WPS
- Selecting Between USGS and MODIS-based Land Use Data
- Selecting Static Data for the Gravity Wave Drag Scheme
- Using Multiple Meteorological Data Sources
- Using Non-isobaric Meteorological Datasets
- Alternative Initialization of Lake SSTs
- Parallelism in the WPS
- Checking WPS Output
- WPS Utility Programs
- Writing Meteorological Data to the Intermediate Format
- Required Meteorological Fields for Running WRF
- Creating and Editing Vtables
- Writing Static Data to the Geogrid Binary Format
- Creating an Urban Fraction Field from NLCD Data
- Description of Namelist Variables
- Description of GEOGRID.TBL Options
- Description of index Options
- Description of METGRID.TBL Options
- Available Interpolation Options in Geogrid and Metgrid
- Land Use and Soil Categories in the Static Data
- WPS Output Fields

Introduction

The WRF Preprocessing System (WPS) is a set of three programs whose collective role is to prepare input to the real program for real-data simulations. Each of the programs performs one stage of the preparation: geogrid defines model domains and interpolates static geographical data to the grids; ungrib extracts meteorological fields from GRIB-formatted files; and metgrid horizontally interpolates the meteorological fields extracted by ungrib to the model grids defined by geogrid. The work of vertically interpolating meteorological fields to WRF eta levels is performed within the real program.
The data flow between the programs of the WPS is shown in the figure above. Each of the WPS programs reads parameters from a common namelist file, as shown in the figure. This namelist file has separate namelist records for each of the programs and a shared namelist record, which defines parameters that are used by more than one WPS program. Not shown in the figure are additional table files that are used by individual programs. These tables provide additional control over the programs’ operations, though they generally do not need to be changed by the user. The GEOGRID.TBL, METGRID.TBL, and Vtable files are explained later in this document, though for now, the user need not be concerned with them.

The build mechanism for the WPS, which is very similar to the build mechanism used by the WRF model, provides options for compiling the WPS on a variety of platforms. When MPI libraries and suitable compilers are available, the metgrid and geogrid programs may be compiled for distributed memory execution, which allows large model domains to be processed in less time. The work performed by the ungrib program is not amenable to parallelization, so ungrib may only be run on a single processor.

Function of Each WPS Program

The WPS consists of three independent programs: geogrid, ungrib, and metgrid. Also included in the WPS are several utility programs, which are described in the section on utility programs. A brief description of each of the three main programs is given below, with further details presented in subsequent sections.
Program geogrid

The purpose of geogrid is to define the simulation domains, and interpolate various terrestrial data sets to the model grids. The simulation domains are defined using information specified by the user in the “geogrid” namelist record of the WPS namelist file, namelist.wps. In addition to computing the latitude, longitude, and map scale factors at every grid point, geogrid will interpolate soil categories, land use category, terrain height, annual mean deep soil temperature, monthly vegetation fraction, monthly albedo, maximum snow albedo, and slope category to the model grids by default. Global data sets for each of these fields are provided through the WRF download page, and, because these data are time-invariant, they only need to be downloaded once. Several of the data sets are available in only one resolution, but others are made available in resolutions of 30", 2', 5', and 10'; here, " denotes arc seconds and ' denotes arc minutes. The user need not download all available resolutions for a data set, although the interpolated fields will generally be more representative if a resolution of data near to that of the simulation domain is used. However, users who expect to work with domains having grid spacings that cover a large range may wish to eventually download all available resolutions of the static terrestrial data.

Besides interpolating the default terrestrial fields, the geogrid program is general enough to be able to interpolate most continuous and categorical fields to the simulation domains. New or additional data sets may be interpolated to the simulation domain through the use of the table file, GEOGRID.TBL. The GEOGRID.TBL file defines each of the fields that will be produced by geogrid; it describes the interpolation methods to be used for a field, as well as the location on the file system where the data set for that field is located.

Output from geogrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, geogrid can be made to write its output in NetCDF for easy visualization using external software packages, including ncview, NCL, and RIP4.

Program ungrib

The ungrib program reads GRIB files, "degribs" the data, and writes the data in a simple format called the intermediate format (see the section on writing data to the intermediate format for details on the format). The GRIB files contain time-varying meteorological fields and are typically from another regional or global model, such as NCEP's NAM or GFS models. The ungrib program can read GRIB Edition 1 and, if compiled with a "GRIB2" option, GRIB Edition 2 files.

GRIB files typically contain more fields than are needed to initialize WRF. Both versions of the GRIB format use various codes to identify the variables and levels in the GRIB file. Ungrib uses tables of these codes – called Vtables, for "variable tables" – to define which fields to extract from the GRIB file and write to the intermediate format. Details about the codes can be found in the WMO GRIB documentation and in documentation from the originating center. Vtables for common GRIB model output files are provided with the ungrib software.
Vtables are provided for NAM 104 and 212 grids, the NAM AWIP format, GFS, the NCEP/NCAR Reanalysis archived at NCAR, RUC (pressure level data and hybrid coordinate data), AFWA's AGRMET land surface model output, ECMWF, and other data sets. Users can create their own Vtable for other model output using any of the Vtables as a template; further details on the meaning of fields in a Vtable are provided in the section on creating and editing Vtables.

Ungrib can write intermediate data files in any one of three user-selectable formats: WPS – a new format containing additional information useful for the downstream programs; SI – the previous intermediate format of the WRF system; and MM5 format, which is included here so that ungrib can be used to provide GRIB2 input to the MM5 modeling system. Any of these formats may be used by WPS to initialize WRF, although the WPS format is recommended.

**Program metgrid**

The metgrid program horizontally interpolates the intermediate-format meteorological data that are extracted by the ungrib program onto the simulation domains defined by the geogrid program. The interpolated metgrid output can then be ingested by the WRF real program. The range of dates that will be interpolated by metgrid are defined in the “share” namelist record of the WPS namelist file, and date ranges must be specified individually in the namelist for each simulation domain. Since the work of the metgrid program, like that of the ungrib program, is time-dependent, metgrid is run every time a new simulation is initialized.

Control over how each meteorological field is interpolated is provided by the METGRID.TBL file. The METGRID.TBL file provides one section for each field, and within a section, it is possible to specify options such as the interpolation methods to be used for the field, the field that acts as the mask for masked interpolations, and the grid staggering (e.g., U, V in ARW; H, V in NMM) to which a field is interpolated.

Output from metgrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, metgrid can be made to write its output in NetCDF for easy visualization using external software packages, including the new version of RIP4.

**Installing the WPS**

The WRF Preprocessing System uses a build mechanism similar to that used by the WRF model. External libraries for geogrid and metgrid are limited to those required by the WRF model, since the WPS uses the WRF model's implementations of the WRF I/O API; consequently, *WRF must be compiled prior to installation of the WPS* so that the I/O API libraries in the WRF external directory will be available to WPS programs. Additionally, the ungrib program requires three compression libraries for GRIB Edition 2.
support; however, if support for GRIB2 data is not needed, ungrib can be compiled without these compression libraries.

**Required Libraries**

The only library that is required to build the WRF model is NetCDF. The user can find the source code, precompiled binaries, and documentation at the UNIDATA home page ([http://www.unidata.ucar.edu/software/netcdf/](http://www.unidata.ucar.edu/software/netcdf/)). Most users will select the NetCDF I/O option for WPS due to the easy access to utility programs that support the NetCDF data format, and before configuring the WPS, users should ensure that the environment variable NETCDF is set to the path of the NetCDF installation.

Where WRF adds a software layer between the model and the communications package, the WPS programs geogrid and metgrid make MPI calls directly. Most multi-processor machines come preconfigured with a version of MPI, so it is unlikely that users will need to install this package by themselves.

Three libraries are required by the ungrib program for GRIB Edition 2 compression support. Users are encouraged to engage their system administrators for the installation of these packages so that traditional library paths and include paths are maintained. Paths to user-installed compression libraries are handled in the configure.wps file by the COMPRESSION_LIBS and COMPRESSION_INC variables. As an alternative to manually editing the COMPRESSION_LIBS and COMPRESSION_INC variables in the configure.wps file, users may set the environment variables JASPERLIB and JASPERINC to the directories holding the JasPer library and include files before configuring the WPS; for example, if the JasPer libraries were installed in `/usr/local/jasper-1.900.1`, one might use the following commands (in csh or tcsh):

```
> setenv JASPERLIB /usr/local/jasper-1.900.1/lib
> setenv JASPERINC /usr/local/jasper-1.900.1/include
```

If the zlib and PNG libraries are not in a standard path that will be checked automatically by the compiler, the paths to these libraries can be added on to the JasPer environment variables; for example, if the PNG libraries were installed in `/usr/local/libpng-1.2.29` and the zlib libraries were installed in `/usr/local/zlib-1.2.3`, one might use

```
> setenv JASPERLIB "${JASPERLIB} -L/usr/local/libpng-1.2.29/lib -L/usr/local/zlib-1.2.3/lib"
> setenv JASPERINC "${JASPERINC} -I/usr/local/libpng-1.2.29/include -I/usr/local/zlib-1.2.3/include"
```

after having previously set JASPERLIB and JASPERINC.

1) JasPer (an implementation of the JPEG2000 standard for "lossy" compression)  
[http://www.ece.uvic.ca/~mdadams/jasper/](http://www.ece.uvic.ca/~mdadams/jasper/)  
Go down to “JasPer software”, one of the "click here" parts is the source.

```
> ./configure
```
> make
> make install

Note: The GRIB2 libraries expect to find include files in "jasper/jasper.h", so it may be necessary to manually create a "jasper" subdirectory in the "include" directory created by the JasPer installation, and manually link header files there.

2) PNG (compression library for "lossless" compression)
http://www.libpng.org/pub/png/libpng.html
Scroll down to "Source code" and choose a mirror site.

    > ./configure
    > make check
    > make install

3) zlib (a compression library used by the PNG library)
http://www.zlib.net/
Go to "The current release is publicly available here" section and download.

    > ./configure
    > make
    > make install

To get around portability issues, the NCEP GRIB libraries, w3 and g2, have been included in the WPS distribution. The original versions of these libraries are available for download from NCEP at http://www.nco.ncep.noaa.gov/pmb/codes/GRIB2/. The specific tar files to download are g2lib and w3lib. Because the ungrib program requires modules from these files, they are not suitable for usage with a traditional library option during the link stage of the build.

Required Compilers and Scripting Languages

The WPS requires the same Fortran and C compilers as were used to build the WRF model, since the WPS executables link to WRF's I/O API libraries. After executing the ./configure command in the WPS directory, a list of supported compilers on the current system architecture are presented.

WPS Installation Steps

- Download the WPSV3.TAR.gz file and unpack it at the same directory level as WRFV3, as shown below.

    > ls
    -rw-r--r-- 1 563863 WPS.TAR.gz
drwxr-xr-x 18 4096 WRFV3
    > gzip -d WPSV3.TAR.gz
    > tar xf WPSV3.TAR
At this point, a listing of the current working directory should at least include the directories WRFV3 and WPS. First, compile WRF (see the instructions for installing WRF in Chapter 2). Then, after the WRF executables are generated, change to the WPS directory and issue the configure command followed by the compile command as below.

> cd WPS

> ./configure

  o Choose one of the configure options

> ./compile >& compile.output

After issuing the compile command, a listing of the current working directory should reveal symbolic links to executables for each of the three WPS programs: geogrid.exe, ungrib.exe, and metgrid.exe. If any of these links do not exist, check the compilation output in compile.output to see what went wrong.
Running the WPS

There are essentially three main steps to running the WRF Preprocessing System:

1. Define a model coarse domain and any nested domains with `geogrid`.
2. Extract meteorological fields from GRIB data sets for the simulation period with `ungrib`.
3. Horizontally interpolate meteorological fields to the model domains with `metgrid`.

When multiple simulations are to be run for the same model domains, it is only necessary to perform the first step once; thereafter, only time-varying data need to be processed for each simulation using steps two and three. Similarly, if several model domains are being run for the same time period using the same meteorological data source, it is not necessary to run `ungrib` separately for each simulation. Below, the details of each of the three steps are explained.

Step 1: Define model domains with `geogrid`

In the root of the WPS directory structure, symbolic links to the programs `geogrid.exe`, `ungrib.exe`, and `metgrid.exe` should exist if the WPS software was successfully installed. In addition to these three links, a `namelist.wps` file should exist. Thus, a listing in the WPS root directory should look something like:

```
$ ls
drwxr-xr-x 2 4096 arch
-rw-r-xr-x 1 1672 clean
-rw-r-xr-x 1 3510 compile
-rw-r--r-- 1 85973 compile.output
-rw-r-xr-x 1 4257 configure
-rw-r--r-- 1 2486 configure.wps
drwxr-xr-x 4 4096 geogrid
lrwxrwxrwx 1 23 geogrid.exe -> geogrid/src/geogrid.exe
-rw-r-xr-x 1 1328 link_grib.csh
drwxr-xr-x 3 4096 metgrid
lrwxrwxrwx 1 23 metgrid.exe -> metgrid/src/metgrid.exe
-rw-r--r-- 1 1101 namelist.wps
-rw-r--r-- 1 1987 namelist.wps.all_options
-rw-r--r-- 1 1075 namelist.wps.global
-rw-r--r-- 1 652 namelist.wps.nmm
-rw-r--r-- 1 4786 README
drwxr-xr-x 4 4096 ungrib
lrwxrwxrwx 1 21 ungrib.exe -> ungrib/src/ungrib.exe
drwxr-xr-x 3 4096 util
```

The model coarse domain and any nested domains are defined in the “geogrid” namelist record of the `namelist.wps` file, and, additionally, parameters in the “share” namelist record need to be set. An example of these two namelist records is given below, and the user is referred to the description of namelist variables for more information on the purpose and possible values of each variable.
&share
  wrf_core = 'ARW',
  max_dom = 2,
  start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
  end_date   = '2008-03-24_18:00:00','2008-03-24_12:00:00',
  interval_seconds = 21600,
  io_form_geogrid = 2
/

&geogrid
  parent_id         =   1,   1,
  parent_grid_ratio = 1,   3,
  i_parent_start    =   1,  31,
  j_parent_start    =   1,  17,
  e_we              =  74, 112,
  e_sn              =  61,  97,
  geog_data_res     = 'default','default',
  dx = 30000,
  dy = 30000,
  map_proj = 'lambert',
  ref_lat   = 34.83,
  ref_lon   = -81.03,
  truelat1  = 30.0,
  truelat2  = 60.0,
  stand_lon = -98.,
  geog_data_path = '/mmm/users/wrfhelp/WPS_GEOG/
/

To summarize a set of typical changes to the “share” namelist record relevant to geogrid, the WRF dynamical core must first be selected with wrf_core. If WPS is being run for an ARW simulation, wrf_core should be set to ‘ARW’, and if running for an NMM simulation, it should be set to ‘NMM’. After selecting the dynamical core, the total number of domains (in the case of ARW) or nesting levels (in the case of NMM) must be chosen with max_dom. Since geogrid produces only time-independent data, the start_date, end_date, and interval_seconds variables are ignored by geogrid. Optionally, a location (if not the default, which is the current working directory) where domain files should be written to may be indicated with the opt_output_from_geogrid_path variable, and the format of these domain files may be changed with io_form_geogrid.

In the “geogrid” namelist record, the projection of the simulation domain is defined, as are the size and location of all model grids. The map projection to be used for the model domains is specified with the map_proj variable. Each of the four possible map projections in the ARW are shown graphically in the full-page figure below, and the namelist variables used to set the parameters of the projection are summarized in the following table.

<table>
<thead>
<tr>
<th>Map projection / value of map_proj</th>
<th>Projection parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lambert Conformal / 'lambert'</td>
<td>truelat1</td>
</tr>
<tr>
<td></td>
<td>truelat2 (optional)</td>
</tr>
<tr>
<td></td>
<td>stand_lon</td>
</tr>
<tr>
<td>Mercator / 'mercator'</td>
<td>truelat1</td>
</tr>
</tbody>
</table>
In the illustrations of the Lambert conformal, polar stereographic, and Mercator projections, it may be seen that the so-called true latitude (or true latitudes, in the case of the Lambert conformal), is the latitude at which the surface of projection intersects or is tangent to the surface of the earth. At this latitude, there is no distortion in the distances in the map projection, while at other latitudes, the distance on the surface of the earth is related to the distance on the surface of projection by a \textit{map scale factor}. Ideally, the map projection and its accompanying parameters should be chosen to minimize the maximum distortion within the area covered by the model grids, since a high amount of distortion, evidenced by map scale factors significantly different from unity, can restrict the model time step more than necessary. As a general guideline, the polar stereographic projection is best suited for high-latitude WRF domains, the Lambert conformal projection is well-suited for mid-latitude domains, and the Mercator projection is good for low-latitude domains or domains with predominantly west-east extent. The cylindrical equidistant projection is required for global ARW simulations, although in its rotated aspect (i.e., when pole_lat, pole_lon, and stand_lon are changed from their default values) it can also be well-suited for regional domains anywhere on the earth’s surface.
When configuring a rotated latitude-longitude grid, the namelist parameters `pole_lat`, `pole_lon`, and `stand_lon` are changed from their default values. The parameters `pole_lat` and `pole_lon` specify the latitude and longitude of the geographic north pole within the model’s computational grid, and `stand_lon` gives the rotation about the earth’s axis. In the context of the ARW, the computational grid refers to the regular latitude-longitude grid on which model computation is done, and on whose latitude circles Fourier filters are applied at high latitudes; users interested in the details of this filtering are referred to the WRF Version 3 Technical Note, and here, it suffices to note that the computational latitude-longitude grid is always represented with computational latitude lines running parallel to the x-axis of the model grid and computational longitude lines running parallel to the y-axis of the grid.

If the earth’s geographic latitude-longitude grid coincides with the computational grid, a global ARW domain shows the earth’s surface as it is normally visualized on a regular latitude-longitude grid. If instead the geographic grid does not coincide with the model computational grid, geographical meridians and parallels appear as complex curves. The difference is most easily illustrated by way of example. In top half of the figure below, the earth is shown with the geographical latitude-longitude grid coinciding with the computational latitude-longitude grid. In the bottom half, the geographic grid (not shown) has been rotated so that the geographic poles of the earth are no longer located at the poles of the computational grid.
When WRF is to be run for a regional domain configuration, the location of the coarse domain is determined using the ref_lat and ref_lon variables, which specify the latitude and longitude, respectively, of the center of the coarse domain. If nested domains are to be processed, their locations with respect to the parent domain are specified with the i_parent_start and j_parent_start variables; further details of setting up nested domains are provided in the section on nested domains. Next, the dimensions of the coarse domain are determined by the variables dx and dy, which specify the nominal grid distance in the x-direction and y-direction, and e_we and e_sn, which give the number of velocity points (i.e., u-staggered or v-staggered points) in the x- and y-directions; for the 'lambert', 'mercator', and 'polar' projections, dx and dy are given in meters, and for the 'lat-lon' projection, dx and dy are given in degrees. For nested domains, only the variables e_we and e_sn are used to determine the dimensions of the grid, and dx and dy should not be specified for nests, since their values are determined recursively based on the values of the parent_grid_ratio and parent_id variables, which specify the ratio of a nest's parent grid distance to the nest's grid distance and the grid number of the nest's parent, respectively.

If the regular latitude-longitude projection will be used for a regional domain, care must be taken to ensure that the map scale factors in the region covered by the domain do not deviate significantly from unity. This can be accomplished by rotating the projection such that the area covered by the domain is located near the equator of the projection, since, for the regular latitude-longitude projection, the map scale factors in the x-direction are given by the cosine of the computational latitude. For example, in the figure above showing the unrotated and rotated earth, it can be seen that, in the rotated aspect, New Zealand is located along the computational equator, and thus, the rotation used there would be suitable for a domain covering New Zealand. As a general guideline for rotating the latitude-longitude projection for regional domains, the namelist parameters pole_lat, pole_lon, and stand_lon may be chosen according to the formulas in the following table.

<table>
<thead>
<tr>
<th></th>
<th>(ref_lat, ref_lon) in N.H.</th>
<th>(ref_lat, ref_lon) in S.H.</th>
</tr>
</thead>
<tbody>
<tr>
<td>pole_lat</td>
<td>90.0 - ref_lat</td>
<td>90.0 + ref_lat</td>
</tr>
<tr>
<td>pole_lon</td>
<td>180.0</td>
<td>0.0</td>
</tr>
<tr>
<td>stand_lon</td>
<td>-ref_lon</td>
<td>180.0 - ref_lon</td>
</tr>
</tbody>
</table>

For global WRF simulations, the coverage of the coarse domain is, of course, global, so ref_lat and ref_lon do not apply, and dx and dy should not be specified, since the nominal grid distance is computed automatically based on the number of grid points. Also, it should be noted that the latitude-longitude, or cylindrical equidistant, projection (map_proj = 'lat-lon') is the only projection in WRF that can support a global domain. Nested domains within a global domain must not cover any area north of computational latitude +45 or south of computational latitude -45, since polar filters are applied poleward of these latitudes (although the cutoff latitude can be changed in the WRF namelist).
Besides setting variables related to the projection, location, and coverage of model domains, the path to the static geographical data sets must be correctly specified with the `geog_data_path` variable. Also, the user may select which resolution of static data geogrid will interpolate from using the `geog_data_res` variable, whose value should match one of the resolutions of data in the GEOGRID.TBL. If the complete set of static data are downloaded from the WRF download page, possible resolutions include '30s', '2m', '5m', and '10m', corresponding to 30-arc-second data, 2-, 5-, and 10-arc-minute data.

Depending on the value of the `wrf_core` namelist variable, the appropriate GEOGRID.TBL file must be used with geogrid, since the grid staggerings that WPS interpolates to differ between dynamical cores. For the ARW, the GEOGRID.TBL.ARW file should be used, and for the NMM, the GEOGRID.TBL.NMM file should be used. Selection of the appropriate GEOGRID.TBL is accomplished by linking the correct file to GEOGRID.TBL in the geogrid directory (or in the directory specified by `opt_geogrid_tbl_path`, if this variable is set in the namelist).

```
> ls geogrid/GEOGRID.TBL
lrwxrwxrwx 1      15 GEOGRID.TBL -> GEOGRID.TBL.ARW
```

For more details on the meaning and possible values for each variable, the user is referred to a description of the namelist variables.

Having suitably defined the simulation coarse domain and nested domains in the namelist.wps file, the geogrid.exe executable may be run to produce domain files. In the case of ARW domains, the domain files are named `geo_em.d0N.nc`, where `N` is the number of the nest defined in each file. When run for NMM domains, geogrid produces the file `geo_nmm.d01.nc` for the coarse domain, and `geo_nmm_nest.l0N.nc` files for each nesting level `N`. Also, note that the file suffix will vary depending on the `io_form_geogrid` that is selected. To run geogrid, issue the following command:

```
> ./geogrid.exe
```

When geogrid.exe has finished running, the message

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Successful completion of geogrid.         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

should be printed, and a listing of the WPS root directory (or the directory specified by `opt_output_from_geogrid_path`, if this variable was set) should show the domain files. If not, the geogrid.log file may be consulted in an attempt to determine the possible cause of failure. For more information on checking the output of geogrid, the user is referred to the section on checking WPS output.

```
> ls
drwxr-xr-x 2 4096 arch
-rwxr-xr-x 1 1672 clean
```
Step 2: Extracting meteorological fields from GRIB files with ungrib

Having already downloaded meteorological data in GRIB format, the first step in extracting fields to the intermediate format involves editing the “share” and “ungrib” namelist records of the namelist.wps file – the same file that was edited to define the simulation domains. An example of the two namelist records is given below.

```plaintext
&share
  wrf_core = 'ARW',
  max_dom = 2,
  start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
  end_date   = '2008-03-24_18:00:00','2008-03-24_12:00:00',
  interval_seconds = 21600,
  io_form_geogrid = 2
/

&ungrib
  out_format = 'WPS',
  prefix     = 'FILE'
/
```

In the “share” namelist record, the variables that are of relevance to ungrib are the starting and ending times of the coarse domain (start_date and end_date; alternatively, start_year, start_month, start_day, start_hour, end_year, end_month, end_day, and end_hour) and the interval between meteorological data files (interval_seconds). In the “ungrib” namelist record, the variable out_format is used to select the format of the intermediate data to be written by ungrib; the metgrid program can read any of the formats supported by ungrib, and thus, any of ‘WPS’, ‘SI’, and ‘MM5’ may be specified for out_format, although ‘WPS’ is recommended. Also in the "ungrib" namelist, the user may specify a path and prefix for the intermediate files with the prefix variable. For
example, if prefix were set to 'ARGRMET', then the intermediate files created by ungrib would be named according to AGRMET: YYYY-MM-DD_HH, where YYYY-MM-DD_HH is the valid time of the data in the file.

After suitably modifying the namelist.wps file, a Vtable must be supplied, and the GRIB files must be linked (or copied) to the filenames that are expected by ungrib. The WPS is supplied with Vtable files for many sources of meteorological data, and the appropriate Vtable may simply be symbolically linked to the file Vtable, which is the Vtable name expected by ungrib. For example, if the GRIB data are from the GFS model, this could be accomplished with

> ln -s ungrib/Variable_Tables/Vtable.GFS Vtable

The ungrib program will try to read GRIB files named GRIBFILE.AAA, GRIBFILE.AAB, ..., GRIBFILE.ZZZ. In order to simplify the work of linking the GRIB files to these filenames, a shell script, link_grib.csh, is provided. The link_grib.csh script takes as a command-line argument a list of the GRIB files to be linked. For example, if the GRIB data were downloaded to the directory /data/gfs, the files could be linked with link_grib.csh as follows:

> ls /data/gfs
-rw-r--r-- 1 42728372 gfs_080324_12_00
-rw-r--r-- 1 48218303 gfs_080324_12_06

> ./link_grib.csh /data/gfs/gfs*

After linking the GRIB files and Vtable, a listing of the WPS directory should look something like the following:

> ls
drwxr-xr-x 2 4096 arch
-rwxr-xr-x 1 1672 clean
-rwxr-xr-x 1 3510 compile
-rwxr-xr-x 1 85973 compile.output
-rwxr-xr-x 1 4257 configure
-rwxr-xr-x 1 2486 configure.wps
-rwxr-xr-x 1 1957004 geo_em.d01.nc
-rwxr-xr-x 1 4745324 geo_em.d02.nc
drwxr-xr-x 4 4096 geogrid
lrwxrwxrwx 1 23 geogrid.exe -> geogrid/src/geogrid.exe
-rwxr-xr-x 1 11169 geogrid.log
lrwxrwxrwx 1 38 GRIBFILE.AAA -> /data/gfs/gfs_080324_12_00
lrwxrwxrwx 1 38 GRIBFILE.AAB -> /data/gfs/gfs_080324_12_06
-rwxr-xr-x 1 1328 link_grib.csh
drwxr-xr-x 3 4096 metgrid
lrwxrwxrwx 1 23 metgrid.exe -> metgrid/src/metgrid.exe
-rwxr-xr-x 1 1094 namelist.wps
-rwxr-xr-x 1 1987 namelist.wps.all_options
-rwxr-xr-x 1 1075 namelist.wps.global
-rwxr-xr-x 1 652 namelist.wps.nmm
-rwxr-xr-x 1 4786 README
drwxr-xr-x 4 4096 ungrib
lrwxrwxrwx 1 21 ungrib.exe -> ungrib/src/ungrib.exe
After editing the namelist.wps file and linking the appropriate Vtable and GRIB files, the ungrb.exe executable may be run to produce files of meteorological data in the intermediate format. Ungrib may be run by simply typing the following:

```plaintext
> ./ungrib.exe >& ungrib.output
```

Since the ungrb program may produce a significant volume of output, it is recommended that ungrb output be redirected to a file, as in the command above. If ungrb.exe runs successfully, the message

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Successful completion of ungrb.        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

will be written to the end of the ungrb.output file, and the intermediate files should appear in the current working directory. The intermediate files written by ungrb will have names of the form `FILE:YYYY-MM-DD_HH` (unless, of course, the prefix variable was set to a prefix other than 'FILE').

```plaintext
ls
```

```
drwxr-xr-x 2  4096 arch
-rw-r--r-- 1  1672 clean
-rwxr-xr-x 1  3510 compile
-rw-r--r-- 1  85973 compile.output
-rwxr-xr-x 1  4257 configure
-rw-r--r-- 1  2486 configure.wps
-rw-r--r-- 1 154946888 FILE:2008-03-24_12
-rw-r--r-- 1 154946888 FILE:2008-03-24_18
-rw-r--r-- 1  1957004 geo_em.d01.nc
-rw-r--r-- 1  4745324 geo_em.d02.nc
-rw-r--r-- 4  11169 geogrid.log
-rw-r--r-- 1   1094 namelist.wps
-rw-r--r-- 1   1075 namelist.wps.nmm
-rw-r--r-- 1      652 namelist.wps.global
-rw-r--r-- 1  1987 namelist.wps.all_options
```

```
/data/gfs/gfs_080324_12_00
```

```
-rw-r--r-- 1   1328 link_grib.csh
-rw-r--r-- 3  4096 metgrid
-rw-r--r-- 1   1094 namelist.wps
-rw-r--r-- 1  1987 namelist.wps.all_options
-rw-r--r-- 1  1075 namelist.wps.global
-rw-r--r-- 1    652 namelist.wps.nmm
-rw-r--r-- 1    4786 README
```

```
/data/gfs/gfs_080324_12_06
```

```
-rw-r--r-- 1   1328 link_grib.csh
-rw-r--r-- 3  4096 metgrid
-rw-r--r-- 1   1094 namelist.wps
-rw-r--r-- 1  1987 namelist.wps.all_options
-rw-r--r-- 1  1075 namelist.wps.global
-rw-r--r-- 1    652 namelist.wps.nmm
-rw-r--r-- 1    4786 README
```

```
```
```
Step 3: Horizontally interpolating meteorological data with metgrid

In the final step of running the WPS, meteorological data extracted by ungrib are horizontally interpolated to the simulation grids defined by geogrid. In order to run metgrid, the namelist.wps file must be edited. In particular, the “share” and “metgrid” namelist records are of relevance to the metgrid program. Examples of these records are shown below.

```
&share
    wrf_core = 'ARW',
    max_dom = 2,
    start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
    end_date = '2008-03-24_18:00:00','2008-03-24_12:00:00',
    interval_seconds = 21600,
    io_form_geogrid = 2
/

&metgrid
    fg_name                      = 'FILE',
    io_form_metgrid              = 2,
/
```

By this point, there is generally no need to change any of the variables in the “share” namelist record, since those variables should have been suitably set in previous steps. If the "share" namelist was not edited while running geogrid and ungrib, however, the WRF dynamical core, number of domains, starting and ending times, interval between meteorological data, and path to the static domain files must be set in the “share” namelist record, as described in the steps to run geogrid and ungrib.

In the “metgrid” namelist record, the path and prefix of the intermediate meteorological data files must be given with \texttt{fg\_name}, the full path and file names of any intermediate files containing constant fields may be specified with the \texttt{constants\_name} variable, and the output format for the horizontally interpolated files may be specified with the \texttt{io\_form\_metgrid} variable. Other variables in the “metgrid” namelist record, namely, \texttt{opt\_output\_from\_metgrid\_path} and \texttt{opt\_metgrid\_tbl\_path}, allow the user to specify where interpolated data files should be written by metgrid and where the METGRID.TBL file may be found.

As with geogrid and the GEOGRID.TBL file, a METGRID.TBL file appropriate for the WRF core must be linked in the metgrid directory (or in the directory specified by \texttt{opt\_metgrid\_tbl\_path}, if this variable is set).

```
> ls metgrid/METGRID.TBL
```

```
1rwxrwxrwx 1 15 METGRID.TBL -> METGRID.TBL.ARW
```
After suitably editing the namelist.wps file and verifying that the correct METGRID.TBL will be used, metgrid may be run by issuing the command

```bash
> ./metgrid.exe
```

If metgrid successfully ran, the message

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Successful completion of metgrid.       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

will be printed. After successfully running, metgrid output files should appear in the WPS root directory (or in the directory specified by `opt_output_from_metgrid_path`, if this variable was set). These files will be named `met_em.d0N.YYYY-MM-DD_HH:mm:ss.nc` in the case of ARW domains, where `N` is the number of the nest whose data reside in the file, or `met_nmm.d01.YYYY-MM-DD_HH:mm:ss.nc` in the case of NMM domains. Here, `YYYY-MM-DD_HH:mm:ss` refers to the date of the interpolated data in each file. If these files do not exist for each of the times in the range given in the “share” namelist record, the metgrid.log file may be consulted to help in determining the problem in running metgrid.

```bash
> ls
```

```bash
drwxr-xr-x 2 4096 arch
-rwxr-xr-x 1 1672 clean
-rwxr-xr-x 1 3510 compile
-rw-r--r-- 1 85973 compile.output
-rwxr-xr-x 1 4257 configure
-rw-r--r-- 1 2486 configure.wps
-rw-r--r-- 1 15494688 FILE:2008-03-24_12
-rw-r--r-- 1 15494688 FILE:2008-03-24_18
-rw-r--r-- 1 1957004 geo_em.d01.nc
drwxr-xr-x 4 4096 geogrid
lrwxrwxrwx 1 23 geogrid.exe -> geogrid/src/geogrid.exe
-rw-r--r-- 1 11169 geogrid.log
lrwxrwxrwx 1 38 GRIBFILE.AAA ->
/data/gfs/gfs_080324_12_00
lrwxrwxrwx 1 38 GRIBFILE.AAB ->
```

```bash
/data/gfs/gfs_080324_12_06
```

```bash
-rw-r-xr-x 1 1328 link_grib.csh
-rw-r--r-- 1 5217648 met_em.d01.2008-03-24_12:00:00.nc
-rw-r--r-- 1 5217648 met_em.d01.2008-03-24_18:00:00.nc
-rw-r--r-- 1 12658200 met_em.d02.2008-03-24_12:00:00.nc
drwxr-xr-x 3 4096 metgrid
lrwxrwxrwx 1 23 metgrid.exe -> metgrid/src/metgrid.exe
-rw-r--r-- 1 65970 metgrid.log
-rw-r--r-- 1 1094 namelist.wps
-rw-r--r-- 1 1987 namelist.wps.all_options
-rw-r--r-- 1 1075 namelist.wps.global
-rw-r--r-- 1 652 namelist.wps.nmm
-rw-r--r-- 1 4786 README
drwxr-xr-x 4 4096 ungrib
lrwxrwxrwx 1 21 ungrib.exe -> ungrib/src/ungrib.exe
-rw-r--r-- 1 1418 ungrib.log
```
Creating Nested Domains with the WPS

To run the WPS for nested-domain simulations is essentially no more difficult than running for a single-domain case; the difference with nested-domain simulations is that the geogrid and metgrid programs process more than one grid when they are run, rather than a single grid for the simulation. In order to specify the size and location of nests, a number of variables in the namelist.wps file must be given lists of values, one value per nest.

```
&share
  wrf_core = 'ARW',
  max_dom = 2,
  start_date = '2008-03-24_12:00:00','2008-03-24_12:00:00',
  end_date = '2008-03-24_18:00:00','2008-03-24_12:00:00',
  interval_seconds = 21600,
  io_form_geogrid = 2
/

&geogrid
  parent_id         =   1,   1,
  parent_grid_ratio =   1,   3,
  i_parent_start    =   1,  31,
  j_parent_start    =   1,  17,
  e_we              =  74, 112,
  e_sn              =  61,  97,
  geog_data_res     = 'default','default',
  dx = 30000,
  dy = 30000,
  map_proj = 'lambert',
  ref_lat   = 34.83,
  ref_lon   = -81.03,
  truelat1  = 30.0,
  truelat2  = 60.0,
  stand_lon = -98.
  geog_data_path = '/mmm/users/wrfhelp/WPS_GEOG/'
/
```

The namelist variables that are affected by nests are shown in the (partial) namelist records above. The example shows namelist variables for a two-domain run (the coarse domain plus a single nest), and the effect on the namelist variables generalize to multiple nests in the obvious way: rather than specifying lists of two values, lists of \( N \) values must be specified, where \( N \) is the total number of model grids.

In the above example, the first change to the “share” namelist record is to the `max_dom` variable, which must be set to the total number of nests in the simulation, including the
coarse domain. Having determined the number of nests, all of the other affected namelist variables must be given a list of \( N \) values, one for each grid. The only other change to the “share” namelist record is to the starting and ending times. Here, a starting and ending time must be given for each nest, with the restriction that a nest cannot begin before its parent domain or end after its parent domain; also, it is suggested that nests be given starting and ending times that are identical to the desired starting times of the nest \textit{when running WPS}. This is because the nests get their lateral boundary conditions from their parent domain, and thus, only the initial time for a nest needs to be processed by WPS, except when grid nudging, also called analysis nudging, is used in WRF. It is important to note that, \textit{when running WRF}, the actual starting and ending times for all nests must be given in the WRF namelist.input file.

The remaining changes are to the “geogrid” namelist record. In this record, the parent of each nest must be specified with the \texttt{parent_id} variable. Every nest must be a child of exactly one other nest, with the coarse domain being its own parent. Related to the identity of a nest's parent is the nest refinement ratio with respect to its parent, which is given by the \texttt{parent_grid_ratio} variable; this ratio determines the nominal grid spacing for a nest in relation to the grid spacing of the its parent.

Next, the lower-left corner of a nest is specified as an \((i, j)\) location in the nest’s parent domain; this is done through the \texttt{i_parent_start} and \texttt{j_parent_start} variables, and the specified location is given with respect to the unstaggered grid. Finally, the dimensions of each nest, in grid points, are given for each nest using the \texttt{s_we}, \texttt{e_we}, \texttt{s_sn}, and \texttt{e_sn} variables. The nesting setup in our example namelist is illustrated in the figure above, where it may be seen how each of the above-mentioned variables is determined. Currently, the starting grid point values in the south-north (\texttt{s_sn}) and west-east (\texttt{s_we}) directions must be specified as 1, and the ending grid point values (\texttt{e_sn} and \texttt{e_we}) determine, essentially, the full dimensions of the nest; to ensure that the upper-right corner of the nest's grid is coincident with an unstaggered grid point in the parent domain, both \texttt{e_we} and \texttt{e_sn} must be one greater than some integer multiple of the nesting ratio. Also, for each nest, the resolution (or list or resolutions; see the \texttt{description}}
of namelist variables) of source data to interpolate from is specified with the `geog_data_res` variable. For a complete description of these namelist variables, the user is referred to the description of namelist variables.

Selecting Between USGS and MODIS-based Land Use Classifications

By default, the geogrid program will interpolate land use categories from MODIS IGBP 21-category data. However, the user may select an alternative set of land use categories based on the USGS land-cover classification. Although the MODIS-based data contain 21 categories of land use, these categories are not a subset of the 24 USGS categories; users interested in the specific categories in either data set can find a listing of the land use classes in the section on land use and soil categories.

The 24-category USGS-based land use data may be selected instead of the MODIS data at run-time through the `geog_data_res` variable in the `&geogrid` namelist record. This is accomplished by prefixing each resolution of static data with the string “usgs_30s+”. For example, in a two-domain configuration, where the `geog_data_res` variable would ordinarily be specified as

```
`geog_data_res = 'default', 'default',`
```

the user should instead specify

```
`geog_data_res = 'usgs_30s+default', 'usgs_30s+default',`
```

The effect of this change is to instruct the geogrid program to look, in each entry of the GEOGRID.TBL file, for a resolution of static data with a resolution denoted by ‘usgs_30s’, and if such a resolution is not available, to instead look for a resolution denoted by the string following the ‘+’. Thus, for the GEOGRID.TBL entry for the LANDUSEF field, the USGS-based land use data, which is identified with the string ‘usgs_30s’, would be used instead of the ‘default’, resolutions (or source) of land-use data in the example above; for all other fields, the ‘default’ resolutions would be used for the first and second. As an aside, when none of the resolutions specified for a domain in `geog_data_res` are found in a GEOGRID.TBL entry, the resolution denoted by ‘default’ will be used.

When changing from the default 21-class MODIS land-use data, the user must also ensure that the `num_land_cat` namelist variable is set correctly in `&physics` namelist record in the WRF namelist.input file. For 24-class USGS data, `num_land_cat` should be set to 24.
Selecting Static Data for the Gravity Wave Drag Scheme

The gravity wave drag by orography (GWDO) scheme in the ARW requires ten static fields from the WPS. In fact, these fields will be interpolated by the geogrid program regardless of whether the GWDO scheme will be used in the model. When the GWDO scheme will not be used, the fields will simply be ignored in WRF, and the user need not be concerned with the resolution of data from which the fields are interpolated. However, it is recommended that these fields be interpolated from a resolution of source data that is slightly lower (i.e., coarser) in resolution than the model grid; consequently, if the GWDO scheme will be used, care should be taken to select an appropriate resolution of GWDO static data. Currently, five resolutions of GWDO static data are available: 2-degree, 1-degree, 30-minute, 20-minute, and 10-minute, denoted by the strings ‘2deg’, ‘1deg’, ‘30m’, ‘20m’, and ‘10m’, respectively. To select the resolution to interpolate from, the user should prefix the resolution specified for the \texttt{geog\_data\_res} variable in the “geogrid” namelist record by the string “XXX+”, where XXX is one of the five available resolutions of GWDO static data. For example, in a model configuration with a 48-km grid spacing, the \texttt{geog\_data\_res} variable might typically be specified as

\begin{verbatim}
    geog_data_res = '10m',
\end{verbatim}

However, if the GWDO scheme were employed, the finest resolution of GWDO static data that is still lower in resolution than the model grid would be the 30-minute data, in which case the user should specify

\begin{verbatim}
    geog_data_res = '30m+10m',
\end{verbatim}

If none of ‘2deg’, ‘1deg’, ‘30m’, or ‘20m’ are specified in combination with other resolutions of static data in the \texttt{geog\_data\_res} variable, the ‘10m’ GWDO static data will be used, since it is also designated as the ‘default’ resolution in the GEOGRID.TBL file. It is worth noting that, if 10-minute resolution GWDO data are to be used, but a different resolution is desired for other static fields (e.g., topography height), the user should simply omit ‘10m’ from the value given to the \texttt{geog\_data\_res} variable, since specifying

\begin{verbatim}
    geog_data_res = '10m+30s',
\end{verbatim}

for example, would cause geogrid to use the 10-minute data in preference to the 30-second data for the non-GWDO fields, such as topography height and land use category, as well as for the GWDO fields.

Using Multiple Meteorological Data Sources

The metgrid program is capable of interpolating time-invariant fields, and it can also interpolate from multiple sources of meteorological data. The first of these capabilities uses the \texttt{constants\_name} variable in the \&metgrid namelist record. This variable may
be set to a list of filenames – including path information where necessary – of intermediate-formatted files which contains time-invariant fields, and which should be used in the output for every time period processed by metgrid. For example, short simulations may use a constant SST field; this field need only be available at a single time, and may be used by setting the constants_name variable to the path and filename of the SST intermediate file. Typical uses of constants_name might look like

```plaintext
&metgrid  
  constants_name = '/data/ungribbed/const sharp/SST_FILE:2006-08-16_12'
/
```

or

```plaintext
&metgrid  
  constants_name = 'LANDSEA', 'SOILHGT'
/
```

The second metgrid capability – that of interpolating data from multiple sources – may be useful in situations where two or more complementary data sets need to be combined to produce the full input data needed by real.exe. To interpolate from multiple sources of time-varying, meteorological data, the fg_name variable in the &metgrid namelist record should be set to a list of prefixes of intermediate files, including path information when necessary. When multiple path-prefixes are given, and the same meteorological field is available from more than one of the sources, data from the last-specified source will take priority over all preceding sources. Thus, data sources may be prioritized by the order in which the sources are given.

As an example of this capability, if surface fields are given in one data source and upper-air data are given in another, the values assigned to the fg_name variable may look something like:

```plaintext
&metgrid  
  fg_name = '/data/ungribbed/SFC', '/data/ungribbed/UPPER_AIR'
/
```

To simplify the process of extracting fields from GRIB files, the prefix namelist variable in the &ungrib record may be employed. This variable allows the user to control the names of (and paths to) the intermediate files that are created by ungrrib. The utility of this namelist variable is most easily illustrated by way of an example. Suppose we wish to work with the North American Regional Reanalysis (NARR) data set, which is split into separate GRIB files for 3-dimensional atmospheric data, surface data, and fixed-field data. We may begin by linking all of the "3D" GRIB files using the link_grib.csh script, and by linking the NARR Vtable to the filename vtable. Then, we may suitably edit the &ungrib namelist record before running ungrrib.exe so that the resulting intermediate files have an appropriate prefix:

```plaintext
&ungrib
```
out_format = 'WPS',
prefix = 'NARR_3D',
/

After running ungrib.exe, the following files should exist (with a suitable substitution for the appropriate dates):

NARR_3D:2008-08-16_12
NARR_3D:2008-08-16_15
NARR_3D:2008-08-16_18
...

Given intermediate files for the 3-dimensional fields, we may process the surface fields by linking the surface GRIB files and changing the prefix variable in the namelist:

&ungrib
  out_format = 'WPS',
  prefix = 'NARR_SFC',
  /

Again running ungrib.exe, the following should exist in addition to the NARR_3D files:

NARR_SFC:2008-08-16_12
NARR_SFC:2008-08-16_15
NARR_SFC:2008-08-16_18
...

Finally, the fixed file is linked with the link_grib.csh script, and the prefix variable in the namelist is again set:

&ungrib
  out_format = 'WPS',
  prefix = 'NARR_FIXED',
  /

Having run ungrib.exe for the third time, the fixed fields should be available in addition to the surface and "3D" fields:

NARR_FIXED:1979-11-08_00

For the sake of clarity, the fixed file may be renamed to remove any date information, for example, by renaming it to simply NARR_FIXED, since the fields in the file are static. In this example, we note that the NARR fixed data are only available at a specific time, 1979 November 08 at 0000 UTC, and thus, the user would need to set the correct starting and ending time for the data in the &share namelist record before running ungrib on the NARR fixed file; of course, the times should be re-set before metgrid is run.

Given intermediate files for all three parts of the NARR data set, metgrid.exe may be run after the constants_name and fg_name variables in the &metgrid namelist record are set:
&metgrid
  constants_name = 'NARR_FIXED',
  fg_name = 'NARR_3D', 'NARR_SFC'
/

Although less common, another situation where multiple data sources would be required is when a source of meteorological data from a regional model is insufficient to cover the entire simulation domain, and data from a larger regional model, or a global model, must be used when interpolating to the remaining points of the simulation grid.

For example, to use NAM data wherever possible, and GFS data elsewhere, the following values might be assigned in the namelist:

&metgrid
  fg_name = '/data/ungribbed/GFS', '/data/ungribbed/NAM'
/

Then the resulting model domain would use data as shown in the figure below.

If no field is found in more than one source, then no prioritization need be applied by metgrid, and each field will simply be interpolated as usual; of course, each source should cover the entire simulation domain to avoid areas of missing data.

Using Non-isobaric Meteorological Datasets

When using non-isobaric meteorological datasets to initialize a WRF simulation, it is important that such datasets are supplied to the metgrid.exe program with 3-d pressure and geopotential height fields on the same levels as other 3-d atmospheric variables, such as temperature and humidity. These fields are used by the WRF real.exe pre-processor for vertical interpolation to WRF model levels, for surface pressure computation, and for other purposes.
For some data sources, namely ECMWF model-level data and UK Met Office model data, the 3-d pressure and/or geopotential height fields can be derived from the surface pressure and/or surface height fields using an array of coefficients, and the WPS provides utility programs for performing this derivation; see the section on WPS Utility Programs for more information on the calc_ecmwf_p.exe and height_ukmo.exe programs.

Other meteorological datasets explicitly provide 3-d pressure and geopotential height fields, and the user must only ensure that these fields exist in the set of intermediate files provided to the metgrid.exe program.

**Alternative Initialization of Lake SSTs**

The default treatment of sea-surface temperatures – both for oceans and lakes – in the metgrid program involves simply interpolating the SST field from the intermediate files to all water points in the WRF domain. However, if the lakes that are resolved in the WRF domain are not resolved in the GRIB data, and especially if those lakes are geographically distant from resolved water bodies, the SST field over lakes will most likely be extrapolated from the nearest resolved water bodies in the GRIB data; this situation can lead to lake SST values that are either unrealistically warm or unrealistically cold.

Without a higher-resolution SST field for metgrid to use, one alternative to extrapolating SST values for lakes is to manufacture a “best guess” at the SST for lakes. In the metgrid and real programs, this can be done using a combination of a special land use data set that distinguishes between lakes and oceans, and a field to be used as a proxy for SST over lakes. A special land use data set is necessary, since WRF’s real pre-processing program needs to know where the manufactured SST field should be used instead of the interpolated SST field from the GRIB data.

The alternative procedure for initializing lake SSTs is summarized in the following steps:

1. If they have not already been downloaded (either as a separate tar file or as part of the ‘full’ geographical data tar file), obtain the special land use data sets that distinguish between lakes and oceans. Two such data sets – based on USGS and MODIS land use categories – may be downloaded through the WRF download page. For simplicity, it is recommended to place the two directories in the same directory as the other static geographical data sets (e.g., topo_30s, soiltype_top_30s, etc.) used by geogrid, since doing so will eliminate the need to modify the GEOGRID.TBL file. If the landuse_30s_with_lakes and modis_landuse_21class_30s directories are placed in a location different from the other static data sets, it will be necessary to change the paths to these directories from relative paths to absolute paths in the GEOGRID.TBL file.
2. Before running geogrid, change the specification of `geog_data_res` in the `&geogrid namelist` record to specify either the USGS-based or the MODIS-based land use data with inland water bodies. For example, in a two-domain configuration, setting

```
geog_data_res = 'usgs_lakes+10m', 'usgs_lakes+2m',
```

would tell geogrid to use the USGS-based land use data for both domains, and to use the 10-minute resolution data for other static fields in domain 1 and the 2-minute resolution data for other static fields in domain 2; for MODIS-based data, `usgs_lakes` should be replaced by `modis_lakes`.

Running geogrid should result in output files that use a separate category for inland water bodies instead of the general water category used for oceans and seas. The lake category is identified by the global attribute ISLAKE in the geogrid output files; this attribute should be set to either 28 (in the case of USGS-based data) or 21 (in the case of the MODIS-based data). See, e.g., the list of [WPS output fields](#), where a value of -1 for ISLAKE indicates that there is no separate lake category.

3. After running the ungrib program, use the avg_tsfc.exe utility program to create an intermediate file containing a daily-average surface air temperature field, which will be substituted for the SST field only over lakes by the real program; for more information on the avg_tsfc.exe utility, see the section on [WPS utility programs](#).

4. Before running the metgrid program, add the TAVGSFC file created in the previous step to the specification of `constants_name` in the `&metgrid namelist` record of the namelist.wps file.

5. Run WRF’s real.exe program as usual after setting the number of land categories (`num_land_cat`) in the `&physics namelist` record of the namelist.input file so that it matches the value of the global attribute `NUM_LAND_CAT` in the metgrid files. If the global attribute ISLAKE in the metgrid files indicates that there is a special land use category for lakes, the real program will substitute the TAVGSFC field for the SST field only over those grid points whose category matches the lake category; additionally, the real program will change the land use category of lakes back to the general water category (the category used for oceans), since neither the LANDUSE.TBL nor the VEGPARM.TBL files contain an entry for a lake category.

### Parallelism in the WPS

If the dimensions of the domains to be processed by the WPS become too large to fit in the memory of a single CPU, it is possible to run the geogrid and metgrid programs in a distributed memory configuration. In order to compile geogrid and metgrid for distributed memory execution, the user must have MPI libraries installed on the target machine, and must have compiled WPS using one of the "DM parallel" configuration options. Upon
successful compilation, the geogrid and metgrid programs may be run with the `mpirun` or
`mpiexec` commands, or through a batch queuing system, depending on the machine.

As mentioned earlier, the work of the ungrib program is not amenable to parallelization,
and, further, the memory requirements for ungrib's processing are independent of the
memory requirements of geogrid and metgrid; thus, ungrib is always compiled for a
single processor and run on a single CPU, regardless of whether a "DM parallel"
configuration option was selected during configuration.

Each of the standard WRF I/O API formats (NetCDF, GRIB1, binary) has a
corresponding parallel format, whose number is given by adding 100 to the io_form value
(i.e., the value of `io_form_geogrid` and `io_form_metgrid`) for the standard format. It is
not necessary to use a parallel io_form, but when one is used, each CPU will read/write
its input/output to a separate file, whose name is simply the name that would be used
during serial execution, but with a four-digit processor ID appended to the name. For
example, running geogrid on four processors with `io_form_geogrid` = 102 would create
output files named `geo_em.d01.nc.0000`, `geo_em.d01.nc.0001`, `geo_em.d01.nc.0002`,
and `geo_em.d01.nc.0003` for the coarse domain.

During distributed-memory execution, model domains are decomposed into rectangular
patches, with each processor working on a single patch. When reading/writing from/to
the WRF I/O API format, each processor reads/writes only its patch. Consequently, if a
parallel io_form is chosen for the output of geogrid, metgrid must be run using the same
number of processors as were used to run geogrid. Similarly, if a parallel io_form is
chosen for the metgrid output files, the real program must be run using the same number
of processors. Of course, it is still possible to use a standard io_form when running on
multiple processors, in which case all data for the model domain will be
distributed/collection upon input/output. As a final note, when geogrid or metgrid are run
on multiple processors, each processor will write its own log file, with the log file names
being appended with the same four-digit processor ID numbers that are used for the I/O
API files.

**Checking WPS Output**

When running the WPS, it may be helpful to examine the output produced by the
programs. For example, when determining the location of nests, it may be helpful to see
the interpolated static geographical data and latitude/longitude fields. As another
example, when importing a new source of data into WPS – either static data or
meteorological data – it can often be helpful to check the resulting interpolated fields in
order to make adjustments the interpolation methods used by geogrid or metgrid.

By using the NetCDF format for the geogrid and metgrid I/O forms, a variety of
visualization tools that read NetCDF data may be used to check the domain files
processed by geogrid or the horizontally interpolated meteorological fields produced by
metgrid. In order to set the file format for geogrid and metgrid to NetCDF, the user should specify 2 as the `io_form_geogrid` and `io_form_metgrid` in the WPS namelist file (Note: 2 is the default setting for these options):

```plaintext
&share
  io_form_geogrid = 2,
/
&metgrid
  io_form_metgrid = 2,
/
```

Among the available tools, the ncdump, ncview, and new RIP4 programs may be of interest. The ncdump program is a compact utility distributed with the NetCDF libraries that lists the variables and attributes in a NetCDF file. This can be useful, in particular, for checking the domain parameters (e.g., west-east dimension, south-north dimension, or domain center point) in geogrid domain files, or for listing the fields in a file. The ncview program provides an interactive way to view fields in NetCDF files. Also, for users wishing to produce plots of fields suitable for use in publications, the new release of the RIP4 program may be of interest. The new RIP4 is capable of plotting horizontal contours, map backgrounds, and overlaying multiple fields within the same plot.

Output from the ungrib program is always written in a simple binary format (either ‘WPS’, ‘SI’, or ‘MM5’), so software for viewing NetCDF files will almost certainly be of no use. However, an NCAR Graphics-based utility, `plotfmt`, is supplied with the WPS source code. This utility produces contour plots of the fields found in an intermediate-format file. If the NCAR Graphics libraries are properly installed, the plotfmt program is automatically compiled, along with other utility programs, when WPS is built.

**WPS Utility Programs**

Besides the three main WPS programs – geogrid, ungrib, and metgrid – there are a number of utility programs that come with the WPS, and which are compiled in the util directory. These utilities may be used to examine data files, visualize the location of nested domains, compute pressure fields, and compute average surface temperature fields.

**A. avg_tsfc.exe**

The `avg_tsfc.exe` program computes a daily mean surface temperature given input files in the intermediate format. Based on the range of dates specified in the "share" namelist section of the namelist.wps file, and also considering the interval between intermediate files, `avg_tsfc.exe` will use as many complete days' worth of data as possible in computing the average, beginning at the starting date specified in the namelist. If a complete day's worth of data is not available, no output file will be written, and the program will halt as soon as this can be determined. Similarly, any intermediate files for
dates that cannot be used as part of a complete 24-hour period are ignored; for example, if there are five intermediate files available at a six-hour interval, the last file would be ignored. The computed average field is written to a new file named TAVGSFC using the same intermediate format version as the input files. This daily mean surface temperature field can then be ingested by metgrid by specifying 'TAVGSFC' for the constants_name variable in the "metgrid" namelist section.

B. mod_levs.exe

The mod_levs.exe program is used to remove levels of data from intermediate format files. The levels which are to be kept are specified in a new namelist record in the namelist.wps file:

```
&mod_levs
press_pa = 201300 , 200100 , 100000 ,
          95000 ,  90000 ,
          85000 ,  80000 ,
          75000 ,  70000 ,
          65000 ,  60000 ,
          55000 ,  50000 ,
          45000 ,  40000 ,
          35000 ,  30000 ,
          25000 ,  20000 ,
          15000 ,  10000 ,
           5000 ,   1000
/
```

Within the &mod_levs namelist record, the variable press_pa is used to specify a list of levels to keep; the specified levels should match values of x1v1 in the intermediate format files (see the discussion of the WPS intermediate format for more information on the fields of the intermediate files). The mod_levs program takes two command-line arguments as its input. The first argument is the name of the intermediate file to operate on, and the second argument is the name of the output file to be written.

Removing all but a specified subset of levels from meteorological data sets is particularly useful, for example, when one data set is to be used for the model initial conditions and a second data set is to be used for the lateral boundary conditions. This can be done by providing the initial conditions data set at the first time period to be interpolated by metgrid, and the boundary conditions data set for all other times. If the both data sets have the same number of vertical levels, then no work needs to be done; however, when these two data sets have a different number of levels, it will be necessary, at a minimum, to remove \((m - n)\) levels, where \(m > n\) and \(m\) and \(n\) are the number of levels in each of the two data sets, from the data set with \(m\) levels. The necessity of having the same number of vertical levels in all files is due to a limitation in real.exe, which requires a constant number of vertical levels to interpolate from.

The mod_levs utility is something of a temporary solution to the problem of accommodating two or more data sets with differing numbers of vertical levels. Should a user choose to use mod_levs, it should be noted that, although the vertical locations of the
levels need not match between data sets, all data sets should have a surface level of data, and, when running real.exe and wrf.exe, the value of $p_{\text{top}}$ must be chosen to be below the lowest top among the data sets.

C. calc_ecmwf_p.exe

In the course of vertically interpolating meteorological fields, the real program requires 3-d pressure and geopotential height fields on the same levels as the other atmospheric fields. The calc_ecmwf_p.exe utility may be used to create these fields for use with ECMWF sigma-level data sets. Given a surface pressure field (or log of surface pressure field) and a list of coefficients A and B, calc_ecmwf_p.exe computes the pressure at an ECMWF sigma level $k$ at grid point $(i,j)$ as $P_{ijk} = A_k + B_k*P_{sfcij}$. The list of coefficients used in the pressure computation can be copied from a table appropriate to the number of sigma levels in the data set from one of the following links:

http://www.ecmwf.int/en/forecasts/documentation-and-support/16-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/19-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/31-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/40-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/50-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/60-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/62-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/91-model-levels

http://www.ecmwf.int/en/forecasts/documentation-and-support/137-model-levels

This table should be written in plain text to a file, ecmwf_coeffs, in the current working directory; for example, with 16 sigma levels, the file ecmwf_coeffs would contain something like:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000000</td>
<td>0.000000000</td>
</tr>
<tr>
<td>1</td>
<td>5000.000000</td>
<td>0.000000000</td>
</tr>
<tr>
<td>2</td>
<td>9890.519531</td>
<td>0.001720764</td>
</tr>
<tr>
<td>3</td>
<td>14166.304688</td>
<td>0.013197623</td>
</tr>
<tr>
<td>4</td>
<td>17346.066406</td>
<td>0.042217135</td>
</tr>
<tr>
<td>5</td>
<td>19121.152344</td>
<td>0.093761623</td>
</tr>
<tr>
<td>6</td>
<td>19371.250000</td>
<td>0.169571340</td>
</tr>
<tr>
<td>7</td>
<td>18164.472656</td>
<td>0.268015683</td>
</tr>
<tr>
<td>8</td>
<td>15742.183594</td>
<td>0.384274483</td>
</tr>
<tr>
<td>9</td>
<td>12488.050781</td>
<td>0.510830879</td>
</tr>
<tr>
<td>10</td>
<td>8881.824219</td>
<td>0.638268471</td>
</tr>
<tr>
<td>11</td>
<td>5437.539063</td>
<td>0.756384850</td>
</tr>
</tbody>
</table>
Additionally, if soil height (or soil geopotential), 3-d temperature, and 3-d specific humidity fields are available, calc_ecmwf_p.exe computes a 3-d geopotential height field, which is required to obtain an accurate vertical interpolation in the real program.

Given a set of intermediate files produced by ungrib and the file ecmwf_coeffs, calc_ecmwf_p loops over all time periods in namelist.wps, and produces an additional intermediate file, PRES:YYYY-MM-DD HH, for each time, which contains pressure and geopotential height data for each full sigma level, as well as a 3-d relative humidity field. This intermediate file should be specified to metgrid, along with the intermediate data produced by ungrib, by adding 'PRES' to the list of prefixes in the fg_name namelist variable.

D. height_ukmo.exe

The real program requires 3-d pressure and geopotential height fields to vertically interpolate the output of the metgrid program; however, data sets from the UKMO Unified Model contain a 3-d pressure field, but do not contain a geopotential height field. Accordingly, the height_ukmo.exe program may be used to compute a geopotential height field for data sets from the UKMO Unified Model. The height_ukmo.exe program requires no command-line arguments, but reads the &metgrid namelist record to get the prefix of the intermediate files created by ungrib.exe; the intermediate files indicated by the first prefix in the fg_name variable of the &metgrid namelist record are expected to contain a SOILHGT field, from which the height_ukmo.exe program computes, with the aid of an auxiliary table, the 3-d geopotential height field. The computed height field is written to a new intermediate file with the prefix HGT, and the prefix ‘HGT’ should then be added to the fg_name namelist variable in the &metgrid namelist record before running metgrid.exe. The name of the file containing the auxiliary table is currently hard-wired in the source code of the height_ukmo.exe program, and it is the responsibility of the user to change this file name in WPS/util/src/height_ukmo.F to the name of the table with the same number of levels as the GRIB data processed by ungrib.exe; tables for data with 38, 50, and 70 levels are provided in the WPS/util directory with file names vertical_grid_38_20m_G3.txt, vertical_grid_50_20m_63km.txt, and vertical_grid_70_20m_80km.txt, respectively.

E. plotgrids.ncl

The plotgrids.ncl program is an NCAR Graphics-based utility whose purpose is to plot the locations of all nests defined in the namelist.wps file. The program operates on the namelist.wps file, and thus, may be run without having run any of the three main WPS programs. Upon successful completion, plotgrids produces an Graphics file in the chosen format (see inside the plotgrids.ncl script for making changes to the output format). The coarse domain is drawn to fill the plot frame, a map outline with political boundaries is
drawn over the coarse domain, and any nested domains are drawn as rectangles outlining the extent of each nest. This utility may be useful particularly during initial placement of domains, at which time the user can iteratively adjust the locations of nests by editing the namelist.wps file, running plotgrids.ncl, and determining a set of adjustments to the nest locations. To run this program, simply type 'ncl util/plotgrids.ncl' in the command line from inside the WPS/ directory. Currently, this utility does not work for ARW domains that use the latitude-longitude projection (i.e., when map_proj = 'lat-lon').

**F. g1print.exe**

The g1print.exe program takes as its only command-line argument the name of a GRIB Edition 1 file. The program prints a listing of the fields, levels, and dates of the data in the file.

**G. g2print.exe**

Similar to g1print.exe, the g2print.exe program takes as its only command-line argument the name of a GRIB Edition 2 file. The program prints a listing of the fields, levels, and dates of the data in the file.

**H. rd_intermediate.exe**

Given the name of a singe intermediate format file on the command line, the rd_intermediate.exe program prints information about the fields contained in the file.

**Writing Meteorological Data to the Intermediate Format**

The role of the ungrib program is to decode GRIB data sets into a simple intermediate format that is understood by metgrid. If meteorological data are not available in GRIB Edition 1 or GRIB Edition 2 formats, the user is responsible for writing such data into the intermediate file format. Fortunately, the intermediate format is relatively simple, consisting of a sequence of unformatted Fortran writes. It is important to note that these unformatted writes use big-endian byte order, which can typically be specified with compiler flags. Below, we describe the WPS intermediate format; users interested in the SI or MM5 intermediate formats can first gain familiarity with the WPS format, which is very similar, and later examine the Fortran subroutines that read and write all three intermediate formats (metgrid/src/read_met_module.F and metgrid/src/write_met_module.F, respectively).

When writing data to the WPS intermediate format, 2-dimensional fields are written as a rectangular array of real values. 3-dimensional arrays must be split across the vertical dimension into 2-dimensional arrays, which are written independently. It should also be
noted that, for global data sets, either a Gaussian or cylindrical equidistant projection must be used, and for regional data sets, either a Mercator, Lambert conformal, polar stereographic, or cylindrical equidistant may be used. The sequence of writes used to write a single 2-dimensional array in the WPS intermediate format is as follows (note that not all of the variables declared below are used for a given projection of the data).

integer :: version  ! Format version (must =5 for WPS format)
integer :: nx, ny  ! x- and y-dimensions of 2-d array
integer :: iproj  ! Code for projection of data in array:
                  !  0 = cylindrical equidistant
                  !  1 = Mercator
                  !  3 = Lambert conformal conic
                  !  4 = Gaussian (global only!)
                  !  5 = Polar stereographic
real :: nlats  ! Number of latitudes north of equator
              ! (for Gaussian grids)
real :: xfcst  ! Forecast hour of data
real :: xlvl  ! Vertical level of data in 2-d array
real :: startlat, startlon  ! Lat/lon of point in array indicated by startloc string
real :: deltalat, deltalon  ! Grid spacing, degrees
real :: dx, dy  ! Grid spacing, km
real :: xlonc  ! Standard longitude of projection
real :: truelat1, truelat2  ! True latitudes of projection
real :: earth_radius  ! Earth radius, km
real, dimension(nx,ny) :: slab  ! The 2-d array holding the data
logical :: is_wind_grid_rel  ! Flag indicating whether winds are
                              ! relative to source grid (TRUE) or relative to earth (FALSE)
character (len=8) :: startloc  ! Which point in array is given by startlat/startlon; set either
                               ! to 'SWCORNER' or 'CENTER '
character (len=9) :: field  ! Name of the field
character (len=24) :: hdate  ! Valid date for data YYYY:MM:DD_HH:00:00
character (len=25) :: units  ! Units of data
character (len=32) :: map_source  ! Source model / originating center
character (len=46) :: desc  ! Short description of data

! 1) WRITE FORMAT VERSION
write(unit=ounit) version

! 2) WRITE METADATA
! Cylindrical equidistant
if (iproj == 0) then
  write(unit=ounit) hdate, xfcst, map_source, field, &
                  units, desc, xlvl, nx, ny, iproj
  write(unit=ounit) startloc, startlat, startlon, &
                  deltalat, deltalon, earth_radius
else if (iproj == 1) then
  write(unit=ounit) hdate, xfcst, map_source, field, &
                  units, desc, xlvl, nx, ny, iproj
  write(unit=ounit) startloc, startlat, startlon, dx, dy, &
                  truelat1, earth_radius
Required Meteorological Fields for Running WRF

In order to successfully initialize a WRF simulation, the real.exe pre-processor requires a minimum set of meteorological and land-surface fields to be present in the output from the metgrid.exe program. Accordingly, these required fields must be available in the intermediate files processed by metgrid.exe. The set of required fields is described in the table, below.

<table>
<thead>
<tr>
<th>Field name in intermediate file</th>
<th>Units</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>K</td>
<td>3-d air temperature</td>
<td></td>
</tr>
<tr>
<td>RH</td>
<td>%</td>
<td>3-d relative humidity</td>
<td>Not needed if SPECHUMD is available</td>
</tr>
<tr>
<td>SPECHUMD</td>
<td>kg kg⁻¹</td>
<td>3-d specific humidity</td>
<td>Not needed if RH is available</td>
</tr>
<tr>
<td>UU</td>
<td>m s⁻¹</td>
<td>3-d wind u-component</td>
<td></td>
</tr>
<tr>
<td>VV</td>
<td>m s⁻¹</td>
<td>3-d wind v-component</td>
<td></td>
</tr>
<tr>
<td>GHT</td>
<td>m</td>
<td>3-d geopotential height</td>
<td></td>
</tr>
</tbody>
</table>
### Creating and Editing Vtables

Although Vtables are provided for many common data sets, it would be impossible for ungrib to anticipate every possible source of meteorological data in GRIB format. When a new source of data is to be processed by ungrib.exe, the user may create a new Vtable either from scratch, or by using an existing Vtable as an example. In either case, a basic knowledge of the meaning and use of the various fields of the Vtable will be helpful.

Each Vtable contains either seven or eleven fields, depending on whether the Vtable is for a GRIB Edition 1 data source or a GRIB Edition 2 data source, respectively. The fields of a Vtable fall into one of three categories: fields that describe how the data are identified within the GRIB file, fields that describe how the data are identified by the ungrib and metgrid programs, and fields specific to GRIB Edition 2. Each variable to be extracted by ungrib.exe will have one or more lines in the Vtable, with multiple lines for data that are split among different level types – for example, a surface level and upper-air levels. The fields that must be specified for a line, or entry, in the Vtable depends on the specifics of the field and level.

The first group of fields – those that describe how the data are identified within the GRIB file – are given under the column headings of the Vtable shown below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Unit</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRESSURE</td>
<td>Pa</td>
<td>3-d pressure</td>
<td>Only needed for non-isobaric datasets</td>
</tr>
<tr>
<td>PSFC</td>
<td>Pa</td>
<td>Surface pressure</td>
<td></td>
</tr>
<tr>
<td>PMSL</td>
<td>Pa</td>
<td>Mean sea-level pressure</td>
<td></td>
</tr>
<tr>
<td>SKINTEMP</td>
<td>K</td>
<td>Skin temperature</td>
<td></td>
</tr>
<tr>
<td>SOILHGT</td>
<td>m</td>
<td>Soil height</td>
<td></td>
</tr>
<tr>
<td>TT</td>
<td>K</td>
<td>2-meter air temperature</td>
<td></td>
</tr>
<tr>
<td>RH</td>
<td>%</td>
<td>2-meter relative humidity</td>
<td></td>
</tr>
<tr>
<td>SPECHUMD</td>
<td>kg kg⁻¹</td>
<td>2-meter specific humidity</td>
<td>Not needed if SPECHUMD is available</td>
</tr>
<tr>
<td>UU</td>
<td>m s⁻¹</td>
<td>2-meter wind u-component</td>
<td></td>
</tr>
<tr>
<td>VV</td>
<td>m s⁻¹</td>
<td>2-meter wind v-component</td>
<td></td>
</tr>
<tr>
<td>LANDSEA</td>
<td>fraction</td>
<td>Land-sea mask (0=water, 1=land)</td>
<td></td>
</tr>
<tr>
<td>SMtttbbb</td>
<td>m³ m⁻³</td>
<td>Soil moisture</td>
<td>'ttt' is the layer top depth in cm, and 'bbb' is the layer bottom depth in cm</td>
</tr>
<tr>
<td>STtttbbb</td>
<td>K</td>
<td>Soil temperature</td>
<td></td>
</tr>
<tr>
<td>SOILMmmm</td>
<td>kg m⁻³</td>
<td>Soil moisture</td>
<td>'mmm' is the level depth in cm, not needed if SMtttbbb available</td>
</tr>
<tr>
<td>SOILTmmm</td>
<td>K</td>
<td>Soil temperature</td>
<td></td>
</tr>
</tbody>
</table>
The "GRIB1 Param" field specifies the GRIB code for the meteorological field, which is a number unique to that field within the data set. However, different data sets may use different GRIB codes for the same field – for example, temperature at upper-air levels has GRIB code 11 in GFS data, but GRIB code 130 in ECMWF data. To find the GRIB code for a field, the g1print.exe and g2print.exe utility program may be used.

Given a GRIB code, the "Level Type", "From Level1", and "From Level2" fields are used to specify which levels a field may be found at. As with the "GRIB1 Param" field, the g1print.exe and g2print.exe programs may be used to find values for the level fields. The meanings of the level fields are dependent on the "Level Type" field, and are summarized in the following table.

<table>
<thead>
<tr>
<th>Level</th>
<th>Level Type</th>
<th>From Level1</th>
<th>To Level2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper-air</td>
<td>100</td>
<td>*</td>
<td>(blank)</td>
</tr>
<tr>
<td>Surface</td>
<td>1</td>
<td>0</td>
<td>(blank)</td>
</tr>
<tr>
<td>Sea-level</td>
<td>102</td>
<td>0</td>
<td>(blank)</td>
</tr>
<tr>
<td>Levels at a specified</td>
<td>105</td>
<td>Height, in meters, of the level above ground</td>
<td>(blank)</td>
</tr>
<tr>
<td>height AGL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fields given as layers</td>
<td>112</td>
<td>Starting level for the layer</td>
<td>Ending level for the layer</td>
</tr>
</tbody>
</table>

When layer fields (Level Type 112) are specified, the starting and ending points for the layer have units that are dependent on the field itself; appropriate values may be found with the g1print.exe and g2print.exe utility programs.

The second group of fields in a Vtable, those that describe how the data are identified within the metgrid and real programs, fall under the column headings shown below.

<table>
<thead>
<tr>
<th>metgrid</th>
<th>metgrid</th>
<th>metgrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>

The most important of these three fields is the "metgrid Name" field, which determines the variable name that will be assigned to a meteorological field when it is written to the intermediate files by ungrib. This name should also match an entry in the METGRID.TBL file, so that the metgrid program can determine how the field is to be horizontally interpolated. The "metgrid Units" and "metgrid Description" fields specify the units and a short description for the field, respectively; here, it is important to note
that if no description is given for a field, then *ungrib will not write that field out to the intermediate files.*

The final group of fields, which provide GRIB2-specific information, are found under the column headings below.

```
<table>
<thead>
<tr>
<th>GRIB2</th>
<th>GRIB2</th>
<th>GRIB2</th>
<th>GRIB2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discp</td>
<td>Catgy</td>
<td>Param</td>
<td>Level</td>
</tr>
</tbody>
</table>
```

The GRIB2 fields are only needed in a Vtable that is to be used for GRIB Edition 2 data sets, although having these fields in a Vtable does not prevent that Vtable from also being used for GRIB Edition 1 data. For example, the Vtable.GFS file contains GRIB2 Vtable fields, but is used for both 1-degree (GRIB1) GFS and 0.5-degree (GRIB2) GFS data sets. Since Vtables are provided for most known GRIB Edition 2 data sets, the corresponding Vtable fields are not described here at present.

**Writing Static Data to the Geogrid Binary Format**

The static geographical data sets that are interpolated by the geogrid program are stored as regular 2-d and 3-d arrays written in a simple binary raster format. Users with a new source for a given static field can ingest their data with WPS by writing the data set into this binary format. The geogrid format is capable of supporting single-level and multi-level continuous fields, categorical fields represented as dominant categories, and categorical fields given as fractional fields for each category. The most simple of these field types in terms of representation in the binary format is a categorical field given as a dominant category at each source grid point, an example of which is the 30-second USGS land use data set.

For a categorical field given as dominant categories, the data must first be stored in a regular 2-d array of integers, with each integer giving the dominant category at the
corresponding source grid point. Given this array, the data are written to a file, row-by-row, beginning at the bottom, or southern-most, row. For example, in the figure above, the elements of the $n \times m$ array would be written in the order $x_{11}, x_{12}, \ldots, x_{1m}, x_{21}, \ldots, x_{2m}, \ldots, x_{n1}, \ldots, x_{nm}$. When written to the file, every element is stored as a 1-, 2-, 3-, or 4-byte integer in big-endian byte order (i.e., for the 4-byte integer $ABCD$, byte $A$ is stored at the lowest address and byte $D$ at the highest), although little-endian files may be used by setting `endian=little` in the "index" file for the data set. Every element in a file must use the same number of bytes for its storage, and, of course, it is advantageous to use the fewest number of bytes needed to represent the complete range of values in the array.

When writing the binary data to a file, no header, record marker, or additional bytes should be written. For example, a 2-byte $1000 \times 1000$ array should result in a file whose size is exactly 2,000,000 bytes. Since Fortran unformatted writes add record markers, it is not possible to write a geogrid binary-formatted file directly from Fortran; instead, it is recommended that the C routines in `read_geogrid.c` and `write_geogrid.c` (in the geogrid/src directory) be called when writing data, either from C or Fortran code.

Similar in format to a field of dominant categories is the case of a field of continuous, or real, values. Like dominant-category fields, single-level continuous fields are first organized as a regular 2-d array, then written, row-by-row, to a binary file. However, because a continuous field may contain non-integral or negative values, the storage representation of each element within the file is slightly more complex. All elements in the array must first be converted to integral values. This is done by first scaling all elements by a constant, chosen to maintain the required precision, and then removing any remaining fractional part through rounding. For example, if three decimal places of precision are required, the value -2.71828 would need to be divided by 0.001 and rounded to -2718. Following conversion of all array elements to integral values, if any negative values are found in the array, a second conversion must be applied: if elements are stored using 1 byte each, then $2^8$ is added to each negative element; for storage using 2 bytes, $2^{16}$ is added to each negative element; for storage using 3 bytes, $2^{24}$ is added to each negative element; and for storage using 4 bytes, a value of $2^{32}$ is added to each negative element. It is important to note that no conversion is applied to positive elements. Finally, the resulting positive, integral array is written as in the case of a dominant-category field.

Multi-level continuous fields are handled much the same as single-level continuous fields. For an $n \times m \times r$ array, conversion to a positive, integral field is first performed as described above. Then, each $n \times m$ sub-array is written contiguously to the binary file as before, beginning with the smallest $r$-index. Categorical fields that are given as fractional fields for each possible category can be thought of as multi-level continuous fields, where each level $k$, $1 \leq k \leq r$, is the fractional field for category $k$.

When writing a field to a file in the geogrid binary format, the user should adhere to the naming convention used by the geogrid program, which expects data files to have names of the form $xstart$-$xend$,$ystart$-$yend$, where $xstart$, $xend$, $ystart$, and $yend$ are five-digit positive integers specifying, respectively, the starting $x$-index of the array contained in
the file, the ending $x$-index of the array, the starting $y$-index of the array, and the ending $y$-index of the array; here, indexing begins at 1, rather than 0. So, for example, an $800 \times 1200$ array (i.e., 800 rows and 1200 columns) might be named 00001-01200.00001-00800.

When a data set is given in several pieces, each of the pieces may be formed as a regular rectangular array, and each array may be written to a separate file. In this case, the relative locations of the arrays are determined by the range of $x$- and $y$-indices in the file names for each of the arrays. It is important to note, however, that every tile in a data set must have the same $x$- and $y$-dimensions, and that tiles of data within a data set must not overlap; furthermore, all tiles must start and end on multiples of the index ranges. For example, the global 30-second USGS topography data set is divided into arrays of dimension $1200 \times 1200$, with each array containing a 10-degree $\times$ 10-degree piece of the data set; the file whose south-west corner is located at $(90S, 180W)$ is named 00001-01200.00001-01200, and the file whose north-east corner is located at $(90N, 180E)$ is named 42001-43200.20401-21600.

If a data set is to be split into multiple tiles, and the number of grid points in, say, the $x$-direction is not evenly divided by the number of tiles in the $x$-direction, then the last column of tiles must be padded with a flag value (specified in the index file using the missing_value keyword) so that all tiles have the same dimensions. For example, if a data set has 2456 points in the $x$-direction, and three tiles in the $x$-direction will be used, the range of $x$-coordinates of the tiles might be $1 – 820, 821 – 1640, and 1641 – 2460$, with columns 2457 through 2460 being filled with a flag value.

Clearly, since the starting and ending indices must have five digits, a field cannot have more than 99999 data points in either of the $x$- or $y$-directions. In case a field has more than 99999 data points in either dimension, the user can simply split the data set into several smaller data sets which will be identified separately to geogrid. For example, a very large global data set may be split into data sets for the Eastern and Western hemispheres.

Besides the binary data files, geogrid requires one extra metadata file per data set. This metadata file is always named 'index', and thus, two data sets cannot reside in the same directory. Essentially, this metadata file is the first file that geogrid looks for when processing a data set, and the contents of the file provide geogrid with all of the information necessary for constructing names of possible data files. The contents of an example index file are given below.

```
type = continuous
signed = yes
projection = regular_ll
dx = 0.00833333
dy = 0.00833333
known_x = 1.0
known_y = 1.0
known_lat = -89.99583
known_lon = -179.99583
```
Creating an Urban Fraction Field from NLCD Data

In order to create a more inhomogeneous and detailed urban fraction field for use with NUDAPT, users may obtain high-resolution land cover information from the National Land Cover Database (NLCD) through the Multi-Resolution Land Characteristics Consortium. Generation of the urban fraction field, called FRC_URB2D in WRF, involves first downloading the NLCD data over the region covered by the WRF domain, converting the data into the binary format used by geogrid, and finally extracting only the urban categories to a new urban fraction field. The following steps can serve as a guide through this process.

1. Download NLCD data from http://gisdata.usgs.net/website/MRLC/viewer.php. Either of the 1992, 2001, or 2006 data sets may be used. After selecting an area to download, make sure to select GeoTIFF format in the "Request Summary Page" by clicking on "Modify Data Request". If available, data may instead be downloaded in BIL format, in which case the format conversion described in the next step can be skipped.

2. After downloading the data, unpacking the archive should yield a directory with a .tif file and a .tfw file, among others. In order for the information in the GeoTIFF file to be useful, the .tif image must be converted into the binary format used by the WPS. This conversion can be accomplished using the GDAL translation tool, gdal_translate, (http://gdal.org) by running the command:

   > gdal_translate -of ENVI foo.tif data.bil

where foo.tif is the name of the GeoTIFF image that was downloaded in Step 1. The output format "ENVI" is a simple binary raster format that matches the format used by geogrid. After converting the GeoTIFF to a binary file, the resulting data.bil file must be renamed to 00001-ncols.00001-nrows, where ncols is the number of columns (in i5.5 format) and nrows is the number of rows (also in i5.5 format) in the image; these values should have been printed to the screen when the gdal_translate program was run.

For a complete listing of keywords that may appear in an index file, along with the meaning of each keyword, the user is referred to the section on index file options.

```plaintext
wordsize = 2
tile_x = 1200
tile_y = 1200
tile_z = 1
tile_bdr=3
units="meters MSL"
description="Topography height"
```
3. Use the converter program available from http://www2.mmm.ucar.edu/people/duda/uf/ to extract the urban categories from the binary tile and write a new tile of data containing urban fraction. The output file of this converter should be copied over the original land use tile, i.e., the urban fraction file should be renamed to 00001-ncols.00001-nrows, where ncols is the number of columns (in i5.5 format) and nrows is the number of rows (also in i5.5 format) in the tile, as in Step 2.

4. Create an index metadata file for the urban fraction data. In the directory created by unpacking the land use data, a .tfw file should also exist. The last two lines in this file give the location of the north-west corner of the data tile, which is used in the index file for variables known_lat and known_lon. If this location is given as (x,y) coordinates, in meters, then the coordinate converter utility available from http://www2.mmm.ucar.edu/people/duda/uf/ may be used to convert to (latitude, longitude), which is required by the index file. The basic index file should contain the following elements:

```
type=continuous
projection=albers_nad83
dx=30.0
dy=30.0
known_x=1.0
known_y=2351.0    # <- edit
known_lat = 40.096571  # <- edit
known_lon = -105.405615  # <- edit
truelat1=29.5
truelat2=45.5
stdlon=-96.0
wordsize=1
scale_factor=0.01
row_order=top_bottom
tile_x=2407    # <- edit
tile_y=2351    # <- edit
tile_z=1
units="unitless"
description="urban fraction"
```

5. Add the following entry to the GEOGRID.TBL file before re-running the geogrid.exe program:

```

name=FRC_URB2D
priority=1
dest_type=continuous
fill_missing = 0.
interp_option=default: average_gcell(1.0)+four_pt
abs_path=default:/path/to/dataset/

```

The path to the dataset and index files created in Step 3 and Step 4, respectively, should be substituted for “/path/to/dataset/” in the entry above.
Description of the Namelist Variables

A. SHARE section

This section describes variables that are used by more than one WPS program. For example, the \texttt{wrf\_core} variable specifies whether the WPS is to produce data for the ARW or the NMM core – information which is needed by both the geogrid and metgrid programs.

1. WRF\_CORE : A character string set to either ‘\texttt{ARW}’ or ‘\texttt{NMM}’ that tells the WPS which dynamical core the input data are being prepared for. Default value is ‘\texttt{ARW}’.

2. MAX\_DOM : An integer specifying the total number of domains/nests, including the parent domain, in the simulation. Default value is 1.

3. START\_YEAR : A list of MAX\_DOM 4-digit integers specifying the starting UTC year of the simulation for each nest. No default value.

4. START\_MONTH : A list of MAX\_DOM 2-digit integers specifying the starting UTC month of the simulation for each nest. No default value.

5. START\_DAY : A list of MAX\_DOM 2-digit integers specifying the starting UTC day of the simulation for each nest. No default value.

6. START\_HOUR : A list of MAX\_DOM 2-digit integers specifying the starting UTC hour of the simulation for each nest. No default value.

7. END\_YEAR : A list of MAX\_DOM 4-digit integers specifying the ending UTC year of the simulation for each nest. No default value.

8. END\_MONTH : A list of MAX\_DOM 2-digit integers specifying the ending UTC month of the simulation for each nest. No default value.

9. END\_DAY : A list of MAX\_DOM 2-digit integers specifying the ending UTC day of the simulation for each nest. No default value.

10. END\_HOUR : A list of MAX\_DOM 2-digit integers specifying the ending UTC hour of the simulation for each nest. No default value.

11. START\_DATE : A list of MAX\_DOM character strings of the form ‘YYYY-MM-DD_HH:mm:ss’ specifying the starting UTC date of the simulation for each nest. The \texttt{start\_date} variable is an alternate to specifying \texttt{start\_year}, \texttt{start\_month}, \texttt{start\_day}, and \texttt{start\_hour}, and if both methods are used for specifying the starting time, the \texttt{start\_date} variable will take precedence. No default value.
12. END_DATE : A list of MAX_DOM character strings of the form 'YYYY-MM-DD_HH:mm:ss' specifying the ending UTC date of the simulation for each nest. The end_date variable is an alternate to specifying end_year, end_month, end_day, and end_hour, and if both methods are used for specifying the ending time, the end_date variable will take precedence. No default value.

13. INTERVAL SECONDS : The integer number of seconds between time-varying meteorological input files. No default value.

14. ACTIVE_GRID : A list of MAX_DOM logical values specifying, for each grid, whether that grid should be processed by geogrid and metgrid. Default value is .TRUE..

15. IO_FORM_GEOGRID : The WRF I/O API format that the domain files created by the geogrid program will be written in. Possible options are: 1 for binary; 2 for NetCDF; 3 for GRIB1. When option 1 is given, domain files will have a suffix of .int; when option 2 is given, domain files will have a suffix of .nc; when option 3 is given, domain files will have a suffix of .gr1. Default value is 2 (NetCDF).

16. OPT_OUTPUT_FROM_GEOGRID_PATH : A character string giving the path, either relative or absolute, to the location where output files from geogrid should be written to and read from. Default value is './'.

17. DEBUG_LEVEL : An integer value indicating the extent to which different types of messages should be sent to standard output. When debug_level is set to 0, only generally useful messages and warning messages will be written to standard output. When debug_level is greater than 100, informational messages that provide further runtime details are also written to standard output. Debugging messages and messages specifically intended for log files are never written to standard output, but are always written to the log files. Default value is 0.

B. GEOGRID section

This section specifies variables that are specific to the geogrid program. Variables in the geogrid section primarily define the size and location of all model domains, and where the static geographical data are found.

1. PARENT_ID : A list of MAX_DOM integers specifying, for each nest, the domain number of the nest’s parent; for the coarsest domain, this variable should be set to 1. Default value is 1.

2. PARENT_GRID_RATIO : A list of MAX_DOM integers specifying, for each nest, the nesting ratio relative to the domain’s parent. No default value.
3. I_PARENT_START : A list of MAX_DOM integers specifying, for each nest, the x-coordinate of the lower-left corner of the nest in the parent unstaggered grid. For the coarsest domain, a value of 1 should be specified. No default value.

4. J_PARENT_START : A list of MAX_DOM integers specifying, for each nest, the y-coordinate of the lower-left corner of the nest in the parent unstaggered grid. For the coarsest domain, a value of 1 should be specified. No default value.

5. S_WE : A list of MAX_DOM integers which should all be set to 1. Default value is 1.

6. E_WE : A list of MAX_DOM integers specifying, for each nest, the nest’s full west-east dimension. For nested domains, e_we must be one greater than an integer multiple of the nest's parent_grid_ratio (i.e., e_we = n*parent_grid_ratio + 1 for some positive integer n). No default value.

7. S_SN : A list of MAX_DOM integers which should all be set to 1. Default value is 1.

8. E_SN : A list of MAX_DOM integers specifying, for each nest, the nest’s full south-north dimension. For nested domains, e_sn must be one greater than an integer multiple of the nest's parent_grid_ratio (i.e., e_sn = n*parent_grid_ratio + 1 for some positive integer n). No default value.

9. GEOG_DATA_RES : A list of MAX_DOM character strings specifying, for each nest, a corresponding resolution or list of resolutions separated by + symbols of source data to be used when interpolating static terrestrial data to the nest’s grid. For each nest, this string should contain a resolution matching a string preceding a colon in a rel_path or abs_path specification (see the description of GEOGRID.TBL options) in the GEOGRID.TBL file for each field. If a resolution in the string does not match any such string in a rel_path or abs_path specification for a field in GEOGRID.TBL, a default resolution of data for that field, if one is specified, will be used. If multiple resolutions match, the first resolution to match a string in a rel_path or abs_path specification in the GEOGRID.TBL file will be used. Default value is 'default'.

10. DX : A real value specifying the grid distance in the x-direction where the map scale factor is 1. For ARW, the grid distance is in meters for the 'polar', 'lambert', and 'mercator' projection, and in degrees longitude for the 'lat-lon' projection; for NMM, the grid distance is in degrees longitude. Grid distances for nests are determined recursively based on values specified for parent_grid_ratio and parent_id. No default value.

11. DY : A real value specifying the nominal grid distance in the y-direction where the map scale factor is 1. For ARW, the grid distance is in meters for the 'polar', 'lambert', and 'mercator' projection, and in degrees latitude for the 'lat-lon' projection; for NMM, the grid distance is in degrees latitude. Grid distances for nests are determined recursively based on values specified for parent_grid_ratio and parent_id. No default value.
12. MAP_PROJ : A character string specifying the projection of the simulation domain. For ARW, accepted projections are 'lambert', 'polar', 'mercator', and 'lat-lon'; for NMM, a projection of 'rotated_ll' must be specified. Default value is 'lambert'.

13. REF_LAT : A real value specifying the latitude part of a (latitude, longitude) location whose (i,j) location in the simulation domain is known. For ARW, ref_lat gives the latitude of the center-point of the coarse domain by default (i.e., when ref_x and ref_y are not specified). For NMM, ref_lat always gives the latitude to which the origin is rotated. No default value.

14. REF_LON : A real value specifying the longitude part of a (latitude, longitude) location whose (i, j) location in the simulation domain is known. For ARW, ref_lon gives the longitude of the center-point of the coarse domain by default (i.e., when ref_x and ref_y are not specified). For NMM, ref_lon always gives the longitude to which the origin is rotated. For both ARW and NMM, west longitudes are negative, and the value of ref_lon should be in the range [-180, 180]. No default value.

15. REF_X : A real value specifying the i part of an (i, j) location whose (latitude, longitude) location in the simulation domain is known. The (i, j) location is always given with respect to the mass-staggered grid, whose dimensions are one less than the dimensions of the unstaggered grid. Default value is (((E_WE-1.)+1.)/2.) = (E_WE/2.).

16. REF_Y : A real value specifying the j part of an (i, j) location whose (latitude, longitude) location in the simulation domain is known. The (i, j) location is always given with respect to the mass-staggered grid, whose dimensions are one less than the dimensions of the unstaggered grid. Default value is (((E_SN-1.)+1.)/2.) = (E_SN/2.).

17. TRUELAT1 : A real value specifying, for ARW, the first true latitude for the Lambert conformal projection, or the only true latitude for the Mercator and polar stereographic projections. For NMM, truelat1 is ignored. No default value.

18. TRUELAT2 : A real value specifying, for ARW, the second true latitude for the Lambert conformal conic projection. For all other projections, truelat2 is ignored. No default value.

19. STAND_LON : A real value specifying, for ARW, the longitude that is parallel with the y-axis in the Lambert conformal and polar stereographic projections. For the regular latitude-longitude projection, this value gives the rotation about the earth's geographic poles. For NMM, stand_lon is ignored. No default value.

20. POLE_LAT : For the latitude-longitude projection for ARW, the latitude of the North Pole with respect to the computational latitude-longitude grid in which -90.0° latitude is at the bottom of a global domain, 90.0° latitude is at the top, and 180.0° longitude is at the center. Default value is 90.0.
21. **POLE_LON**: For the latitude-longitude projection for ARW, the longitude of the North Pole with respect to the computational lat/lon grid in which -90.0° latitude is at the bottom of a global domain, 90.0° latitude is at the top, and 180.0° longitude is at the center. Default value is 0.0.

22. **GEOG_DATA_PATH**: A character string giving the path, either relative or absolute, to the directory where the geographical data directories may be found. This path is the one to which `rel_path` specifications in the GEOGRID.TBL file are given in relation to. No default value.

23. **OPT_GEOGRID_TBL_PATH**: A character string giving the path, either relative or absolute, to the GEOGRID.TBL file. The path should not contain the actual file name, as GEOGRID.TBL is assumed, but should only give the path where this file is located. Default value is './geogrid/'.

**C. UNGRIB section**

Currently, this section contains only two variables, which determine the output format written by ungrib and the name of the output files.

1. **OUT_FORMAT**: A character string set either to 'WPS', 'SI', or 'MM5'. If set to 'MM5', ungrib will write output in the format of the MM5 pregrid program; if set to 'SI', ungrib will write output in the format of grib_prep.exe; if set to 'WPS', ungrib will write data in the WPS intermediate format. Default value is 'WPS'.

2. **PREFIX**: A character string that will be used as the prefix for intermediate-format files created by ungrib; here, prefix refers to the string `PREFIX` in the filename `PREFIX:YYYY-MM-DD_HH` of an intermediate file. The prefix may contain path information, either relative or absolute, in which case the intermediate files will be written in the directory specified. This option may be useful to avoid renaming intermediate files if ungrib is to be run on multiple sources of GRIB data. Default value is 'FILE'.

**D. METGRID section**

This section defines variables used only by the metgrid program. Typically, the user will be interested in the `fg_name` variable, and may need to modify other variables of this section less frequently.

1. **FG_NAME**: A list of character strings specifying the path and prefix of ungribbed data files. The path may be relative or absolute, and the prefix should contain all characters of the filenames up to, but not including, the colon preceding the date. When more than one `fg_name` is specified, and the same field is found in two or more input
sources, the data in the last encountered source will take priority over all preceding sources for that field. Default value is an empty list (i.e., no meteorological fields).

2. CONSTANTS_NAME : A list of character strings specifying the path and full filename of ungribbed data files which are time-invariant. The path may be relative or absolute, and the filename should be the complete filename; since the data are assumed to be time-invariant, no date will be appended to the specified filename. Default value is an empty list (i.e., no constant fields).

3. IO_FORM_METGRID : The WRF I/O API format that the output created by the metgrid program will be written in. Possible options are: 1 for binary; 2 for NetCDF; 3 for GRIB1. When option 1 is given, output files will have a suffix of .int; when option 2 is given, output files will have a suffix of .nc; when option 3 is given, output files will have a suffix of .gr1. Default value is 2 (NetCDF).

4. OPT_OUTPUT_FROM_METGRID_PATH : A character string giving the path, either relative or absolute, to the location where output files from metgrid should be written to. The default value is the current working directory (i.e., the default value is './').

5. OPT_METGRID_TBL_PATH : A character string giving the path, either relative or absolute, to the METGRID.TBL file; the path should not contain the actual file name, as METGRID.TBL is assumed, but should only give the path where this file is located. Default value is './metgrid/'.

5. PROCESS_ONLY_BDY: An integer specifying the number of boundary rows and columns to be processed by metgrid for time periods after the initial time; for the initial time, metgrid will always interpolate to every grid point. Setting this option to the intended value of spec_bdy_width in the WRF namelist.input will speed up processing in metgrid, but it should not be set if interpolated data are needed in the domain interior. If this option is set to zero, metgrid will horizontally interpolate meteorological data to every grid point in the model domains. This option is only available for ARW. Default value is 0.

Description of GEOGRID.TBL Options

The GEOGRID.TBL file is a text file that defines parameters of each of the data sets to be interpolated by geogrid. Each data set is defined in a separate section, with sections being delimited by a line of equality symbols (e.g., '='). Within each section, there are specifications, each of which has the form of keyword=value. Some keywords are required in each data set section, while others are optional; some keywords are mutually exclusive with other keywords. Below, the possible keywords and their expected range of values are described.

1. NAME : A character string specifying the name that will be assigned to the interpolated field upon output. No default value.
2. **PRIORITY**: An integer specifying the priority that the data source identified in the table section takes with respect to other sources of data for the same field. If a field has \( n \) sources of data, then there must be \( n \) separate table entries for the field, each of which must be given a unique value for priority in the range \([1, n]\). No default value.

3. **DEST_TYPE**: A character string, either *categorical* or *continuous*, that tells whether the interpolated field from the data source given in the table section is to be treated as a continuous or a categorical field. No default value.

4. **INTERP_OPTION**: A sequence of one or more character strings, which are the names of interpolation methods to be used when horizontally interpolating the field. Available interpolation methods are: `average_4pt`, `average_16pt`, `wt_average_4pt`, `wt_average_16pt`, `nearest_neighbor`, `four_pt`, `sixteen_pt`, `search(r)`, and `average_gcell(r)`. For the search method (`search`), the optional argument \( r \) specifies the maximum search radius in units of grid points in the grid of the source data; the default search radius is 1200 points. For the grid cell average method (`average_gcell`), the optional argument \( r \) specifies the minimum ratio of source data resolution to simulation grid resolution at which the method will be applied; unless specified, \( r = 0.0 \), and the option is used for any ratio. When a sequence of two or more methods are given, the methods should be separated by a `+` sign. No default value.

5. **SMOOTH_OPTION**: A character string giving the name of a smoothing method to be applied to the field after interpolation. Available smoothing options are: `1-2-1`, `smth-desmth`, and `smth-desmth_special` (ARW only). Default value is null (i.e., no smoothing is applied).

6. **SMOOTH_PASSES**: If smoothing is to be performed on the interpolated field, `smooth_passes` specifies an integer number of passes of the smoothing method to apply to the field. Default value is 1.

7. **REL_PATH**: A character string specifying the path relative to the path given in the namelist variable `geog_data_path`. A specification is of the general form `RES_STRING:REL_PATH`, where `RES_STRING` is a character string identifying the source or resolution of the data in some unique way and may be specified in the namelist variable `geog_data_res`, and `REL_PATH` is a path relative to `geog_data_path` where the index and data tiles for the data source are found. More than one `rel_path` specification may be given in a table section if there are multiple sources or resolutions for the data source, just as multiple resolutions may be specified (in a sequence delimited by `+` symbols) for `geog_data_res`. See also `abs_path`. No default value.

8. **ABS_PATH**: A character string specifying the absolute path to the index and data tiles for the data source. A specification is of the general form `RES_STRING:ABS_PATH`, where `RES_STRING` is a character string identifying the source or resolution of the data in some unique way and may be specified in the namelist variable `geog_data_res`, and `ABS_PATH` is the absolute path to the data source's files. More than one `abs_path` specification may be given in a table section if there are multiple sources or resolutions.
for the data source, just as multiple resolutions may be specified (in a sequence delimited by + symbols) for geog_data_res. See also rel_path. No default value.

9. OUTPUT_STAGGER : A character string specifying the grid staggering to which the field is to be interpolated. For ARW domains, possible values are u, v, and h; for NMM domains, possible values are hh and vv. Default value for ARW is h; default value for NMM is hh.

10. LANDMASK_WATER : One or more comma-separated integer values giving the indices of the categories within the field that represents water. When landmask_water is specified in the table section of a field for which dest_type=categorical, the LANDMASK field will be computed from the field using the specified categories as the water categories. The keywords landmask_water and landmask_land are mutually exclusive. Default value is null (i.e., a landmask will not be computed from the field).

11. LANDMASK_LAND : One or more comma-separated integer values giving the indices of the categories within the field that represents land. When landmask_water is specified in the table section of a field for which dest_type=categorical, the LANDMASK field will be computed from the field using the specified categories as the land categories. The keywords landmask_water and landmask_land are mutually exclusive. Default value is null (i.e., a landmask will not be computed from the field).

12. MASKED : Either land or water, indicating that the field is not valid at land or water points, respectively. If the masked keyword is used for a field, those grid points that are of the masked type (land or water) will be assigned the value specified by fill_missing. Default value is null (i.e., the field is not masked).

13. FILL_MISSING : A real value used to fill in any missing or masked grid points in the interpolated field. Default value is 1.E20.

14. HALT_ON_MISSING : Either yes or no, indicating whether geogrid should halt with a fatal message when a missing value is encountered in the interpolated field. Default value is no.

15. DOMINANT_CATEGORY : When specified as a character string, the effect is to cause geogrid to compute the dominant category from the fractional categorical field, and to output the dominant category field with the name specified by the value of dominant_category. This option can only be used for fields with dest_type=categorical. Default value is null (i.e., no dominant category will be computed from the fractional categorical field).

16. DOMINANT_ONLY : When specified as a character string, the effect is similar to that of the dominant_category keyword: geogrid will compute the dominant category from the fractional categorical field and output the dominant category field with the name specified by the value of dominant_only. Unlike with dominant_category, though, when dominant_only is used, the fractional categorical field will not appear in the
geogrid output. This option can only be used for fields with dest_type=categorical. Default value is null (i.e., no dominant category will be computed from the fractional categorical field).

17. DF_DX : When df_dx is assigned a character string value, the effect is to cause geogrid to compute the directional derivative of the field in the x-direction using a central difference along the interior of the domain, or a one-sided difference at the boundary of the domain; the derivative field will be named according to the character string assigned to the keyword df_dx. Default value is null (i.e., no derivative field is computed).

18. DF_DY : When df_dy is assigned a character string value, the effect is to cause geogrid to compute the directional derivative of the field in the y-direction using a central difference along the interior of the domain, or a one-sided difference at the boundary of the domain; the derivative field will be named according to the character string assigned to the keyword df_dy. Default value is null (i.e., no derivative field is computed).

19. Z_DIM_NAME : For 3-dimensional output fields, a character string giving the name of the vertical dimension, or z-dimension. A continuous field may have multiple levels, and thus be a 3-dimensional field, and a categorical field may take the form of a 3-dimensional field if it is written out as fractional fields for each category. No default value.

20. FLAG_IN_OUTPUT : A character string giving the name of a global attribute which will be assigned a value of 1 and written to the geogrid output. Default value is null (i.e., no flag will be written for the field).

21. OPTIONAL : Either yes or no, indicating whether the dataset identified by the resolution specified in the geo_data_res namelist option is optional. If an entry in the GEOGRID.TBL file is optional and if the specified resolution of data cannot be read, geogrid will print an informational message indicating that the dataset was not interpolated and continue; otherwise, if the entry is not optional and the specified resolution of data cannot be read, geogrid will halt with an error. It is possible for different priority level entries for the same field to specify different values of the optional keyword, e.g., the priority=2 entry for a field can be optional, while the priority=1 entry can be non-optional (i.e., optional=no). Default value is no.

Description of index Options

Related to the GEOGRID.TBL are the index files that are associated with each static data set. An index file defines parameters specific to that data set, while the GEOGRID.TBL file describes how each of the data sets should be treated by geogrid. As with the GEOGRID.TBL file, specifications in an index file are of the form keyword=value. Below are possible keywords and their possible values.
1. **PROJECTION** : A character string specifying the projection of the data, which may be either lambert, polar, mercator, regular_ll, albers_nad83, or polar_wgs84. No default value.

2. **TYPE** : A character string, either categorical or continuous, that determines whether the data in the data files should be interpreted as a continuous field or as discrete indices. For categorical data represented by a fractional field for each possible category, type should be set to continuous. No default value.

3. **SIGNED** : Either yes or no, indicating whether the values in the data files (which are always represented as integers) are signed in two's complement form or not. Default value is no.

4. **UNITS** : A character string, enclosed in quotation marks ("), specifying the units of the interpolated field; the string will be written to the geogrid output files as a variable time-independent attribute. No default value.

5. **DESCRIPTION** : A character string, enclosed in quotation marks ("), giving a short description of the interpolated field; the string will be written to the geogrid output files as a variable time-independent attribute. No default value.

6. **DX** : A real value giving the grid spacing in the x-direction of the data set. If projection is one of lambert, polar, mercator, albers_nad83, or polar_wgs84, dx gives the grid spacing in meters; if projection is regular_ll, dx gives the grid spacing in degrees. No default value.

7. **DY** : A real value giving the grid spacing in the y-direction of the data set. If projection is one of lambert, polar, mercator, albers_nad83, or polar_wgs84, dy gives the grid spacing in meters; if projection is regular_ll, dy gives the grid spacing in degrees. No default value.

8. **KNOWN_X** : A real value specifying the i-coordinate of an (i,j) location corresponding to a (latitude, longitude) location that is known in the projection. Default value is 1.

9. **KNOWN_Y** : A real value specifying the j-coordinate of an (i,j) location corresponding to a (latitude, longitude) location that is known in the projection. Default value is 1.

10. **KNOWN_LAT** : A real value specifying the latitude of a (latitude, longitude) location that is known in the projection. No default value.

11. **KNOWN_LON** : A real value specifying the longitude of a (latitude, longitude) location that is known in the projection. No default value.
12. STDNON : A real value specifying the longitude that is parallel with the y-axis in conic and azimuthal projections. No default value.

13. TRUELAT1 : A real value specifying the first true latitude for conic projections or the only true latitude for azimuthal projections. No default value.

14. TRUELAT2 : A real value specifying the second true latitude for conic projections. No default value.

15. WORDSIZE : An integer giving the number of bytes used to represent the value of each grid point in the data files. No default value.

16. TILE_X : An integer specifying the number of grid points in the x-direction, excluding any halo points, for a single tile of source data. No default value.

17. TILE_Y : An integer specifying the number of grid points in the y-direction, excluding any halo points, for a single tile of source data. No default value.

18. TILE_Z : An integer specifying the number of grid points in the z-direction for a single tile of source data; this keyword serves as an alternative to the pair of keywords tile_z_start and tile_z_end, and when this keyword is used, the starting z-index is assumed to be 1. No default value.

19. TILE_Z_START : An integer specifying the starting index in the z-direction of the array in the data files. If this keyword is used, tile_z_end must also be specified. No default value.

20. TILE_Z_END : An integer specifying the ending index in the z-direction of the array in the data files. If this keyword is used, tile_z_start must also be specified. No default value.

21. CATEGORY_MIN : For categorical data (type=categorical), an integer specifying the minimum category index that is found in the data set. If this keyword is used, category_max must also be specified. No default value.

22. CATEGORY_MAX : For categorical data (type=categorical), an integer specifying the maximum category index that is found in the data set. If this keyword is used, category_min must also be specified. No default value.

23. TILE_BDR : An integer specifying the halo width, in grid points, for each tile of data. Default value is 0.

24. MISSING_VALUE : A real value that, when encountered in the data set, should be interpreted as missing data. No default value.
25. SCALE_FACTOR : A real value that data should be scaled by (through multiplication) after being read in as integers from tiles of the data set. Default value is 1.

26. ROW_ORDER : A character string, either bottom_top or top_bottom, specifying whether the rows of the data set arrays were written proceeding from the lowest-index row to the highest (bottom_top) or from highest to lowest (top_bottom). This keyword may be useful when utilizing some USGS data sets, which are provided in top_bottom order. Default value is bottom_top.

27. ENDIAN : A character string, either big or little, specifying whether the values in the static data set arrays are in big-endian or little-endian byte order. Default value is big.

28. ISWATER : An integer specifying the land use category of water. Default value is 16.

29. ISLAKE : An integer specifying the land use category of inland water bodies. Default value is -1 (i.e., no separate inland water category).

30. ISICE : An integer specifying the land use category of ice. Default value is 24.

31. ISURBAN : An integer specifying the land use category of urban areas. Default value is 1.

32. ISOILWATER : An integer specifying the soil category of water. Default value is 14.

33. MMINLU : A character string, enclosed in quotation marks ("), indicating which section of WRF's LANDUSE.TBL and VEGPARM.TBL will be used when looking up parameters for land use categories. Default value is "USGS".

34. FILENAME_DIGITS : An integer specifying the number of digits used in the names of data tiles. Possible values are 5 or 6. Default value is 5.

**Description of METGRID.TBL Options**

The METGRID.TBL file is a text file that defines parameters of each of the meteorological fields to be interpolated by metgrid. Parameters for each field are defined in a separate section, with sections being delimited by a line of equality symbols (e.g., ‘==============’). Within each section, there are specifications, each of which has the form of keyword=value. Some keywords are required in a section, while others are optional; some keywords are mutually exclusive with other keywords. Below, the possible keywords and their expected range of values are described.

1. NAME : A character string giving the name of the meteorological field to which the containing section of the table pertains. The name should exactly match that of the field
as given in the intermediate files (and, thus, the name given in the Vtable used in generating the intermediate files). This field is required. No default value.

2. OUTPUT : Either yes or no, indicating whether the field is to be written to the metgrid output files or not. Default value is yes.

3. MANDATORY : Either yes or no, indicating whether the field is required for successful completion of metgrid. Default value is no.

4. OUTPUT_NAME : A character string giving the name that the interpolated field should be output as. When a value is specified for output_name, the interpolation options from the table section pertaining to the field with the specified name are used. Thus, the effects of specifying output_name are two-fold: The interpolated field is assigned the specified name before being written out, and the interpolation methods are taken from the section pertaining to the field whose name matches the value assigned to the output_name keyword. No default value.

5. FROM_INPUT : A character string used to compare against the values in the fg_name namelist variable; if from_input is specified, the containing table section will only be used when the time-varying input source has a filename that contains the value of from_input as a substring. Thus, from_input may be used to specify different interpolation options for the same field, depending on which source of the field is being processed. No default value.

6. OUTPUT_STAGGER : The model grid staggering to which the field should be interpolated. For ARW, this must be one of U, V, and M; for NMM, this must be one of HH and VV. Default value for ARW is M; default value for NMM is HH.

7. IS_U_FIELD : Either yes or no, indicating whether the field is to be used as the wind U-component field. For ARW, the wind U-component field must be interpolated to the U staggering (output_stagger=U); for NMM, the wind U-component field must be interpolated to the V staggering (output_stagger=VV). Default value is no.

8. IS_V_FIELD : Either yes or no, indicating whether the field is to be used as the wind V-component field. For ARW, the wind V-component field must be interpolated to the V staggering (output_stagger=V); for NMM, the wind V-component field must be interpolated to the V staggering (output_stagger=VV). Default value is no.

9. INTERP_OPTION : A sequence of one or more character strings, which are the names of interpolation methods to be used when horizontally interpolating the field. Available interpolation methods are: average_4pt, average_16pt, wt_average_4pt, wt_average_16pt, nearest_neighbor, four_pt, sixteen_pt, search(r), and average_gcell(r). For the search method (search), the optional argument r specifies the maximum search radius in units of grid points in the grid of the source data; the default search radius is 1200 points. For the grid cell average method (average_gcell), the optional argument r specifies the minimum ratio of source data resolution to
simulation grid resolution at which the method will be applied; unless specified, \( r = 0.0 \), and the option is used for any ratio. When a sequence of two or more methods are given, the methods should be separated by a + sign. Default value is nearest_neighbor.

10. INTERP_MASK : The name of the field to be used as an interpolation mask, along with the value within that field which signals masked points and an optional relational symbol, < or >. A specification takes the form field(?maskval), where field is the name of the field, ? is an optional relational symbol (< or >), and maskval is a real value. Source data points will not be used in interpolation if the corresponding point in the field field is equal, greater than, or less than, the value of maskval for no relational symbol, a > symbol, or a < symbol, respectively. Default value is no mask.

11. INTERP_LAND_MASK : The name of the field to be used as an interpolation mask when interpolating to water points (determined by the static LANDMASK field), along with the value within that field which signals land points and an optional relational symbol, < or >. A specification takes the form field(?maskval), where field is the name of the field, ? is an optional relational symbol (< or >), and maskval is a real value. Default value is no mask.

12. INTERP_WATER_MASK : The name of the field to be used as an interpolation mask when interpolating to land points (determined by the static LANDMASK field), along with the value within that field which signals water points and an optional relational symbol, < or >. A specification takes the form field(?maskval), where field is the name of the field, ? is an optional relational symbol (< or >), and maskval is a real value. Default value is no mask.

13. FILL_MISSING : A real number specifying the value to be assigned to model grid points that received no interpolated value, for example, because of missing or incomplete meteorological data. Default value is 1.E20.

14. Z_DIM_NAME : For 3-dimensional meteorological fields, a character string giving the name of the vertical dimension to be used for the field on output. Default value is num_metgrid_levels.

15. DERIVED : Either yes or no, indicating whether the field is to be derived from other interpolated fields, rather than interpolated from an input field. Default value is no.

16. FILL_LEV : The fill_lev keyword, which may be specified multiple times within a table section, specifies how a level of the field should be filled if that level does not already exist. A generic value for the keyword takes the form DLEVEL:FIELD(SLEVEL), where DLEVEL specifies the level in the field to be filled, FIELD specifies the source field from which to copy levels, and SLEVEL specifies the level within the source field to use. DLEVEL may either be an integer or the string all. FIELD may either be the name of another field, the string const, or the string vertical_index. If FIELD is specified as const, then SLEVEL is a constant value that will be used to fill with; if FIELD is specified as vertical_index, then (SLEVEL) must not be specified, and the value of the
vertical index of the source field is used; if \textit{DLEVEL} is 'all', then all levels from the field specified by the \texttt{level_template} keyword are used to fill the corresponding levels in the field, one at a time. No default value.

17. \textbf{LEVEL TEMPLATE} : A character string giving the name of a field from which a list of vertical levels should be obtained and used as a template. This keyword is used in conjunction with a \texttt{fill_lev} specification that uses all in the \textit{DLEVEL} part of its specification. No default value.

18. \textbf{MASKED} : Either \texttt{land}, \texttt{water}, or \texttt{both}. Setting \texttt{MASKED} to \texttt{land} or \texttt{water} indicates that the field should not be interpolated to WRF land or water points, respectively; however, setting \texttt{MASKED} to \texttt{both} indicates that the field should be interpolated to WRF land points using only land points in the source data and to WRF water points using only water points in the source data. When a field is masked, or invalid, the static \texttt{LANDMASK} field will be used to determine which model grid points the field should be interpolated to; invalid points will be assigned the value given by the \texttt{FILL_MISSING} keyword. Whether a source data point is land or water is determined by the masks specified using the \texttt{INTERP_LAND_MASK} and \texttt{INTERP_WATER_MASK} options. Default value is null (i.e., the field is valid for both land and water points).

19. \textbf{MISSING VALUE} : A real number giving the value in the input field that is assumed to represent missing data. No default value.

20. \textbf{VERTICAL_INTERP_OPTION} : A character string specifying the vertical interpolation method that should be used when vertically interpolating to missing points. Currently, this option is not implemented. No default value.

21. \textbf{FLAG_IN_OUTPUT} : A character string giving the name of a global attribute which will be assigned a value of 1 and written to the metgrid output if the interpolated field is to be output (output=yes). Default value is null (i.e., no flag will be written for the field).

\section*{Available Interpolation Options in Geogrid and Metgrid}

Through the \texttt{GEOGRID.TBL} and \texttt{METGRID.TBL} files, the user can control the method by which source data – either static fields in the case of geogrid or meteorological fields in the case of metgrid – are interpolated. In fact, a list of interpolation methods may be given, in which case, if it is not possible to employ the \textit{i}-th method in the list, the \textit{(i+1)}-st method will be employed, until either some method can be used or there are no methods left to try in the list. For example, to use a four-point bi-linear interpolation scheme for a field, we could specify \texttt{interp_option=four_pt}. However, if the field had areas of missing values, which could prevent the \texttt{four_pt} option from being used, we could request that a simple four-point average be tried if the \texttt{four_pt} method couldn't be used by specifying \texttt{interp_option=four_pt+average_4pt} instead. Below, each of the available interpolation options in the WPS are described conceptually; for the details of
each method, the user is referred to the source code in the file WPS/geogrid/src/interp_options.F.

1. four_pt : Four-point bi-linear interpolation

![Four-point bi-linear interpolation diagram]

The four-point bi-linear interpolation method requires four valid source points \(a_{ij}, 1 \leq i, j \leq 2\), surrounding the point \((x,y)\), to which geogrid or metgrid must interpolate, as illustrated in the figure above. Intuitively, the method works by linearly interpolating to the \(x\)-coordinate of the point \((x,y)\) between \(a_{11}\) and \(a_{12}\), and between \(a_{21}\) and \(a_{22}\), and then linearly interpolating to the \(y\)-coordinate using these two interpolated values.

2. sixteen_pt : Sixteen-point overlapping parabolic interpolation

![Sixteen-point overlapping parabolic interpolation diagram]

The sixteen_pt overlapping parabolic interpolation method requires sixteen valid source points surrounding the point \((x,y)\), as illustrated in the figure above. The method works by fitting one parabola to the points \(a_{i1}, a_{i2}, a_{i3}\), and another parabola to the points \(a_{i2}, a_{i3}, a_{i4}\), for row \(i, 1 \leq i \leq 4\); then, an intermediate interpolated value \(p_i\) within row \(i\) at the \(x\)-coordinate of the point is computed by taking an average of the values of the two parabolas evaluated at \(x\), with the average being weighted linearly by the distance of \(x\) from \(a_{i2}\) and \(a_{i3}\). Finally, the interpolated value at \((x,y)\) is found by performing the same operations as for a row of points, but for the column of interpolated values \(p_i\) to the \(y\)-coordinate of \((x,y)\).
3. **average_4pt : Simple four-point average interpolation**

The four-point average interpolation method requires at least one valid source data point from the four source points surrounding the point \((x,y)\). The interpolated value is simply the average value of all valid values among these four points.

4. **wt_average_4pt : Weighted four-point average interpolation**

The weighted four-point average interpolation method can handle missing or masked source data points, and the interpolated value is given as the weighted average of all valid values, with the weight \(w_{ij}\) for the source point \(a_{ij}\), \(1 \leq i, j \leq 2\), given by

\[
w_{ij} = \max \{0, 1 - \sqrt{(x-x_i)^2 + (y-y_j)^2}\}.
\]

Here, \(x_i\) is the \(x\)-coordinate of \(a_{ij}\) and \(y_j\) is the \(y\)-coordinate of \(a_{ij}\).

5. **average_16pt : Simple sixteen-point average interpolation**

The sixteen-point average interpolation method works in an identical way to the four-point average, but considers the sixteen points surrounding the point \((x,y)\).

6. **wt_average_16pt : Weighted sixteen-point average interpolation**

The weighted sixteen-point average interpolation method works like the weighted four-point average, but considers the sixteen points surrounding \((x,y)\); the weights in this method are given by

\[
w_{ij} = \max \{0, 2 - \sqrt{(x-x_i)^2 + (y-y_j)^2}\},
\]

where \(x_i\) and \(y_j\) are as defined for the weighted four-point method, and \(1 \leq i, j \leq 4\).

7. **nearest_neighbor : Nearest neighbor interpolation**

The nearest neighbor interpolation method simply sets the interpolated value at \((x,y)\) to the value of the nearest source data point, regardless of whether this nearest source point is valid, missing, or masked.
8. search : Breadth-first search interpolation

The breadth-first search option works by treating the source data array as a 2-d grid graph, where each source data point, whether valid or not, is represented by a vertex. Then, the value assigned to the point \((x,y)\) is found by beginning a breadth-first search at the vertex corresponding to the nearest neighbor of \((x,y)\), and stopping once a vertex representing a valid (i.e., not masked or missing) source data point is found. In effect, this method can be thought of as "nearest valid neighbor".

9. average_gcell : Model grid-cell average

The grid-cell average interpolator may be used when the resolution of the source data is higher than the resolution of the model grid. For a model grid cell \(\Gamma\), the method takes a simple average of the values of all source data points that are nearer to the center of \(\Gamma\) than to the center of any other grid cell. The operation of the grid-cell average method is illustrated in the figure above, where the interpolated value for the model grid cell – represented as the large rectangle – is given by the simple average of the values of all of the shaded source grid cells.

Land Use and Soil Categories in the Static Data

The default land use and soil category data sets that are provided as part of the WPS static data tar file contain categories that are matched with the USGS categories described in the VEGPARM.TBL and SOILPARM.TBL files in the WRF run directory. Descriptions of the 24 land use categories and 16 soil categories are provided in the tables below.
### Table 1: USGS 24-category Land Use Categories

<table>
<thead>
<tr>
<th>Land Use Category</th>
<th>Land Use Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Urban and Built-up Land</td>
</tr>
<tr>
<td>2</td>
<td>Dryland Cropland and Pasture</td>
</tr>
<tr>
<td>3</td>
<td>Irrigated Cropland and Pasture</td>
</tr>
<tr>
<td>4</td>
<td>Mixed Dryland/Irrigated Cropland and Pasture</td>
</tr>
<tr>
<td>5</td>
<td>Cropland/Grassland Mosaic</td>
</tr>
<tr>
<td>6</td>
<td>Cropland/Woodland Mosaic</td>
</tr>
<tr>
<td>7</td>
<td>Grassland</td>
</tr>
<tr>
<td>8</td>
<td>Shrubland</td>
</tr>
<tr>
<td>9</td>
<td>Mixed Shrubland/Grassland</td>
</tr>
<tr>
<td>10</td>
<td>Savanna</td>
</tr>
<tr>
<td>11</td>
<td>Deciduous Broadleaf Forest</td>
</tr>
<tr>
<td>12</td>
<td>Deciduous Needleleaf Forest</td>
</tr>
<tr>
<td>13</td>
<td>Evergreen Broadleaf</td>
</tr>
<tr>
<td>14</td>
<td>Evergreen Needleleaf</td>
</tr>
<tr>
<td>15</td>
<td>Mixed Forest</td>
</tr>
<tr>
<td>16</td>
<td>Water Bodies</td>
</tr>
<tr>
<td>17</td>
<td>Herbaceous Wetland</td>
</tr>
<tr>
<td>18</td>
<td>Wooden Wetland</td>
</tr>
<tr>
<td>19</td>
<td>Barren or Sparsely Vegetated</td>
</tr>
<tr>
<td>20</td>
<td>Herbaceous Tundra</td>
</tr>
<tr>
<td>21</td>
<td>Wooded Tundra</td>
</tr>
<tr>
<td>22</td>
<td>Mixed Tundra</td>
</tr>
<tr>
<td>23</td>
<td>Bare Ground Tundra</td>
</tr>
<tr>
<td>24</td>
<td>Snow or Ice</td>
</tr>
</tbody>
</table>

### Table 2: IGBP-Modified MODIS 20-category Land Use Categories

<table>
<thead>
<tr>
<th>Land Use Category</th>
<th>Land Use Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Evergreen Needleleaf Forest</td>
</tr>
<tr>
<td>2</td>
<td>Evergreen Broadleaf Forest</td>
</tr>
<tr>
<td>3</td>
<td>Deciduous Needleleaf Forest</td>
</tr>
<tr>
<td>4</td>
<td>Deciduous Broadleaf Forest</td>
</tr>
<tr>
<td>5</td>
<td>Mixed Forest</td>
</tr>
<tr>
<td>6</td>
<td>Closed Shrublands</td>
</tr>
<tr>
<td>7</td>
<td>Open Shrublands</td>
</tr>
<tr>
<td>8</td>
<td>Woody Savannas</td>
</tr>
<tr>
<td>9</td>
<td>Savannas</td>
</tr>
<tr>
<td>10</td>
<td>Grasslands</td>
</tr>
<tr>
<td>11</td>
<td>Permanent Wetlands</td>
</tr>
<tr>
<td>12</td>
<td>Croplands</td>
</tr>
</tbody>
</table>
Table 3: 16-category Soil Categories

<table>
<thead>
<tr>
<th>Soil Category</th>
<th>Soil Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sand</td>
</tr>
<tr>
<td>2</td>
<td>Loamy Sand</td>
</tr>
<tr>
<td>3</td>
<td>Sandy Loam</td>
</tr>
<tr>
<td>4</td>
<td>Silt Loam</td>
</tr>
<tr>
<td>5</td>
<td>Silt</td>
</tr>
<tr>
<td>6</td>
<td>Loam</td>
</tr>
<tr>
<td>7</td>
<td>Sandy Clay Loam</td>
</tr>
<tr>
<td>8</td>
<td>Silty Clay Loam</td>
</tr>
<tr>
<td>9</td>
<td>Clay Loam</td>
</tr>
<tr>
<td>10</td>
<td>Sandy Clay</td>
</tr>
<tr>
<td>11</td>
<td>Silty Clay</td>
</tr>
<tr>
<td>12</td>
<td>Clay</td>
</tr>
<tr>
<td>13</td>
<td>Organic Material</td>
</tr>
<tr>
<td>14</td>
<td>Water</td>
</tr>
<tr>
<td>15</td>
<td>Bedrock</td>
</tr>
<tr>
<td>16</td>
<td>Other (land-ice)</td>
</tr>
</tbody>
</table>

WPS Output Fields

Below, a listing of the global attributes and fields that are written to the geogrid program's output files is given. This listing is an abridged version of the output from the ncdump program when run on a typical geo_em.d01.nc file.

```python
netcdf geo_em.d01 {
  dimensions:
    Time = UNLIMITED ; // (1 currently)
    DateStrLen = 19 ;
    west_east = 73 ;
    south_north = 60 ;
    south_north_stag = 61 ;
    west_east_stag = 74 ;
    land_cat = 21 ;
}```
soil_cat = 16;
month = 12;
um_urb_params = 132;

variables:
  char Times(Time, DateStrLen);
  float XLAT_M(Time, south_north, west_east);
    XLAT_M:units = "degrees latitude";
    XLAT_M:description = "Latitude on mass grid";
  float XLONG_M(Time, south_north, west_east);
    XLONG_M:units = "degrees longitude";
    XLONG_M:description = "Longitude on mass grid";
  float XLAT_V(Time, south_north_stag, west_east);
    XLAT_V:units = "degrees latitude";
    XLAT_V:description = "Latitude on V grid";
  float XLONG_V(Time, south_north_stag, west_east);
    XLONG_V:units = "degrees longitude";
    XLONG_V:description = "Longitude on V grid";
  float XLAT_U(Time, south_north, west_east_stag);
    XLAT_U:units = "degrees latitude";
    XLAT_U:description = "Latitude on U grid";
  float XLONG_U(Time, south_north, west_east_stag);
    XLONG_U:units = "degrees longitude";
    XLONG_U:description = "Longitude on U grid";
  float CLAT(Time, south_north, west_east);
    CLAT:units = "degrees latitude";
    CLAT:description = "Computational latitude on mass grid";
  float CLONG(Time, south_north, west_east);
    CLONG:units = "degrees longitude";
    CLONG:description = "Computational longitude on mass grid";
  float MAPFAC_M(Time, south_north, west_east);
    MAPFAC_M:units = "none";
    MAPFAC_M:description = "Mapfactor on mass grid";
  float MAPFAC_V(Time, south_north_stag, west_east);
    MAPFAC_V:units = "none";
    MAPFAC_V:description = "Mapfactor on V grid";
  float MAPFAC_U(Time, south_north, west_east_stag);
    MAPFAC_U:units = "none";
    MAPFAC_U:description = "Mapfactor on U grid";
  float MAPFAC_MX(Time, south_north, west_east);
    MAPFAC_MX:units = "none";
    MAPFAC_MX:description = "Mapfactor (x-dir) on mass grid";
  float MAPFAC_VX(Time, south_north_stag, west_east);
    MAPFAC_VX:units = "none";
    MAPFAC_VX:description = "Mapfactor (x-dir) on V grid";
  float MAPFAC_UX(Time, south_north, west_east_stag);
    MAPFAC_UX:units = "none";
    MAPFAC_UX:description = "Mapfactor (x-dir) on U grid";
  float MAPFAC_MY(Time, south_north, west_east);
    MAPFAC_MY:units = "none";
    MAPFAC_MY:description = "Mapfactor (y-dir) on mass grid";
  float MAPFAC_VY(Time, south_north_stag, west_east);
    MAPFAC_VY:units = "none";
    MAPFAC_VY:description = "Mapfactor (y-dir) on V grid";
  float MAPFAC_UY(Time, south_north, west_east_stag);
    MAPFAC_UY:units = "none";
    MAPFAC_UY:description = "Mapfactor (y-dir) on U grid";
  float E(Time, south_north, west_east);
    E:units = "-";
    E:description = "Coriolis E parameter";
  float F(Time, south_north, west_east);
    F:units = "-";
    F:description = "Coriolis F parameter";
  float SINALPHA(Time, south_north, west_east);
SINALPHA:units = "none";
SINALPHA:description = "Sine of rotation angle";
float COSALPHA(Time, south_north, west_east);
COSALPHA:units = "none";
COSALPHA:description = "Cosine of rotation angle";
float LANDMASK(Time, south_north, west_east);
LANDMASK:units = "none";
LANDMASK:description = "Landmask: 1=land, 0=water";
float XLAT_C(Time, south_north_stag, west_east_stag);
XLAT_C:units = "degrees latitude";
XLAT_C:description = "Latitude at grid cell corners";
float XLONG_C(Time, south_north_stag, west_east_stag);
XLONG_C:units = "degrees longitude";
XLONG_C:description = "Longitude at grid cell corners";
float LANDUSEF(Time, land_cat, south_north, west_east);
LANDUSEF:units = "category";
LANDUSEF:description = "Noah-modified 21-category IGBP-MODIS landuse";

float LU_INDEX(Time, south_north, west_east);
LU_INDEX:units = "category";
LU_INDEX:description = "Dominant category";
float HGT_M(Time, south_north, west_east);
HGT_M:units = "meters MSL";
HGT_M:description = "GMTED2010 30-arc-second topography height";
float SOILTTEMP(Time, south_north, west_east);
SOILTTEMP:units = "Kelvin";
SOILTTEMP:description = "Annual mean deep soil temperature";
float SOILCTOP(Time, soil_cat, south_north, west_east);
SOILCTOP:units = "category";
SOILCTOP:description = "16-category top-layer soil type";
float SCT_DOM(Time, south_north, west_east);
SCT_DOM:units = "category";
SCT_DOM:description = "Dominant category";
float SOILCBOT(Time, soil_cat, south_north, west_east);
SOILCBOT:units = "category";
SOILCBOT:description = "16-category top-layer soil type";
float SCB_DOM(Time, south_north, west_east);
SCB_DOM:units = "category";
SCB_DOM:description = "Dominant category";
float ALBEDO12M(Time, month, south_north, west_east);
ALBEDO12M:units = "percent";
ALBEDO12M:description = "Monthly surface albedo";
float GREENFRAC(Time, month, south_north, west_east);
GREENFRAC:units = "fraction";
GREENFRAC:description = "MODIS FPAR";
float LAI12M(Time, month, south_north, west_east);
LAI12M:units = "m^2/m^2";
LAI12M:description = "MODIS LAI";
float SNOALB(Time, south_north, west_east);
SNOALB:units = "percent";
SNOALB:description = "Maximum snow albedo";
float SLOPECAT(Time, south_north, west_east);
SLOPECAT:units = "category";
SLOPECAT:description = "Dominant category";
float CON(Time, south_north, west_east);
CON:units = "";
CON:description = "Subgrid-scale orographic convexity";
float VAR(Time, south_north, west_east);
VAR:units = "";
VAR:description = "Subgrid-scale orographic variance";
float OAI(Time, south_north, west_east);
OAI:units = "";
OAI:description = "Subgrid-scale orographic asymmetry";
float OA2(Time, south_north, west_east);
OA2:units = "";
OA2:description = "Subgrid-scale orographic asymmetry";
float OA3(Time, south_north, west_east);
OA3:units = "";
OA3:description = "Subgrid-scale orographic asymmetry";
float OA4(Time, south_north, west_east);
OA4:units = "";
OA4:description = "Subgrid-scale orographic asymmetry";
float OL1(Time, south_north, west_east);
OL1:units = "";
OL1:description = "Subgrid-scale effective orographic length scale";
float OL2(Time, south_north, west_east);
OL2:units = "";
OL2:description = "Subgrid-scale effective orographic length scale";
float OL3(Time, south_north, west_east);
OL3:units = "";
OL3:description = "Subgrid-scale effective orographic length scale";
float OL4(Time, south_north, west_east);
OL4:units = "";
OL4:description = "Subgrid-scale effective orographic length scale";
float VAR_SSO(Time, south_north, west_east);
VAR_SSO:units = "meters2 MSL";
VAR_SSO:description = "Variance of Subgrid Scale Orography";
float LAKE_DEPTH(Time, south_north, west_east);
LAKE_DEPTH:units = "meters MSL";
LAKE_DEPTH:description = "Topography height";
float URB_PARAM(Time, num_urb_params, south_north, west_east);
URB_PARAM:units = "dimensionless";
URB_PARAM:description = "Urban_Parameters";

// global attributes:
:TITLE = "OUTPUT FROM GEOGRID V3.8";
:SIMULATION_START_DATE = "0000-00-00_00:00:00";
:WEST-EAST_GRID_DIMENSION = 74;
:SOUTH-NORTH_GRID_DIMENSION = 61;
:BOTTOM-TOP_GRID_DIMENSION = 0;
:WEST-EAST_PATCH_START_UNSTAG = 1;
:WEST-EAST_PATCH_END_UNSTAG = 73;
:WEST-EAST_PATCH_START_STAG = 1;
:WEST-EAST_PATCH_END_STAG = 74;
:SOUTH-NORTH_PATCH_START_UNSTAG = 1;
:SOUTH-NORTH_PATCH_END_UNSTAG = 60;
:SOUTH-NORTH_PATCH_START_STAG = 1;
:SOUTH-NORTH_PATCH_END_STAG = 61;
:GRIDTYPE = "C";
:DX = 30000.f;
:DY = 30000.f;
:DYN_OPT = 2;
:CEN_LAT = 34.83001f;
:CEN_LON = -81.03f;
:TRUELAT1 = 30.f;
:TRUELAT2 = 60.f;
:MUNIT_CEN_LAT = 34.83001f;
:MUNIT_CEN_LAT = -98.f;
:POLE_LAT = 90.f;
:POLE_LON = 0.f;
:corner_lats = 28.17127f, 44.36657f, 39.63231f, 24.61906f, 28.17842f, 44.37617f, 39.57812f, 24.57806f, 28.03771f, 44.50592f, 39.76032f, 24.49311f, 28.04485f, 44.51553f, 39.70599f, 24.45341f;
The global attributes `corner_lats` and `corner_lons` contain the lat-lon location of the corners of the domain with respect to different grid staggerings (mass, u, v, and unstaggered). The locations referred to by each element of the `corner_lats` and `corner_lons` arrays are summarized in the table and figure below.

<table>
<thead>
<tr>
<th>Array index</th>
<th>Staggering</th>
<th>Corner</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mass</td>
<td>Lower-left</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Upper-left</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Upper-right</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Lower-right</td>
</tr>
<tr>
<td>5</td>
<td>U</td>
<td>Lower-left</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Upper-left</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Upper-right</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Lower-right</td>
</tr>
<tr>
<td>9</td>
<td>V</td>
<td>Lower-left</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Upper-left</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Upper-right</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Lower-right</td>
</tr>
<tr>
<td>13</td>
<td>Unstaggered</td>
<td>Lower-left</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Upper-left</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Upper-right</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Lower-right</td>
</tr>
</tbody>
</table>
In addition to the fields in a geogrid output file (e.g., geo_em.d01.nc), the following fields and global attributes will also be present in a typical output file from the metgrid program, run with the default METGRID.TBL file and meteorological data from NCEP's GFS model.

```plaintext
cell netcdf met_em.d01.2016-04-07_00:00:00 {
dimensions:
    Time = UNLIMITED ; // (1 currently)
    DateStrLen = 19 ;
    west_east = 73 ;
    south_north = 60 ;
    num_metgrid_levels = 27 ;
    num_st_layers = 4 ;
    num_sm_layers = 4 ;
    south_north_stag = 61 ;
    west_east_stag = 74 ;
    z-dimension0132 = 132 ;
    z-dimension0012 = 12 ;
    z-dimension0016 = 16 ;
    z-dimension0021 = 21 ;
variables:
    char Times(Time, DateStrLen) ;
    float PRES(Time, num_metgrid_levels, south_north, west_east) ;
        PRES:units = "" ;
        PRES:description = "" ;
    float SOIL_LAYERS(Time, num_st_layers, south_north, west_east) ;
        SOIL_LAYERS:units = "" ;
        SOIL_LAYERS:description = "" ;
    float SM(Time, num_sm_layers, south_north, west_east) ;
        SM:units = "" ;
        SM:description = "" ;
    float ST(Time, num_st_layers, south_north, west_east) ;
        ST:units = "" ;
        ST:description = "" ;
    float GHT(Time, num_metgrid_levels, south_north, west_east) ;
        GHT:units = "m" ;
        GHT:description = "Height" ;
    float HGTTROP(Time, south_north, west_east) ;
        HGTTROP:units = "m" ;
        HGTTROP:description = "Height of tropopause" ;
    float TTROP(Time, south_north, west_east) ;
        TTROP:units = "K" ;
        TTROP:description = "Temperature at tropopause" ;
```
float PTROPNN(Time, south_north, west_east);
PTROPNN:units = "Pa";
PTROPNN:description = "PTROP, used for nearest neighbor interp";
float PTROP(Time, south_north, west_east);
PTROP:units = "Pa";
PTROP:description = "Pressure of tropopause";
float VTROP(Time, south_north_stag, west_east);
VTROP:units = "m s^{-1}";
VTROP:description = "V at tropopause";
float UTROP(Time, south_north, west_east_stag);
UTROP:units = "m s^{-1}";
UTROP:description = "U at tropopause";
float HGTMAXW(Time, south_north, west_east);
HGTMAXW:units = "m";
HGTMAXW:description = "Height of max wind level";
float TMAXW(Time, south_north, west_east);
TMAXW:units = "K";
TMAXW:description = "Temperature at max wind level";
float PMAXWNN(Time, south_north, west_east);
PMAXWNN:units = "Pa";
PMAXWNN:description = "PMAXW, used for nearest neighbor interp";
float PMAXW(Time, south_north, west_east);
PMAXW:units = "Pa";
PMAXW:description = "Pressure of max wind level";
float VMAXW(Time, south_north_stag, west_east);
VMAXW:units = "m s^{-1}";
VMAXW:description = "V at max wind";
float UMAXW(Time, south_north, west_east_stag);
UMAXW:units = "m s^{-1}";
UMAXW:description = "U at max wind";
float SNOWH(Time, south_north, west_east);
SNOWH:units = "m";
SNOWH:description = "Physical Snow Depth";
float SNOW(Time, south_north, west_east);
SNOW:units = "kg m^{-2}";
SNOW:description = "Water equivalent snow depth";
float SKINTEMP(Time, south_north, west_east);
SKINTEMP:units = "K";
SKINTEMP:description = "Skin temperature";
float SOILHGT(Time, south_north, west_east);
SOILHGT:units = "m";
SOILHGT:description = "Terrain field of source analysis";
float LANDSEA(Time, south_north, west_east);
LANDSEA:units = "proprtn";
LANDSEA:description = "Land/Sea flag (1=land, 0 or 2=sea)";
float SEAICE(Time, south_north, west_east);
SEAICE:units = "proprtn";
SEAICE:description = "Ice flag";
float ST100200(Time, south_north, west_east);
ST100200:units = "K";
ST100200:description = "T 100-200 cm below ground layer (Bottom)";
float ST040100(Time, south_north, west_east);
ST040100:units = "K";
ST040100:description = "T 40-100 cm below ground layer (Upper)";
float ST010040(Time, south_north, west_east);
ST010040:units = "K";
ST010040:description = "T 10-40 cm below ground layer (Upper)";
float ST000010(Time, south_north, west_east);
ST000010:units = "K";
ST000010:description = "T 0-10 cm below ground layer (Upper)";
float SM100200(Time, south_north, west_east);
SM100200:units = "fraction";
SM100200:description = "Soil Moist 100-200 cm below gr layer";
float SM040100(Time, south_north, west_east) ;
SM040100:units = "fraction" ;
SM040100:description = "Soil Moist 40-100 cm below grn layer" ;
float SM010040(Time, south_north, west_east) ;
SM010040:units = "fraction" ;
SM010040:description = "Soil Moist 10-40 cm below grn layer" ;
float SM000010(Time, south_north, west_east) ;
SM000010:units = "fraction" ;
SM000010:description = "Soil Moist 0-10 cm below grn layer (Up)" ;
float PSFC(Time, south_north, west_east) ;
PSFC:units = "Pa" ;
PSFC:description = "Surface Pressure" ;
float RH(Time, num_metgrid_levels, south_north, west_east) ;
RH:units = "%" ;
RH:description = "Relative Humidity" ;
float VV(Time, num_metgrid_levels, south_north_stag, west_east) ;
VV:units = "m s\-1" ;
VV:description = "V" ;
float UU(Time, num_metgrid_levels, south_north, west_east_stag) ;
UU:units = "m s\-1" ;
UU:description = "U" ;
float TT(Time, num_metgrid_levels, south_north, west_east) ;
TT:units = "K" ;
TT:description = "Temperature" ;
float PMSL(Time, south_north, west_east) ;
PMSL:units = "Pa" ;
PMSL:description = "Sea-level Pressure" ;

// global attributes:
:TITLE = "OUTPUT FROM METGRID V3.8" ;
:SIMULATION_START_DATE = "2016-04-07_00:00:00" ;
:WEST-EAST_GRID_DIMENSION = 74 ;
:SOUTH-NORTH_GRID_DIMENSION = 61 ;
:BOTTOM-TOP_GRID_DIMENSION = 27 ;
:WEST-EAST_PATCH_START_UNSTAG = 1 ;
:WEST-EAST_PATCH_END_UNSTAG = 73 ;
:WEST-EAST_PATCH_START_STAG = 1 ;
:WEST-EAST_PATCH_END_STAG = 74 ;
:SOUTH-NORTH_PATCH_START_UNSTAG = 1 ;
:SOUTH-NORTH_PATCH_END_UNSTAG = 60 ;
:SOUTH-NORTH_PATCH_START_STAG = 1 ;
:SOUTH-NORTH_PATCH_END_STAG = 61 ;
:GRIDTYPE = "C" ;
:DX = 30000.f ;
:DY = 30000.f ;
:DYN_OPT = 2 ;
:CENT_LAT = 34.83001f ;
:CENT_LON = -81.03f ;
:TRUELAT1 = 30.f ;
:TRUELAT2 = 60.f ;
:MOAD_CEN_LAT = 34.83001f ;
:STAND_LON = -98.f ;
:POLE_LAT = 90.f ;
:POLE_LON = 0.f ;
:corner_lats = 28.17127f, 44.36657f, 39.63231f, 24.61906f, 28.17842f, 44.37617f, 39.57812f, 24.57806f, 28.03771f, 44.50592f, 39.70632f, 24.49431f, 28.04485f, 44.51553f, 39.70599f, 24.45341f ;
:MPPROJ = 1 ;
:MMINLU = "MODIFIED_IGBP_MODIS_NOAH" ;
:NUM_LAND_CAT = 21 ;
:ISWATER = 17 ;
:ISLAKE = 21 ;
:ISICE = 15 ;
:ISURBAN = 13 ;
:ISOILWATER = 14 ;
:grid_id = 1 ;
:parent_id = 1 ;
:i_parent_start = 1 ;
:j_parent_start = 1 ;
:i_parent_end = 74 ;
:j_parent_end = 61 ;
:parent_grid_ratio = 1 ;
:NUM_METGRID_SOIL_LEVELS = 4 ;
:FLAG_METGRID = 1 ;
:FLAG_EXCLUDED_MIDDLE = 0 ;
:FLAG_SOIL_LAYERS = 1 ;
:FLAG_SNOW = 1 ;
:FLAG_PSF = 1 ;
:FLAG_SM000010 = 1 ;
:FLAG_SM010040 = 1 ;
:FLAG_SM040100 = 1 ;
:FLAG_SM100200 = 1 ;
:FLAG_ST000010 = 1 ;
:FLAG_ST010040 = 1 ;
:FLAG_ST040100 = 1 ;
:FLAG_ST100200 = 1 ;
:FLAG_SLP = 1 ;
:FLAG_SNOWH = 1 ;
:FLAG_SOILHGT = 1 ;
:FLAG_UTROP = 1 ;
:FLAG_VTROP = 1 ;
:FLAG_TTROP = 1 ;
:FLAG_PTROP = 1 ;
:FLAG_PTROPNN = 1 ;
:FLAG_HGTTROP = 1 ;
:FLAG_UMAXW = 1 ;
:FLAG_VMAXW = 1 ;
:FLAG_TMAXW = 1 ;
:FLAG_PMAXW = 1 ;
:FLAG_PMAXWNN = 1 ;
:FLAG_HGTMAXW = 1 ;
:FLAG_MF_XY = 1 ;
:FLAG_LAI12M = 1 ;
:FLAG_LAKE_DEPTH = 1 ;
Chapter 4: WRF Initialization

Table of Contents

- Introduction
- Initialization for Ideal Data Cases
- Initialization for Real Data Cases

Introduction

The WRF model has two large classes of simulations that it is able to generate: those with an ideal initialization and those utilizing real data. The idealized simulations typically manufacture an initial condition file for the WRF model from an existing 1-D or 2-D sounding and assume a simplified analytic orography. The real-data cases usually require pre-processing from the WPS package, which provides each atmospheric and static field with fidelity appropriate to the chosen grid resolution for the model. The WRF model executable itself is not altered by choosing one initialization option over another (idealized vs. real), but the WRF model pre-processors (the real.exe and ideal.exe programs) are specifically built based upon a user's selection.

The real.exe and ideal.exe programs are never used together. Both the real.exe and ideal.exe are the programs that are processed just prior to the WRF model run.

The ideal vs. real cases are divided as follows:

- Ideal cases – initialization programs named “ideal.exe”
  - 3d
    - em_b_wave - baroclinic wave, 100 km
    - em_fire – surface fire, 50 m
    - em_heldsuarez – global case with polar filtering, 625 km
    - em_les – large eddy simulation, 100 m
    - em_quarter_ss - super cell, 2 km
    - em_tropical_cyclone – hurricane, 15 km
  - 2d
    - em_grav2d_x – gravity current, 100 m
    - em_hill2d_x – flow over a hill, 2 km
    - em_seabreeze2d_x – water and land, 2 km, full physics
    - em_squall2d_x – squall line, 250 m
The selection of the type of forecast is made when issuing the `./compile` statement. When selecting a different case to study, the code must be re-compiled to choose the correct initialization for the model. For example, after configuring the setup for the architecture (with the `./configure` command), if the user issues the command `./compile em_real`, then the initialization program is built using `module_initialize_real.F` as the target module (one of the `./WRFV3/dyn_em/module_initialize_*.F` files). Similarly, if the user specifies `./compile em_les`, then the Fortran module for the large eddy simulation (`module_initialize_les.F`) is automatically inserted into the build for ideal.exe.

Note that the WRF forecast model is identical for both of these initialization programs. In each of these initialization modules, the same sort of activities goes on:

- compute a base state / reference profile for geopotential and column pressure
- compute the perturbations from the base state for geopotential and column pressure
- initialize meteorological variables: u, v, potential temperature, vapor mixing ratio
- define a vertical coordinate
- interpolate data to the model’s vertical coordinate
- initialize static fields for the map projection and the physical surface; for many of the idealized cases, these are simplified initializations, such as map factors set to one, and topography elevation set to zero

Both the real.exe program and ideal.exe programs share a large portion of source code, to handle the following duties:

- read data from the namelist
- allocate space for the requested domain, with model variables specified at runtime
- generate initial condition file

The real-data case does some additional processing:

- read meteorological and static input data from the WRF Preprocessing System (WPS)
- prepare soil fields for use in the model (usually, vertical interpolation to the required levels for the specified land surface scheme)
• check to verify that soil categories, land use, land mask, soil temperature, sea surface temperature are all consistent with each other
• multiple input time periods are processed to generate the lateral boundary conditions, which are required unless processing a global forecast
• 3d boundary data (u, v, potential temperature, vapor mixing ratio, total geopotential) are coupled with total column pressure

The “real.exe” program may be run as either a serial or a distributed memory job. Since the idealized cases only require that the initialization run for a single time period (no lateral boundary file is required) and are, therefore, quick to process, all of the “ideal.exe” programs should be run on a single processor. The Makefile for the 2-D cases will not allow the user to build the code with distributed memory parallelism. For large 2-D cases, if the user requires OpenMP, the variables \texttt{nproc}_x and \texttt{nproc}_y must be set in the \texttt{domains} portion of the namelist file \texttt{namelist.input} (\texttt{nproc}_y must be set to 1, and \texttt{nproc}_x then set to the number of processors).

**Initialization for Ideal Cases**

The program "ideal.exe" is the program in the WRF system that allows a user to run a controlled scenario. Typically this program requires no input except for the \texttt{namelist.input} and the \texttt{input_sounding} files (except for the b\_wave case which uses a 2-D binary sounding file). The program outputs the \texttt{wrfinput_d01} file that is read by the WRF model executable ("wrf.exe"). Since no external data is required to run the idealized cases, even for researchers interested in real-data cases, the idealized simulations are an easy way to insure that the model is working correctly on a particular architecture and compiler.

Idealized runs can use any of the boundary conditions except "specified", and are not, by default, set up to run with sophisticated physics (other than from microphysics). Most have no radiation, surface fluxes or frictional effects (other than the sea breeze case, LES, and the global Held-Suarez). The idealized cases are mostly useful for dynamical studies, reproducing converged or otherwise known solutions, and idealized cloud modeling.

There are 1-D, 2-D and 3-D examples of idealized cases, with and without topography, and with and without an initial thermal perturbation. The namelist can control the size of the domain, number of vertical levels, model top height, grid size, time step, diffusion and damping properties, boundary conditions, and physics options. A large number of existing namelist settings are already found within each of the directories associated with a particular case.

The \texttt{input_sounding} file (already in appropriate case directories) can be any set of levels that goes at least up to the model top height (\texttt{ztop}) in the namelist. The first line includes the surface pressure (hPa), potential temperature (K) and moisture mixing ratio.
(g/kg). Each subsequent line has five input values: height (meters above sea-level), potential temperature (K), vapor mixing ratio (g/kg), x-direction wind component (m/s), and y-direction wind component (m/s). The “ideal.exe” program interpolates the data from the input_sounding file, and will extrapolate if not enough data is provided.

The base state sounding for idealized cases is the initial sounding, minus the moisture, and therefore does not have to be defined separately. Note for the baroclinic wave case: a 1-D input sounding is not used because the initial 3-D arrays are read-in from the file input_jet. This means for the baroclinic wave case, the namelist.input file cannot be used to change the horizontal or vertical dimensions since they are specified in the input_jet file.

Making modifications, apart from namelist-controlled options or soundings, has to be done by editing the Fortran code. Such modifications would include changing the topography, the distribution of vertical levels, the properties of an initialization thermal bubble, or preparing a case to use more physics, such as a land-surface model. The Fortran code to edit is contained in ./WRFV3/dyn_em/module_initialize_[case].F, where [case] is the case chosen in compilation, e.g. module_initialize_squall2d_x.F. The subroutine to modify is init_domain_rk. To change the vertical levels, only the 1-D array znw must be defined, containing the full levels, starting from 1 at k=1, and ending with 0 at k=kde. To change the topography, only the 2-D array ht must be defined, making sure it is periodic if those boundary conditions are used. To change the thermal perturbation bubble, search for the string "bubble" to locate the code to change.

Each of the ideal cases provides an excellent set of default examples to the user. The method to specify a thermal bubble is given in the super cell case. In the hill2d case, the topography is accounted for properly in setting up the initial 3-D arrays, so that example should be followed for any topography cases. A symmetry example in the squall line cases tests that your indexing modifications are correct. Full physics options are demonstrated in the seabreeze2d_x case.

Available Ideal Test Cases

The available test cases are

1. 2-D squall2d_x (test/em_squall2d_x)
   - 2D squall line (x,z) using Kessler microphysics and a fixed 300 m^2/s viscosity.
   - Periodicity condition used in y so that 3D model produces 2D simulation.
   - v velocity should be zero and there should be no variation in y in the results.

2. 2-D squall2d_y (test/em_squall2d_y)
   - Same as squall2d_x, except with (x) rotated to (y).
   - u velocity should be zero and there should be no variation in x in the results.
3. 3-D quarter-circle shear supercell simulation (test/em_quarter_ss).
   - Left and right moving supercells are produced.
   - See the README.quarter_ss file in the test directory for more information.

4. 2-D flow over a bell-shaped hill (x,z) (test/em_hill2d_x)
   - 10 km half-width, 2 km grid-length, 100 m high hill, 10 m/s flow,
     N=0.01/s, 30 km high domain, 80 levels, open radiative boundaries,
     absorbing upper boundary.
   - Case is in linear hydrostatic regime, so vertical tilted waves with ~6-km vertical wavelength.

5. 3-D baroclinic waves (test/em_b_wave)
   - Baroclinically unstable jet u(y,z) on an f-plane.
   - Symmetric north and south, periodic east and west boundaries.
   - 100-km grid size, 16-km top, with 4-km damping layer.
   - 41x81 points in (x,y), 64 layers.

6. 2-D gravity current (test/em_grav2d_x)
   - See the README.grav2d_x file in the test directory.

7. 2-D sea breeze (test/em_seabreeze_x)
   - 2-km grid size, 20-km top, land/water.
   - Can be run with full physics, radiation, surface, boundary layer, and land options.

8. 3-D large eddy simulation (test/em_les)
   - 100-m grid size, 2-km top.
   - Surface layer physics with fluxes.
   - Doubly periodic

9. 3-D Held-Suarez (test/em_heldsuarez)
   - Global domain, 625 km in x-direction, 556 km in y-direction, 120-km top.
   - Radiation, polar filter above 45°.
   - Period in x-direction, polar boundary conditions in y-direction

10. 1-D single column model (test/em_scm_xy)
    - 4-km grid size, 12-km top
    - Full physics
    - Doubly periodic

11. 3-D surface fire (test/em_fire)
    - 50-m, 4.5-km top
    - 10:1 subgrid ratio, no physics
    - Open boundaries

12. 3-D tropical cyclone (test/em_tropical_cyclone)
    - 15-km, 25-km top
    - f-plane (f=0.5e-5, about 20 N), SST=28 C
Initialization

1. Full physics with a simple radiative cooling, no cumulus
2. Doubly periodic

13. 3-D convective-radiative equilibrium (test/em_convrad)
   - 1 km grid size, 30 km model top
   - Tropical condition, small f, weak wind, constant SST
   - Full physics
   - Doubly periodic

Initialization for Real Data Cases

The real-data WRF cases are those that have the input data to the “real.exe” program provided by the WRF Preprocessing System (WPS). This data from the WPS was originally generated from a previously-run external analysis or forecast model. The original data was most-likely in GriB format and was most-likely ingested into the WPS by first ftp'ing the raw GriB data from one of the national weather agencies’ anonymous ftp sites.

For example, suppose a single-domain WRF forecast is desired, with the following criteria:

- 2000 January 24 1200 UTC through January 25 1200 UTC
- The original GriB data is available at 6-h increments

The following coarse-grid files will be generated by the WPS (starting date through ending date, at 6-h increments):

- met_em.d01.2000-01-24_12:00:00.nc
- met_em.d01.2000-01-24_18:00:00.nc
- met_em.d01.2000-01-25_00:00:00.nc
- met_em.d01.2000-01-25_06:00:00.nc
- met_em.d01.2000-01-25_12:00:00.nc

The convention is to use "met" to signify data that is output from the WPS “metgrid.exe” program and input into the “real.exe” program. The "d01" portion of the name identifies to which domain this data refers, which permits nesting. The next set of characters is the validation date/time (UTC), where each WPS output file has only a single time-slice of processed data. The file extension suffix “.nc” refers to the output format from WPS which must be in netCDF for the “real.exe” program. For regional forecasts, multiple time periods must be processed by “real.exe” so that a lateral boundary file is available to the model. The global option for WRF requires only an initial condition.

The WPS package delivers data that is ready to be used in the WRF system by the “real.exe” program.
The data adheres to the WRF IO API. Unless you are developing special tools, stick with the netCDF option to communicate between the WPS package and “real.exe”.

The data has already been horizontally interpolated to the correct grid-point staggering for each variable, and the winds are correctly rotated to the WRF model map projection.

3-D meteorological data required from the WPS: pressure, u, v, temperature, relative humidity, geopotential height

Optional 3-D hydrometeor data may be provided to the real program at run-time, but these fields will not be used in the coarse-grid lateral boundary file. Fields named: QR, QC, QS, QI, QG, QH, QNI (mixing ratio for rain, cloud, snow, ice, graupel, hail, and number concentration) are eligible for input from the metgrid output files.

3D soil data from the WPS: soil temperature, soil moisture, soil liquid (optional, depending on physics choices in the WRF model)

2D meteorological data from the WPS: sea level pressure, surface pressure, surface u and v, surface temperature, surface relative humidity, input elevation

2-D meteorological optional data from WPS: sea surface temperature, physical snow depth, water equivalent snow depth

2D static data for the physical surface: terrain elevation, land use categories, soil texture categories, temporally-interpolated monthly data, land sea mask, elevation of the input model’s topography

2D static data for the projection: map factors, Coriolis, projection rotation, computational latitude

constants: domain size, grid distances, date

The WPS data may either be isobaric or some more-generalized vertical coordinate, where each column is monotonic in pressure

All 3-D meteorological data (wind, temperature, height, moisture, pressure) must have the same number of levels, and variables must have the exact same levels. For example, it is not acceptable to have more levels for temperature (for example) than height. Likewise, it is not acceptable to have an extra level for the horizontal wind components, but not for moisture.

Real Data Test Case: 2000 January 24/12 through 25/12

- A test data set is accessible from the WRF download page. Under the "WRF Model Test Data" list, select the January data. This is a 74x61, 30-km domain centered over the eastern US.
- Make sure you have successfully built the code (fine-grid nested initial data is available in the download, so the code may be built with the basic nest option), ./WRFV3/main/real.exe and ./WRFV3/main/wrf.exe must both exist.
- In the ./WRFV3/test/em_real directory, copy the namelist for the January case to the default name
INITIALIZATION

- `cp namelist.input.jan00 namelist.input`

- Link the WPS files (the “met_em*” files from the download) into the `.WRFV3/test/em_real` directory.
- For a single processor, to execute the real program, type `real.exe` (this should take less than a minute for this small case with five time periods).
- After running the “real.exe” program, the files “wrfinput_d01” and “wrfbdy_d01” should be in this directory; these files will be directly used by the WRF model.
- The “wrf.exe” program is executed next (type `wrf.exe`), this should only take a few minutes (only a 12-h forecast is requested in the namelist file).
- The output file `wrfout_d01:2000-01-24_12:00:00` should contain a 12-h forecast at 3-h intervals.

Considerations for Recent Releases

- Since a new simple ocean model has been included in the WRF code, the old namelist option for activating an ocean mixed layer is no longer suitable. The variable OMLCALL has been switched to SF_OCEAN_PHYSICS.
- The default behavior of the base state has been modified. Starting with release version 3.5, the isothermal temperature is no longer zero. With this change, the base state temperature no longer gets colder than 200 K (default in the Registry, though a user can override this option with a namelist setting). This fixes the problem associated with layers being too thick near the model top. A side effect of thinning-out these model layers is that users may need to increase the number of vertical levels.
- The common availability of a valid seaice field in the input provided from the metgrid program has made obsolete the option to autoconvert “cold enough” water points to seaice. By default, the temperature at which water converts to seaice is now 100 K, a temperature cold enough that the option will never be triggered.
Chapter 5: WRF Model

Table of Contents

- Introduction
- Installing WRF
- Running WRF
  - Idealized Case
  - Real Data Case
  - Restart Run
  - Two-Way Nested Runs
  - One-Way Nested Run Using ndown
  - Moving Nested Run
  - Three-dimensional Analysis Nudging
  - Observation Nudging
  - Global Run
  - DFI Run
  - SST Update
  - Using bucket_mm and bucket_J options
  - Adaptive Time Stepping
  - Stochastic Kinetic-Energy Backscatter Option
  - Run-Time IO
  - Output Time Series
  - Using IO Quilting
- Examples of namelists for various applications
- Check Output
- Trouble Shooting
- Physics and Dynamics Options
- Summary of PBL Physics Options
- Summary of Microphysics Options
- Summary of Cumulus Parameterization Options
- Summary of Radiation Options
- Description of Namelist Variables
- WRF Output Fields
- Special WRF Output Variables

Introduction

The WRF model is a fully compressible and nonhydrostatic model (with a run-time hydrostatic option). Its vertical coordinate is a terrain-following hydrostatic pressure coordinate. The grid staggering is the Arakawa C-grid. The model uses the Runge-Kutta 2nd and 3rd order time integration schemes, and 2nd to 6th order advection schemes in
both the horizontal and vertical. It uses a time-split small step for acoustic and gravity-wave modes. The dynamics conserves scalar variables.

The WRF model code contains an initialization program (either for real-data, real.exe, or idealized data, ideal.exe; see Chapter 4), a numerical integration program (wrf.exe), a program to do one-way nesting (ndown.exe), and a program to do tropical storm bogussing (tc.exe). The WRF model, Version 3, supports a variety of capabilities. These include

- Real-data and idealized simulations
- Various lateral boundary condition options for real-data and idealized simulations
- Full physics options, and various filter options
- Positive-definite advection scheme
- Non-hydrostatic and hydrostatic (runtime option)
- One-way and two-way nesting, and a moving nest
- Three-dimensional analysis nudging
- Observation nudging
- Regional and global applications
- Digital filter initialization

Other References

- See chapter 2 of this document for software requirement.

Installing WRF

Before compiling the WRF code on a computer, check to see if the netCDF library is installed. This is because one of the supported WRF I/O options is netCDF, and it is the one commonly used and supported by the post-processing programs. If the netCDF is installed in a directory other than /usr/local/, then find the path, and use the environment variable NETCDF to define where the path is. To do so, type

```
setenv NETCDF path-to-netcdf-library
```

Often the netCDF library and its include/ directory are collocated. If this is not the case, create a directory, link both netCDF lib and include directories in this directory, and use the environment variable to set the path to this directory. For example,

```
netcdf_links/lib -> /netcdf-lib-dir/lib
netcdf_links/include -> /where-include-dir-is/include
```
setenv NETCDF /directory-where-netcdf_links-is/netcdf_links

If the netCDF library is not available on the computer, it needs to be installed first. NetCDF source code or pre-built binary may be downloaded from, and installation instruction can be found on, the Unidata Web page at http://www.unidata.ucar.edu/.

Hint: for Linux users:

If PGI, Intel, gfortran or g95 compilers are used on a Linux computer, make sure netCDF is installed using the same compiler. Use the NETCDF environment variable to point to the PGI/Intel/g95 compiled netCDF library.

Hint: If using netCDF-4, make sure that the new capabilities (such as parallel I/O based on HDF5) are not activated at the install time, unless you intend to use the compression capability from netCDF-4 (supported in V3.5. More info below).

The WRF source code tar file can be downloaded from http://www.mmm.ucar.edu/wrf/users/download/get_source.html. Once the tar file is unzipped (gunzip WRFV3.TAR.gz), and untared (tar -xf WRFV3.TAR), it will create a WRFV3/ directory. This contains:

- Makefile
  Top-level makefile
- README
  General information about the WRF/ARW core
- README_test_cases
  Explanation of the test cases
- README.NMM
  General information for the WRF/NMM core
- README.DA
  General information for WRFDA
- README.rsl_output
  Information for dealing with rsl files
- README.io_config
  Information for runtime IO
- README.windtrubine
  Information on using wind farm parameterization
- README.hydro
  Information on WRF-Hydro
- Registry/
  Directory for WRF Registry files
- arch/
  Directory where compile options are gathered
- clean
  script to clean created files and executables
- compile
  script for compiling the WRF code
- configure
  script to create the configure.wrf file for compiling
- chem/
  WRF chemistry, supported by NOAA/GSD
- dyn_em/
  Directory for ARW dynamics and numerics
- dyn_exp/
  Directory for a 'toy' dynamic core
- dyn_nmm/
  Directory for NMM dynamics and numerics, supported by DTC
- external/
  Directory that contains external packages, such as those for IO, time keeping and MPI
The steps to compile and run the model are:

1. configure: generate a configuration file for compilation
2. compile: compile the code
3. run the model

Go to the WRFV3 (top) directory and type

./configure

and a list of choices for your computer should appear. These choices range from compiling for a single processor job (serial), to using OpenMP shared-memory (smpar) or distributed-memory parallelization (dmpar) options for multiple processors, or a combination of shared-memory and distributed-memory options (dm+sm). When a selection is made, a second choice for compiling nesting will appear. For example, on a Linux computer, the above steps may look like:

> setenv NETCDF /usr/local/netcdf-pgi
> ./configure

checking for perl5... no
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /usr/local/netcdf-pgi32
PHDF5 not set in environment. Will configure WRF for use without.
./configure: WRF operating system set to "Linux" via environment variable $WRF_OS
Will use 'time' to report timing information
$JASPERLIB or $JASPERINC not found in environment, configuring to build without grib2 I/O...

Please select from among the following Linux ARCH options:

1. (serial)  2. (smpar)  3. (dmpar)  4. (dm+sm)  5. (serial)  6. (smpar)  7. (dmpar)  8. (dm+sm)
Enter selection [1-90]:

Compile for nesting? (0=no nesting, 1=basic, 2=preset moves, 3=vortex following) [default 0]: 1

Enter the appropriate options that are best for your computer and application.

When the return key is hit, a configure.wrf file will be created. Edit compile options/paths, if necessary.

**Hint:** It is helpful to start with something simple, such as the serial build. If it is successful, move on to build smpar or dmpar code. Remember to type ‘clean –a’ between each build.
**Hint:** If you anticipate generating a netCDF file that is larger than 2Gb (whether it is a single- or multi-time period data [e.g. model history]) file, you may set the following environment variable to activate the large-file support option from netCDF (in c-shell):

```bash
setenv WRFIO_NCD_LARGE_FILE_SUPPORT 1
```

**Hint:** If you would like to use parallel netCDF (p-netCDF) developed by Argonne National Lab ([http://trac.mcs.anl.gov/projects/parallel-netcdf](http://trac.mcs.anl.gov/projects/parallel-netcdf)), you will need to install p-netCDF separately, and use the environment variable PNETCDF to set the path:

```bash
setenv PNETCDF path-to-pnetcdf-library
```

**Hint:** Since V3.5, compilation may take a bit longer due to the addition of the CLM4 module. If you do not intend to use the CLM4 land-surface model option, you can modify your `configure.wrf` file by removing `-DWRF_USE_CLM` from `ARCH_LOCAL`.

To compile the code, type

```bash
./compile
```

and the following choices will appear:

**Usage:**

```bash
compile wrf
```

compile `wrf` in run dir (Note, no real.exe, ndown.exe or ideal.exe generated)

or choose a test case (see README_test_cases for details):

```bash
compile em_b_wave
compile em_convrad (new in V3.7)
compile em_esmf_exp (example only)
compile em_grav2d_x
compile em_heldsuarez
compile em_hill2d_x
compile em_les
compile em_quarter_ss
compile em_real
compile em_seabreeze2d_x
compile em_squall2d_x
compile em_squall2d_y
compile em_tropical_cyclone
compile em_tropical_cyclone
compile exp_real (example of a toy solver)
compile nmm_real (NMM solver)
```

```bash
compile -h
```

help message
where em stands for the Advanced Research WRF dynamic solver (which currently is the 'Eulerian mass-coordinate' solver). Type one of the above to compile. When you switch from one test case to another, you must type one of the above to recompile. The recompile is necessary to create a new initialization executable (i.e. real.exe, and ideal.exe - there is a different ideal.exe for each of the idealized test cases), while wrf.exe is the same for all test cases.

If you want to remove all object files (except those in the external/ directory) and executables, type 'clean'.

Type 'clean -a' to remove built files in ALL directories, including configure.wrf (the original configure.wrf will be saved to configure.wrf.backup). This is recommended if you make any mistake during the process, or if you have edited the configure.wrf or Registry files.

*Hint:* If you have trouble compiling routines, like solve_em.F, you can try to run the configure script with the optional argument ‘-s’, i.e.

```
./configure -s
```

This will configure to compile solve_em.F and a few other routines with reduced optimization.

If you would like to turn off optimization for all the code, say during code development and debugging, you can run the configure script with option ‘-d’:

```
./configure -d
```

Beginning with V3.5, the compression function in netCDF4 is supported. This option will typically reduce the file size by more than 50%. It will require netCDF4 to be installed with the option --enable-netcdf-4. Before compiling WRF, you will need to set the environment variable NETCDF4. In a C-shell environment, type setenv NETCDF4 1, followed by 'configure' and 'compile'.

For more detailed information, visit:

**a. Idealized case**

For any 2D test case (labeled in the case names), serial or OpenMP (smpar) compile options must be used. Additionally, you must only choose the ‘0=no nesting’ option when you configure. For all other cases, you may use serial or parallel (dmpar) and nesting. Suppose you would like to compile and run the 2-dimensional squall case, type

```
./compile em_squall2d_x >& compile.log
```
After a successful compilation, you should have two executables created in the main/ directory: ideal.exe and wrf.exe. These two executables will be linked to the corresponding test/case_name and run/ directories. cd to either directory to run the model.

It is a good practice to save the entire compile output to a file. When the executables are not present, this output is useful to help diagnose the compile errors.

b. Real-data case

For a real-data case, type

./compile em_real >& compile.log &

When the compile is successful, it will create three executables in the main/directory: ndown.exe, real.exe and wrf.exe.

real.exe: for WRF initialization of real data cases
ndown.exe: for one-way nesting
wrf.exe: WRF model integration

Like in the idealized cases, these executables will be linked to the test/em_real and run/ directories. cd to one of these two directories to run the model.

Running WRF

One may run the model executables in either the run/ directory, or the test/case_name directory. In either case, one should see executables ideal.exe or real.exe (and ndown.exe), and wrf.exe, linked files (mostly for real-data cases), and one or more namelist.input files in the directory.

Hint: If you would like to run the model executables in a different directory, copy or link the files in the test/em_* directory to that directory, and run from there.

a. Idealized case

Suppose the test case em_squall2d_x is compiled. To run, type

cd test/em_squall2d_x

Edit the namelist.input file (see README.namelist in the WRFV3/run/ directory or its Web version) to change length of integration, frequency of output, size of domain, timestep, physics options, and other parameters.
If you see a script in the test case directory, called `run_me_first.csh`, run this one first by typing:

```
./run_me_first.csh
```

This links some physics data files that might be needed to run the case.

*Note: when running `em_fire`, you must copy everything from the ‘hill_simple’ directory into your current working directory in order for it to run correctly.

```
cp hill_simple/* .
```

To run the initialization program, type

```
./ideal.exe
```

This program will typically read an input sounding file located in that directory, and generate an initial condition file `wrfinput_d01`. All idealized cases do not require a lateral boundary file because of the boundary condition choices they use, such as the periodic option. If the job is run successfully, the last thing it prints should be: ‘wrf: SUCCESS COMPLETE IDEAL INIT’.

To run the model and save the standard output to a file, type

```
./wrf.exe >& wrf.out &
```

or for a 3D test case compiled with MPI (dmpar) option,

```
mpirun -np 4 ./wrf.exe
```

If successful, the wrf output file will be written to a file named `wrfout_d01_0001-01-01_00:00:00`.

Pairs of `rsl.out.*` and `rsl.error.*` files will appear with any MPI runs. These are standard output and error files. Note that the execution command for MPI runs may be different on different machines and for different MPI installation. Check the user manual.

If the model run is successful, the last thing printed in the ‘wrf.out’ or `rsl.*.0000` files should be: ‘wrf: SUCCESS COMPLETE WRF’. Output files `wrfout_d01_0001-01-01*` and `wrfrst*` should be present in the run directory, depending on how namelist variables are specified for output. The time stamp on these files originates from the start times in the namelist file.

**b. Real-data case**

To make a real-data case run, `cd` to the working directory by typing
cd test/em_real (or cd run)

Start with the namelist.input template file in the directory and edit it to match your case.

Running a real-data case requires successfully running the WRF Preprocessing System programs (or WPS). Make sure met_em.* files from WPS are seen in the run directory (either link or copy the files):

cd test/em_real
ls -l ../../../WPS/met_em*
ln -s ../../..WPS/met_em* .

Make sure you edit the variables in the &time_control and &domains sections of the namelist.input file (the description of the namelists can be found in the later part of this chapter):

&time_control
run_days = 0, 
run_hours = 24, 
run_minutes = 0, 
run_seconds = 0, 
start_year = 2000, 2000, 2000, 
start_month = 01, 01, 01, 
start_day = 24, 24, 24, 
start_hour = 12, 12, 12, 
end_year = 2000, 2000, 2000, 
end_month = 01, 01, 01, 
end_day = 25, 25, 25, 
end_hour = 12, 12, 12, 
interval_seconds = 21600 
input_from_file = .true., .true., .true., 
history_interval = 180, 60, 60, 
frames_per_outfile = 1000, 1000, 1000, 
/

&domains
max_dom = 1, 
e_we = 74, 112, 94, 
e_sn = 61, 97, 91, 
e_vert = 30, 30, 30, 
p_top_requested = 5000, 
um_metgrid_levels = 27, 
um_metgrid_soil_levels = 4, 
dx = 30000, 10000, 3333.33, 
dy = 30000, 10000, 3333.33, 
grid_id = 1, 2, 3, 
parent_id = 0, 1, 2, 
i_parent_start = 1, 31, 30, 
j_parent_start = 1, 17, 30, 
parent_grid_ratio = 1, 3, 3, 
parent_time_step_ratio = 1, 3, 3,
Make sure that the dates and dimensions of the domain match those set in WPS. If only one domain is used, namelists in other columns can be ignored.

Other options for use to assist vertical interpolation in &domains are:

```plaintext
interp_type = 2
extrap_type = 2
t_extrap_type = 2
lowest_lev_from_sfc = .false.
use_levels_below_ground = .true.
use_surface = .true.
lagrange_order = 1
force_sfc_in_vinterp = 1
zap_close_levels = 500
sfcp_to_sfcp = .false.
adjust_heights = .false.
smooth_cg_topo = .false.
```

To run the real-data initialization program, compiled using serial or OpenMP (smpar) options, type

```
./real.exe >& real.out
```

Successful completion of the job should have ‘real_em: SUCCESS EM_REAL INIT’ printed at the end of the real.out file. It should also produce wrfinput_d01 and wrfbdy_d01 files. In the real data case, both files are required.

Run the WRF model by typing

```
./wrf.exe
```

A successful run should produce one or several output files with names like `wrfout_d<domain>_<date>` (where `<domain>` represents domain ID, and `<date>` represents a date string with the format `yyyy-mm-dd_hh:mm:ss`). For example, if you start the model at 1200 UTC, January 24 2000, then your first output file should have the name:

```
wrfout_d01_2000-01-24_12:00:00
```

The time stamp on the file name is always the first time the output file is written. It is always good to check the times written to the output file by typing:

```
cdump -v Times wrfout_d01_2000-01-24_12:00:00
```

You may have other wrfout files, depending on the namelist options (how often you split the output files is determined by the namelist option `frames_per_outfile`). You may also create restart files if you have a restart frequency (`restart_interval in...`)
the namelist.input file) set within your total integration time. The restart file(s) should have the following naming convention

\texttt{wrfrst_d<domain>_<date>}

The time stamp on a restart file is the time at which that restart file is valid.

For DM (distributed memory) parallel systems, some form of the \texttt{mpirun} command will be needed to run the executables. For example, on a Linux cluster, the command to run MPI code, using 4 processors, may look like:

\texttt{mpirun -np 4 ./real.exe}
\texttt{mpirun -np 4 ./wrf.exe}

On some IBMs, the command for a batch job may be:

\texttt{poe ./real.exe}
\texttt{poe ./wrf.exe}

\texttt{or}

\texttt{mpirun.lsf ./wrf.exe (on NCAR’s yellowstone)}

\textbf{c. Restart Run}

A restart run allows a user to extend a run to a longer simulation period. It is effectively a continuous run made of several shorter runs. Hence the results at the end of one or more restart runs should be identical to a single run without any restart.

In order to do a restart run, one must first create a restart file. This is done by setting the namelist variable \texttt{restart_interval} (unit is in minutes) to be equal to or less than the simulation length in the first model run, as specified by \texttt{run_*} variables or \texttt{start_*} and \texttt{end_*} times. When the model reaches the time to write a restart file, a restart file named \texttt{wrfrst_d<domain>_<date>} will be written. The date string represents the time when the restart file is valid.

When one starts the restart run, edit the namelist.input file, so that your \texttt{start_*} time will be set to the restart time (which is the time the restart file is written). The other namelist variable one must set is \texttt{restart}, this variable should be set to \texttt{.true.} for a restart run.

In summary, these namelists should be modified:

\texttt{start_*}, \texttt{end_*}: start and end times for restart model integration
\texttt{restart}: logical to indicate whether the run is a restart or not
If the history and restart intervals are changed in a restart run, and the outcome isn’t what is expected to be, use namelist ‘override_restart_timers = .true.’.

If history output is desired at the time of restart, use namelist ‘write_hist_at_0h_rst = .true.’

**Hint:** Typically the restart file is a lot bigger in size than the history file, hence one may find that it is even ok to write a single model history output time to a file in netCDF format (frame_per_outfile=1), but it may fail to write a restart file. This is because the basic netCDF file support is only 2Gb. There are two solutions to the problem. The first is to simply set the namelist option io_form_restart = 102 (instead of 2), and this will force the restart file to be written into multiple pieces, one per processor. As long as one restarts the model using the same number of processors, this option works well (and one should restart the model with the same number of processors in any case). The second solution is to recompile the code using the netCDF large file support option (see the section on “Installing WRF” in this chapter).

d. Two-way Nested Runs

A two-way nested run is a run in which multiple domains at different grid resolutions are run simultaneously and communicate with each other: The coarser domain provides boundary values for the nest, and the nest feeds its calculation back to the coarser domain. The model can handle multiple domains at the same nest level (no overlapping nest), and multiple nest levels (telescoping).

When preparing for a nested run, make sure that the code is compiled with basic nest options (option 1).

Most of options to start a nest run are handled through the namelist. All variables in the namelist.input file that have multiple columns of entries need to be edited with caution. Start with a namelist template. The following are the key namelist variables to modify:

- start_*, end_*: start and end simulation times for the nest
- input_from_file: whether a nest requires an input file (e.g. wrfinput_d02). This is typically used for a real data case, since the nest input file contains nest topography and land information.
- fine_input_stream: which fields from the nest input file are used in nest initialization. The fields to be used are defined in the Registry.EM. Typically they include static fields (such as terrain and landuse), and masked surface fields (such as skin temperature, soil moisture and temperature). Useful for a nest starting at a later time than the coarse domain.
max_dom: the total number of domains to run. For example, if you want to have one coarse domain and one nest, set this variable to 2.

grid_id: domain identifier that is used in the wrfout naming convention. The most coarse grid must have grid_id of 1.

parent_id: used to indicate the parent domain of a nest. grid_id value is used.

i_parent_start/j_parent_start: lower-left corner starting indices of the nest domain in its parent domain. These parameters should be the same as in namelist.wps.

parent_grid_ratio: integer parent-to-nest domain grid size ratio. Typically an odd number ratio is used in real-data applications.

parent_time_step_ratio: integer time-step ratio for the nest domain. It may be different from the parent_grid_ratio, though they are typically set the same.

feedback: this is the key setup to define a two-way nested (or one-way nested) run. When feedback is on, the values of the coarse domain are overwritten by the values of the variables (average of cell values for mass points, and average of the cell-face values for horizontal momentum points) in the nest at the coincident points. For masked fields, only the single point value at the collocating points is fed back. If the parent_grid_ratio is even, an arbitrary choice of the southwest corner point value is used for feedback. This is the reason it is better to use an odd parent_grid_ratio with this option. When feedback is off, it is equivalent to a one-way nested run, since nest results are not reflected in the parent domain.

smooth_option: this a smoothing option for the parent domain in the area of the nest if feedback is on. Three options are available: 0 = no smoothing; 1 = 1-2-1 smoothing; 2 = smoothing-desmoothing.

3-D Idealized Cases

For 3-D idealized cases, no nest input files are required. The key here is the specification of the namelist.input file. What the model does is to interpolate all variables required in the nest from the coarse domain fields. Set

\[ \text{input_from_file} = \text{T, F}, \]

Real Data Cases

For real-data cases, three input options are supported. The first one is similar to running the idealized cases. That is to have all fields for the nest interpolated from the coarse domain (input_from_file = T, F). The disadvantage of this option is obvious: one will not benefit from the higher resolution static fields (such as terrain, landuse, and so on).
The second option is to set `input_from_file = T` for each domain, which means that the nest will have a nest wrfinput file to read in. The limitation of this option is that this only allows the nest to start at the same time as the coarse domain.

The third option is, in addition to setting `input_from_file = T` for each domain, also set `fine_input_stream = 2` for each domain. Why a value of 2? This is based on the Registry setting, which designates certain fields to be read in from the auxiliary input stream number 2. This option allows the nest initialization to use 3-D meteorological fields interpolated from the coarse domain, static fields and masked, and time-varying surface fields from the nest wrfinput; hence it allows a nest to start at a later time than hour 0. Setting `fine_input_stream = 0` is equivalent to the second option.

To run `real.exe` for a nested run, one must first run WPS and create data for all the nests. Suppose WPS is run for a 24 hour period, two-domain nested case starting at 1200 UTC Jan 24 2000. Then the following files should be generated in a WPS directory:

```
met_em.d01.2000-01-24_12:00:00
met_em.d01.2000-01-24_18:00:00
met_em.d01.2000-01-25_00:00:00
met_em.d01.2000-01-25_06:00:00
met_em.d01.2000-01-25_12:00:00
met_em.d02.2000-01-24_12:00:00
```

Typically only the first time period of the nest input file is needed to create a nest wrfinput file. Link or move all these files to the run directory.

Edit the `namelist.input` file and set the correct values for all relevant variables, described on the previous pages (in particular, set `max_dom = 2`, for the total number of domains to run), as well as physics options. Type the following to run:

```
./real.exe >& real.out
or
mpirun –np 4 ./real.exe
```

If successful, this will create all input files for coarse, as well as nested domains. For a two-domain example, these are created:

```
wrfinput_d01
wrfinput_d02
wrfbdy_d01
```

To run WRF, type

```
./wrf.exe
or
mpirun –np 4 ./wrf.exe
```
If successful, the model should create wrfout files for both domain 1 and 2:

wrfout_d01_2000-01-24_12:00:00
wrfout_d02_2000-01-24_12:00:00

e. One-way Nested Run Using ndown

WRF supports two separate one-way nested options. In this section, one-way nesting is defined as a finer-grid-resolution run, made as a subsequent run after the coarser-grid-resolution run, where the ndown program is run in-between the two simulations. The initial and lateral boundary conditions for this finer-grid run are obtained from the coarse grid run, with input from higher resolution terrestrial fields (e.g. terrain, landuse, etc.), and masked surface fields (such as soil temperature and moisture). The program that performs this task is ndown.exe.

*Note* that the use of this program requires the code to be compiled for nesting.

When one-way nesting is used, the coarse-to-fine grid ratio is only restricted to be an integer. An integer less than or equal to 5 is recommended. Frequent output (e.g. hourly) from the coarse grid run is also recommended to provide better boundary specifications. Do not change physics options when running ndown (you can change some physics options when making wrf runs), and do not remove fields from Registry if you plan to use ndown.

**Step 1:** Make a coarse grid run.

This is no different than any of the single-domain WRF runs, as described above. Do output history files as often as you can, and this will provide better and more frequent boundary conditions for the next model run.

**Step 2:** Run geogrid.exe (gives geo_em.d01 and geo_em.d02 files) and metgrid.exe for two domains (as if you are making a 2-way nested run).

This will generate WPS output files for domain 1 *(met_em.d01.<date>)* and domain 2 *(met_em.d02.<date>).*

**Step 3:** Run real.exe for 2 domains.

The purpose of this step is to ingest higher resolution terrestrial fields and corresponding land-water masked soil fields.

- Copy the met_em* files into the directory from which you will be running real.exe.
- Edit the namelist.input file, changing ‘max_dom = 2’, and making sure columns 1 and 2 are set-up for a 2 domain run, editing the correct start time and grid dimensions.
- Run real.exe. This will produce a wrfinput_d01 file, a wrfinput_d02
file, and a wrfbdy_d01 file.
- Rename the wrfinput_d02 file to wrfndi_d02.

**Step 4:** Make the final fine-grid initial and boundary condition files, by running ndown.exe

- Since V3.2, one must add `io_form_auxinput2 = 2` in the `&time_control` section of `namelist.input` to run `ndown.exe` successfully. (If one desires to refine the vertical resolution when running `ndown`, set `vert_refine_fact = integer` (new in V3.2). There are no other changes required in the namelist or in the procedure. Another way to refine vertical resolution is to use the utility program `v_interp` (see the chapter for ‘Utilities and Tools’ for details)).
- Change namelist variable `interval_seconds` to reflect the history output interval from the coarse domain model run.
- Do not change physics options until after running ndown program.
- Run `ndown.exe`, which uses input from the coarse grid `wrfout` file(s), and the `wrfndi_d02` file generated from Step 3 above. This will produce a `wrfinput_d02` and `wrfbdy_d02` file.

Note that the program `ndown` may be run serially or in MPI, depending on the selected compile option. The `ndown` program must be built to support nesting, however. To run the program, type

```
./ndown.exe
```

or
```
mpirun -np 4 ./ndown.exe
```

**Step 5:** Make the fine-grid WRF run

- Rename `wrfinput_d02` and `wrfbdy_d02` to `wrfinput_d01` and `wrfbdy_d01`, respectively.
- Rename (or move) the original `wrfout_d01*` files to something else (or another directory) so as to not overwrite them.
- Edit `namelist.input`, moving all of the fine-grid domain data from column 2 to column 1 so that this run will be for the fine-grid domain only. Make sure that the `time_step` is set to comply with the fine-grid domain (typically 6*DX). It may be beneficial to save `namelist.input` to something else prior to this step in case you need to repeat this process in the future. Save the newly-edited namelist as `namelist.input`.

- The WRF model’s physics options may be modified between runs (the WRF model before ndown and the WRF model after ndown, but do use the same physics from the first run when running ndown), except generally for the land surface scheme option
which has different number of soil depths depending on the scheme. Users may take advantage of a feature that allows both the initial and lateral boundaries to use the moist and scalar arrays (have_bcs_moist and have_bcs_scalar, respectively). This option is only to be used during the WRF model run which follows the ndown processing. With this option, a user must keep the microphysics options the same between forecasts. The advantage is that the previous WRF model provides realistic lateral boundary tendencies for all of the microphysical variables, instead of a simple “zero inflow” or “zero gradient outflow”.

- Run WRF for this grid.

*Keep in mind that the output from this run will be in the form wrfout_d01* but it will actually be output for domain 2. It may help to rename these to avoid future confusion.

The figure on the next page summarizes the data flow for a one-way nested run using the program ndown.
f. Moving-Nested Run

Two types of moving tests are allowed in WRF. In the first option, a user specifies the nest movement in the namelist. The second option is to move the nest automatically, based on an automatic vortex-following algorithm. This option is designed to follow the movement of a well-defined tropical cyclone.

To make the specified moving nested run, select the right nesting compile option (option ‘preset moves’). Note that code compiled with this option will not support static nested runs. To run the model, only the coarse grid input files are required. In this option, the nest initialization is defined from the coarse grid data - no nest input is used. In addition to the namelist options applied to a nested run, the following needs to be added to the namelist section &domains:

num_moves: the total number of moves one can make in a model run. A move of any domain counts against this total. The maximum is currently set to 50, but it can be changed by changing MAX_MOVES in frame/module_driver_constants.F.

move_id: a list of nest IDs, one per move, indicating which domain is to move for a given move.

move_interval: the number of minutes from the beginning of the run until a move is supposed to occur. The nest will move on the next time step after the specified instant of model time has passed.

move_cd_x, move_cd_y: distance in the number of grid points and direction of the nest move (positive numbers indicate moving toward east and north, while negative numbers indicate moving toward west and south).

Parameter max_moves is set to be 50, but can be modified in the source code file frame/module_driver_constants.F, if needed.

To make the automatic moving nested runs, select the ‘vortex-following’ option when configuring. Again note that this compile would only support the auto-moving nest, and will not support the specified moving nested run or static nested run at the same time. Again, no nest input is needed. If one wants to use values other than the default ones, add and edit the following namelist variables in the &domains section:

vortex_interval: how often the vortex position is calculated in minutes (default is 15 minutes).

max_vortex_speed: used with vortex_interval to compute the search radius for the new vortex center position (default is 40 m/sec).

corral_dist: the distance in the number of coarse grid cells that the moving nest is allowed to get near the mother domain boundary (default is 8). This parameter can be used to center the telescoped nests so that all nests are moved together with the storm.

track_level: the pressure level (in Pa) where the vortex is tracked.
**time_to_move**: the time (in minutes) to move a nest. This option may help with the case when the storm is still too weak to be tracked by the algorithm.

When the automatic moving nest is employed, the model dumps the vortex center location, with minimum mean sea-level pressure and maximum 10-m winds in a standard-out file (e.g. rsl.out.0000). Typing `grep ATCF rsl.out.0000` will produce a list of storm information at a 15-minute interval:

<table>
<thead>
<tr>
<th>Time</th>
<th>Pressure</th>
<th>Winds</th>
<th>Min Pressure</th>
<th>Max Winds</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-08-20_12:00:00</td>
<td>20.37</td>
<td>-81.80</td>
<td>929.7</td>
<td>133.9</td>
</tr>
<tr>
<td>2007-08-20_12:15:00</td>
<td>20.29</td>
<td>-81.76</td>
<td>929.3</td>
<td>133.2</td>
</tr>
</tbody>
</table>

In both types of moving-nest runs, the initial location of the nest is specified through `i_parent_start` and `j_parent_start` in the namelist.input file.

Since V3.6, a capability to incorporate high-resolution terrain and landuse input in a moving nest run (Chen, Shuyi S., Wei Zhao, Mark A. Donelan, James F. Price, Edward J. Walsh, 2007: The CBLAST-Hurricane Program and the Next-Generation Fully Coupled Atmosphere–Wave–Ocean Models for Hurricane Research and Prediction. *Bull. Amer. Meteor. Soc.*, 88, 311–317. doi: [http://dx.doi.org/10.1175/BAMS-88-3-311](http://dx.doi.org/10.1175/BAMS-88-3-311)). To activate this option,

- At compile time, one needs to set environment variable, TERRAIN_AND_LANDUSE to 1. In cshell,
  
  ```
  setenv TERRAIN_AND_LANDUSE 1
  ```

  followed by doing ‘configure’ and ‘compile’.

- At run time, add these namelists in &time_control:

  ```
  input_from_hires = .true., .true.,
  rsmas_data_path = “terrain_and_landuse_data_directory”
  ```

The automatic moving nest works best for a well-developed vortex.

**g. Analysis Nudging Runs (Upper-Air and/or Surface)**

Prepare input data to WRF as usual using WPS. If nudging is desired in the nest domains, make sure all time periods for all domains are processed in WPS. For surface-analysis nudging (new in Version 3.1), OBSGRID needs to be run after METGRID, and it will output a `wrfsfdda_d01` file that the WRF model reads for this option.

Set the following options before running `real.exe`, in addition to others described earlier (see the namelists in `examples.namelist` in the `test/em_real/` directory, for guidance):

```
grid_fdda = 1
```
grid_sfdda = 1

Run `real.exe` as before, and this will create, in addition to `wrfinput_d0*` and `wrfbdy_d01` files, a file named `wrffdda_d0*`. Other grid-nudging namelists are ignored at this stage, but it is good practice to fill them all in before one runs `real`. In particular, set

```
gffdda_inname    = "wrffdda_d<domain>"
gfdda_interval  = time interval of input data in minutes
gfdda_end_h     = end time of grid-nudging in hours
```

```
sfdda_inname    = "wrfsfdda_d<domain>"
sfdda_interval  = time interval of input data in minutes
sfdda_end_h     = end time of surface grid-nudging in hours
```


A different surface data nudging option is added in V3.8, and activated by setting

grid_sfdda = 2

This option nudges surface air temperature and water vapor mixing ratio similar to that with option 1, but uses the tendencies generated from the direct nudging approach to constrain surface sensible and latent heat fluxes, thus ensuring thermodynamic consistency between the atmosphere and land surface. This works with YSU PBL and Noah LSM. (Alapaty et al. JAMC, 2008)

Spectral Nudging is a new upper-air nudging option since Version 3.1. This selectively nudges the coarser scales only, but is otherwise set up the same way as grid-nudging. This option also nudges geopotential height. The wave numbers defined here are the number of waves contained in the domain, and the number is the maximum one that is nudged.

```
ggrid_fdda = 2
xwavenum   = 3
ywavenum   = 3
```

**h. Observation Nudging Run**

In addition to the usual input data preparation using WPS, station observation files are required. See the [Observation Nudging User's Guide](http://www2.mmm.ucar.edu/wrf/users/wrfv3.1/How_to_run_obs_fdda.html) and [http://www2.mmm.ucar.edu/wrf/users/wrfv3.1/How_to_run_obs_fdda.html](http://www2.mmm.ucar.edu/wrf/users/wrfv3.1/How_to_run_obs_fdda.html) for details. The observation file names expected by WRF are `OBS_DOMAIN101` for domain 1, and `OBS_DOMAIN201` for domain 2, etc.
Observation nudging is activated in the model by the following namelists in &fdda:

obs_nudge_opt = 1
fdda_start = 0 (obs nudging start time in minutes)
fdda_end = 360 (obs nudging end time in minutes)

and in &time_control

auxinput11_interval_s = 180, 180, 180, (set the interval to be small enough so that all observations will be checked)


i. Global Run

WRFV3 supports global capability. To make a global run, run WPS, starting with the namelist template namelist.wps.global. Set map_proj = ‘lat-lon’, and grid dimensions e_we and e_sn without setting dx and dy in namelist.wps. The geogrid program will calculate grid distances, and their values can be found in the global attribute section of geo_em.d01.nc file. Type ncdump -h geo_em.d01.nc to find out the grid distances, which will be needed in filling out WRF’s namelist.input file. Grid distances in x and y directions may be different, but it is best that they are set similarly or the same. WRF and WPS assume the earth is a sphere, and its radius is 6370 km. There are no restrictions on what to use for grid dimensions, but for effective use of the polar filter in WRF, the east-west dimension should be set to \(2^P\times3^Q\times5^R+1\) (where P, Q, and R are any integers, including 0).

Run the rest of the WPS programs as usual but only for one time period. This is because the domain covers the entire globe, and lateral boundary conditions are no longer needed.

Run the program real.exe as usual and for one time period only. The lateral boundary file wrfbdy_d01 is not needed.

Copy namelist.input.global to namelist.input, and edit it. Run the model as usual.

Note: since this is not a commonly-used configuration in the model, use it with caution. Not all physics and diffusion options have been tested with it, and some options may not work well with polar filters. Also, positive-definite and monotonic advection options do not work with polar filters in a global run because polar filters can generate negative
values of scalars. This implies, too, that WRF-Chem cannot be run with positive-definite and monotonic options in a global WRF setup.

As an extension to the global lat-lon grid, the regional domain can also be set using a lat-lon grid. To do so, one needs to set both grid dimensions, and grid distances in degrees. Again geogrid will calculate the grid distance, assuming the earth is a sphere and its radius is 6370 km. Find the grid distance in meters in the netCDF file, and use the value for WRF’s namelist.input file.

j. Using Digital Filter Initialization

Digital filter initialization (DFI) is a new option in V3. It is a way to remove initial model imbalance as, for example, measured by the surface pressure tendency. This might be important when one is interested in the 0 – 6 hour simulation/forecast. It runs a digital filter during a short model integration, backward and forward, and then starts the forecast. In WRF implementation, this is all done in a single job. With the V3.3 release, DFI can be used for multiple domains with concurrent nesting, with feedback disabled.

There is no special requirement for data preparation.

Start with the namelist template namelist.input.dfi. This namelist file contains an extra namelist record for DFI: &dfi_control. Edit it to match your case configuration. For a typical application, the following options are used:

- dfi_opt = 3 (Note: if doing a restart, this must be changed to 0)
- dfi_nfilter = 7 (filter option: Dolph)
- dfi_cutoff_seconds = 3600 (should not be longer than the filter window)

For time specification, it typically needs to integrate backward for 0.5 to 1 hour, and integrate forward for half of the time.

If option dfi_write_filtered_input is set to true, a filtered wrfinput file, wrfinput_INITIALIZED_d01, will be produced when you run wrf.

In Version 3.2, a constant boundary condition option is introduced for DFI. To use it, set constant_bc = 1 in &bdy_control

If a different time step is used for DFI, one may use time_step_dfi to set it.

k. Using the sst_update option

The WRF model physics do not predict sea-surface temperature, vegetation fraction, albedo or sea ice. For long simulations, the model provides an alternative to read-in the time-varying data and to update these fields. In order to use this option, one must have access to time-varying SST and sea ice fields. Twelve monthly values of vegetation fraction and albedo are available from the geogrid program. Once these fields are
processed via WPS, one may activate the following options in the namelist record &time_control before running the program real.exe and wrf.exe:

io_form_auxinput4 = 2
auxinput4_inname = "wrflowinp_d<domain>" (created by real.exe)
auxinput4_interval = 360, 360, 360,

and in &physics

sst_update = 1

Note that this option doesn’t work with sf_ocean_physics options.

I. Using bucket_mm and bucket_J options

These options are for long simulation rainfall accumulations and radiation budget accumulation terms (RAINC, RAINNC, ACSWUPT, ACLWDN, etc.). With 32-bit accuracy, adding small numbers to very large numbers loses accuracy as the accumulation term increases. For simulations of days to weeks, the accumulations are usually okay, but for months to years, this has the effect of truncating the additions, and especially small ones may be zeroed-out.

When these options are activated, part of the term is stored in an integer that increments by 1 each time the bucket value is reached, so we have two terms - RAINNC and I_RAINNC, where RAINNC now only contains the remainder. The total is retrieved from the output with total = RAINNC+bucket_mm*I_RAINNC. A reasonable bucket value may be based on a monthly accumulation such as 100 mm. Total precipitation equals RAINC + RAINNC, where

Total RAINNC = RAINNC+bucket_mm*I_RAINNC

Total RAINC = RAINC+bucket_mm*I_RAINC

The radiation accumulation terms (e.g., ACSWUPT) are in Joules/m^2, so that the mean value over a simulation period is the difference divided by the time between, giving W/m^2.

The bucket_J option is for these terms, and the typical value, based on a monthly accumulation, is 1.e9 J. Here the total is given by (ACSWUPT example - other radiative terms would follow the same equation concept):

total = ACSWUPT+bucket_J*I_ACSWUPT

m. Using Adaptive Time Stepping
Adaptive time stepping is a way to maximize the time step that the model can use while keeping the model numerically stable. The model time step is adjusted based on the domain-wide horizontal and vertical stability criterion (called the Courant-Friedrichs-Lewy (CFL) condition). The following set of values would typically work well.

\[
\begin{align*}
\text{use\_adaptive\_time\_step} &= .\text{true.} \\
\text{step\_to\_output\_time} &= .\text{true.} \; \text{(but nested domains may still be writing output at the desired time. Try to use \text{adjust\_output\_times} = .\text{true.} \; \text{to make up for this.)}} \\
\text{target\_cfl} &= 1.2, 1.2, 1.2, \\
\text{max\_step\_increase\_pct} &= 5, 51, 51, \; \text{(a large percentage value for the nest allows the time step for the nest to have more freedom to adjust)} \\
\text{starting\_time\_step} &= \text{the actual value or -l} \; \text{(which means 4*DX at start time)} \\
\text{max\_time\_step} &= \text{use fixed values for all domains, e.g. 8*DX} \\
\text{min\_time\_step} &= \text{use fixed values for all domains, e.g. 3*DX} \\
\text{adaptation\_domain}: \text{which domain is driving the adaptive time step}
\end{align*}
\]

Also see the description of these options in the list of namelist on page 5-43.

n. Option to stochastically perturb forecast

Stochastic kinetic-energy backscatter scheme (SKEBS)

Since Version 3.3, WRF has an option to stochastically perturb forecasts via a stochastic kinetic-energy backscatter scheme (SKEBS; Shutts, 2005, QJRMS). The scheme introduces temporally and spatially correlated perturbations to the rotational wind components $u$, $v$, and potential temperature $\theta$. An application and verification of this scheme to mesoscale ensemble forecast in the mid-latitudes is available in Berner et. al, 2011, Monthly Weather Review, 139, 1972—1995 (http://journals.ametsoc.org/doi/abs/10.1175/2010MWR3595.1).

SKEBS generates perturbation tendency fields $ru\_tendf\_stoch$ (in $m^2/s^3$), $rv\_tendf\_stoch$ ($m^2/s^3$), $rt\_tendf\_stoch$ (K/$s^2$) for $u$, $v$ and $\theta$, respectively. For new applications we recommend to output the magnitude and spatial patterns of these perturbation fields and compare them to the physics tendency fields for the same variables. Within the scheme, these perturbation fields are then coupled to mass and added to physics tendencies of $u$, $v$, and $\theta$. The stochastic perturbations fields for wind and temperature are controlled by the kinetic and potential energy they inject into the flow. The injected energy is expressed as backscattered dissipation rate for streamfunction and temperature respectively.

Since the scheme uses Fast Fourier Transforms (FFTs) provided in the library FFTPACK, we recommend the number of gridpoints in each direction to be product of small primes. If the number of gridpoints is a large prime in at least one of the directions, the computational cost may increase substantially. Multiple domains are supported by interpolating the forcing from the largest domain for which the scheme is turned on (normally the parent domain) down to all nested domain.
At present, default settings for the scheme have been thoroughly tested on synoptic and meso-scale domains over the mid-latitudes and as such offer a starting point. Relationships between backscatter amplitudes and perturbation fields for a given variable are not necessarily proportional due to the complexity of the scheme.

To facilitate adjustment of the scheme parameters to user-specific applications, many parameters previously hard-coded in the scheme have become namelist parameters in version WRFV3.6. Users wishing to adjust default settings are strongly advised to read details addressed in the technical document available at http://www.cgd.ucar.edu/~berner/skebs.html, which also contains details on version history, derivations, and examples.

Restart capability was added starting with version V3.7.

Further documentation is available at http://www.cgd.ucar.edu/~berner/skebs.html

The namelist parameters to control the stochastic schemes are found in the namelist.input file currently under the &stoch section (new namelist record since V3.6):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>skebs</strong></td>
<td>Replaces the namelist parameter stoch_forc_opt which was used up to V3.6. Latter is still maintained, but obsolete.</td>
</tr>
<tr>
<td><strong>tot_backscat_psi</strong></td>
<td>Total backscattered dissipation rate for streamfunction; Controls amplitude of rotational wind perturbations. Default value is 1.0E-5 m^2/s^3.</td>
</tr>
<tr>
<td><strong>tot_backscat_t</strong></td>
<td>Total backscattered dissipation rate for potential temperature; Controls amplitude of potential temperature perturbations. Default value is 1.0E-6 m^2/s^3.</td>
</tr>
<tr>
<td><strong>ztau_psi</strong></td>
<td>Decorrelation time (in s) for streamfunction perturbations. Default is 10800s (=3h).</td>
</tr>
<tr>
<td><strong>ztau_t</strong></td>
<td>Decorrelation time (in s) for potential temperature perturbations. Default 21600s.</td>
</tr>
<tr>
<td><strong>rexponent_psi</strong></td>
<td>Spectral slope for streamfunction perturbations. Default is -1.83 for a kinetic-energy forcing spectrum with slope -5/3.</td>
</tr>
<tr>
<td><strong>rexponent_t</strong></td>
<td>Spectral slope of potential temperature perturbations. Default is -1.83 for a potential energy forcing spectrum with slope -1.83^2.</td>
</tr>
<tr>
<td><strong>kminforc</strong></td>
<td>Minimal forcing wavenumber in longitude for streamfunction perturbations. Default is</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lminforc</td>
<td>Minimal forcing wavenumber in latitude for streamfunction perturbations. Default is 1.</td>
</tr>
<tr>
<td>kminforct</td>
<td>Minimal forcing wavenumber in longitude for potential temperature perturbations. Default is 1.</td>
</tr>
<tr>
<td>lminforct</td>
<td>Minimal forcing wavenumber in latitude for potential temperature perturbations. Default is 1.</td>
</tr>
<tr>
<td>kmaxforc</td>
<td>Maximal forcing wavenumber in longitude for streamfunction perturbations. Default is maximal possible wavenumbers determined by number of gridpoints in longitude.</td>
</tr>
<tr>
<td>lmaxforc</td>
<td>Maximal forcing wavenumber in latitude for streamfunction perturbations. Default is maximal possible wavenumbers determined by number of gridpoints in latitude.</td>
</tr>
<tr>
<td>kmaxforct</td>
<td>Maximal forcing wavenumber in longitude for potential temperature perturbations. Default is maximal possible wavenumbers determined by number of gridpoints in longitude.</td>
</tr>
<tr>
<td>lmaxforct</td>
<td>Maximal forcing wavenumber in latitude for potential temperature perturbations. Default is maximal possible wavenumbers determined by number of gridpoints in latitude.</td>
</tr>
<tr>
<td>zsigma2_eps</td>
<td>Noise variance in autoregressive process defining streamfunction perturbations.</td>
</tr>
<tr>
<td>zsigma2_eta</td>
<td>Noise variance in autoregressive process defining in potential temperature perturbations.</td>
</tr>
</tbody>
</table>
| skebs_vertstruc | Replaces the namelist parameter stoch_vertstruc_opt which was used up to V3.6. Latter is still maintained, but obsolete.  
= 0, 0, 0: Constant vertical structure of random pattern generator  
= 1, 1, 1: Random phase vertical structure with westward tilt |
| nens            | Seed for random number stream for both stochastic schemes. For ensemble forecasts this parameter needs to be different for |
each member. The seed is a function of initial start time to ensure different random number streams for forecasts starting from different initial times.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iseed_skebs</td>
<td>Seed for random number stream for skebs. Will be combined with seed nens signifying ensemble member number and initial start time to ensure different random number streams for forecasts starting from different initial times and for different ensemble members.</td>
</tr>
</tbody>
</table>

*(From Berner)*

**Option to generate a random perturbation field**
This option generates a random perturbations field. The stochastic pattern is centered on 0 and has a prescribed standard deviation. This pattern can be used for user-implemented perturbations of e.g., lower boundary conditions or parameters in the physic parameterizations. Currently it is only used to generate perturbations of the boundary tendencies of the WRF-CHEM. The namelist parameters to control the random perturbation option are found in the namelist.input file under the &stoch section (new namelist record since V3.6):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rand_perturb</td>
<td>= 0, 0, 0: No random perturbations = 1, 1, 1: Create random perturbation field</td>
</tr>
<tr>
<td>lengthscale_rand_pert</td>
<td>Random Perturbation Lengthscale (in m). Default is 500000m.</td>
</tr>
<tr>
<td>timescale_rand_pert</td>
<td>Temporal decorreleation of randomn field (in s). Default is 21600s.</td>
</tr>
<tr>
<td>gridpt_stddev_rand_pert</td>
<td>Standard deviation of random perturbation field at each gridpoint. Default is 0.03.</td>
</tr>
<tr>
<td>stddev_cutoff_rand_pert</td>
<td>Cutoff tails of perturbation patternf above this threshold standard deviation. Default is 3.0 (standard deviations).</td>
</tr>
<tr>
<td>rand_pert_vertstruc</td>
<td>Vertical structure for random perturbation field: 0=constant; 1=random phase with tilt</td>
</tr>
<tr>
<td>nens</td>
<td>Seed for random number stream. For ensemble forecasts this parameter needs to be different for each member. The seed is a function of initial start time to ensure different random number streams for forecasts starting from different initial times.</td>
</tr>
<tr>
<td>iseed_rand_pert</td>
<td>Seed for random number stream for rand_perturb. Will be combined with seed nens signifying ensemble member</td>
</tr>
</tbody>
</table>
number and initial start time to ensure different random number streams for forecasts starting from different initial times and for different ensemble members.

(From Berner)

**Option to perturb the boundary conditions**

This option allows for the addition of perturbations to the boundary tendencies for $u$ and $v$ wind components and potential temperature in WRF stand-alone runs. Users may provide a pattern or use the pattern generated by SKEBS.

The perturb_bdy option runs independently of SKEBS and as such may be run with or without the SKEB scheme, which operates solely on the interior grid. However, selecting perturb_bdy=1 will require the generation of a domain-size random array, thus computation time may increase.

Selecting perturb_bdy=2 will require the user to provide a pattern. Arrays are initialized and called: field_u_tend_perturb, field_v_tend_perturb, field_t_tend_perturb. These arrays will need to be filled with desired pattern in spec_bdytend_perturb in share/module_bc.F or spec_bdy_dry_perturb in dyn_em/module_bc_em.F.

The namelist parameters to control the perturb boundary conditions option are found in the namelist.input file under the &stoch section:

perturb_bdy = 0 : No boundary perturbations (default)
   = 1 : Use SKEBS pattern for boundary perturbations
   = 2 : Use other user provided pattern for boundary perturbations

(From Fossell, nee Smith)

**Option to perturb the boundary tendencies in WRF-CHEM**

This option adds random perturbations to the chemistry boundary tendencies in WRF-CHEM. The random perturbation pattern is controlled by all the parameters for the namelist option "rand_perturb", and stored as the variable named "RAND_PERT" in the wrfout file.

The perturb_chem_bdy option runs independently of RAND_PERTURB and as such may be run with or without the RAND_PERTURB scheme, which operates solely on the interior grid. However, selecting perturb_chem_bdy=1 will require the generation of a domain-size random array, thus computation time may increase.

perturb_chem_bdy = 0 : No boundary perturbations to tendencies in WRF-CHEM (default)
= 1 : Use RAND_PERTURB pattern for boundary perturbations

(From Ha)

0. Run-Time IO

With the release of WRF version 3.2, IO decisions may now be updated as a run-time option. Previously, any modification to the IO (such as which variable is associated with which stream) was handled via the Registry, and changes to the Registry always necessitate a cycle of clean -a, configure, and compile. This compile-time mechanism is still available and it is how most of the WRF IO is defined. However, should a user wish to add (or remove) variables from various streams, that capability is available as an option.

First, the user lets the WRF model know where the information for the run-time modifications to the IO is located. This is a text file (my_file_d01.txt), one for each domain, defined in the namelist.input file, located in the time_control namelist record.

```
&time_control
iofields_filename = "my_file_d01.txt", "my_file_d02.txt"
ignore_iofields_warning = .true.,
/
```

The contents of the text file associates a stream ID (0 is the default history and input) with a variable, and whether the field is to be added or removed. The state variables must already be defined in the Registry file. Following are a few examples:

```
-:h:0:RAINC,RAINNC
```

would remove the fields RAINC and RAINNC from the standard history file.

```
+:h:7:RAINC,RAINNC
```

would add the fields RAINC and RAINNC to an output stream #7.

The available options are:

+ or -, add or remove a variable

0–24, integer, which stream

i or h, input or history

field name in the Registry – this is the first string in quotes. Note: do not include any spaces in between field names.

It is not necessary to remove fields from one stream to insert them in another. It is OK to have the same field in multiple streams.
If you are interested in outputting variables into a new stream (i.e., not the default history stream 0), then the following namelist variables will also be necessary (example for stream 7):

auxhist7_outname = "yourstreamname_d<domain>_d<date>"
auxhist7_interval = 360, 360,
frames_per_auxhist7 = 1, 1,
io_form_auxhist7 = 2

The namelist variable, ignore_iofields_warning, tells the program what to do if it encounters an error in these user-specified files. The default value, .TRUE., is to print a warning message but continue the run. If set to .FALSE., the program will abort if there are errors in these user-specified files.

Note that any field that can be part of the optional IO (either the input or output streams) must already be declared as a state variable in the Registry. Care needs to be taken when specifying the names of the variables that are selected for the run-time IO. The "name" of the variable to use in the text file (defined in the namelist.input file) is the quoted string from the Registry file. Most of the WRF variables have the same string for the name of the variable used inside the WRF source code (column 3 in the Registry file, non-quoted, and not the string to use) and the name of the variable that appears in the netCDF file (column 9 in the Registry file, quoted, and that is the string to use).

**p. Output Diagnostics**

1. Time series output. To activate the option, a file called “tslist” must be present in the WRF run directory. The tslist file contains a list of locations defined by their latitude and longitude along with a short description and an abbreviation for each location. A sample file looks something like this:

```
#-----------------------------------------------#
# 24 characters for name | pfx | LAT  | LON  |
#-----------------------------------------------#
Cape Hallett              hallt -72.330  170.250
McMurdo Station           mcm  -77.851  166.713
```

The first three lines in the file are regarded as header information, and are ignored. Given a tslist file, for each location inside a model domain (either coarse or nested) a file containing time series variables at each model time step will be written with the name pfx.<domain>.TS, where pfx is the specified prefix for the location in the tslist file. The maximum number of time series locations is controlled by the namelist variable max_ts_locs in the namelist record &domains. The default value is 5. The time series output contains selected variables at the surface, including 2-m temperature, vapor mixing ratio, 10-m wind components, u and v, rotated to the earth coordinate, etc.. More information for time series output can be found in WRFV3/run/README.tslist.
Starting in V3.5, in addition to surface variables, vertical profiles of earth-relative U and V, potential temperature, water vapor, and geopotential height will also be output. The default number of levels in the output is 15, but can be changed with namelist variable max_ts_level.

2. Pressure level output. This is activated by adding a namelist record &diag, and set p_lev_diag = 1. The option can output U, V, wind speed, T, dew point T, RH and geopotential height at a number of pressure levels.

```plaintext
&diag
p_lev_diag = 1
num_press_level = 4,
press_levels = 85000, 70000, 50000, 20000,
```

The output goes to auxiliary output stream 23, so the following should be set in &time_control:

```plaintext
auxhist23_interval = 360, 360,
frames_per_auxhist23 = 100, 100,
io_form_auxhist23 = 2
```

3. nwp_diagnostics = 1 in &time_control. Convective storm diagnostics. This option outputs maximum 10 m wind speed, max helicity in 2 - 5 km layer, maximum vertical velocity in updraft and downdraft below 400 mb, mean vertical velocity in 2 - 5 km layer, and maximum column graupel in a time-window between history output times. The extra fields go to history file.

4. output_diagnostics = 1 in &time_control. Climate diagnostics. This option outputs 36 surface diagnostic variables: maximum and minimum, times when max and min occur, mean value, standard deviation of the mean for T2, Q2, TSK, U10, V10, 10 m wind speed, RAINCV, RAINNCV (the last two are time-step rain). The output goes to auxiliary output stream 3, and hence it needs the following:

```plaintext
auxhist3_outname = “wrfxtrm_d<domain>_<date>”
auxhist3_interval = 1440, 1440,
frames_per_auxhist3 = 100, 100,
io_form_auxhist3 = 2
```

Since this option computes the daily max and min, etc., it is advisable to do a restart at the daily interval.

5. do_avgflx_em = 1 in &dynamics. This option outputs history-time averaged, column-pressure coupled U, V and W for downstream transport models. If Grell-type of schemes is used, do_avg_cugd = 1 will output time-averaged convective mass-fluxes.

6. afwa_diag_opt = 1 in &afwa. Main control option to turn on weather diagnostics contributed by AFWA. Output goes to auxiliary stream 2. (see full documentation at...
http://www2.mmm.ucar.edu/wrf/users/docs/AFWA_Diagnostics_in_WRF.pdf). NOTE: These options cannot be used with OpenMP.

afwa_p_type_opt = 1 precipitation type
afwa_severe_opt = 1 severe weather diagnostics
afwa_vil_opt = 1 vertically integrated liquid
afwa_radar_opt = 1 radar
afwa_icing_opt = 1 icing
afwa_vis_opt = 1 visibility
afwa_cloud_opt = 1 cloud
afwa_therm_opt = 1 thermal index
afwa_turb_opt = 1 turbulence
afwa_buoy_opt = 1 buoyancy

7. Others in &physics

do_radar_ref = 1: compute radar reflectivity using microphysics-specific parameters in the model. Works for mp_physics = 2,4,6,7,8,10,14,16.

prec_acc_dt = 60: Time interval for outputing precipitation variables (rain from cumulus and microphysics schemes, and snow from microphysics scheme) (unit in minutes).

q. WRF-Hydro

This is a new capability in V3.5. It couples WRF model with hydrology processes (such as routing and channeling). Using WRF-Hydro requires a separate compile by using environment variable WRF_HYDRO. In c-shell environment, do

setenv WRF_HYDRO 1

before doing ‘configure’ and ‘compile’. Once WRF is compiled, copy files from hydro/Run/ directory to your working directory (e.g. test/em_real/). A separately prepared geogrid file is also required. Please refer the following web site for detailed information: http://www.ral.ucar.edu/projects/wrf_hydro/. (From W. Yu)

r. Using IO Quilting

This option allows a few processors to be set aside to be responsible for output only. It can be useful and performance-friendly if the domain size is large, and/or the time taken to write an output time is becoming significant when compared to the time taken to integrate the model in between the output times. There are two variables for setting the option:

nio_tasks_per_group: How many processors to use per IO group for IO quilting.
Typically 1 or 2 processors should be sufficient for this purpose.

`nio_groups`:
How many IO groups for IO. Default is 1.

*Note: This option is only used for wrf.exe. It does not work for real or ndown.

**Examples of namelists for various applications**

A few physics options sets (plus model top and the number of vertical levels) are provided here for reference. They may provide a good starting point for testing the model in your application. Also note that other factors will affect the outcome; for example, the domain setup, the distributions of vertical model levels, and input data.

a. 1 – 4 km grid distances, convection-permitting runs for a 1-3 day run (as used for the NCAR spring real-time convection forecast over the US in 2013):

   ```plaintext
   mp_physics                          = 8,
   ra_lw_physics                       = 4,
   ra_sw_physics                       = 4,
   radt                                = 10,
   sf_sfclay_physics                   = 2,
   sf_surface_physics                  = 2,
   bl_pbl_physics                      = 2,
   bldt                                = 0,
   cu_physics                          = 0,
   ptop_requested                      = 5000,
   e_vert                              = 40,
   ```

b. 10 – 20 km grid distances, 1-3 day runs (e.g., NCAR daily real-time runs over the US):

   ```plaintext
   mp_physics                          = 8,
   ra_lw_physics                       = 4,
   ra_sw_physics                       = 4,
   radt                                = 15,
   sf_sfclay_physics                   = 1,
   sf_surface_physics                  = 2,
   bl_pbl_physics                      = 1,
   bldt                                = 0,
   cu_physics                          = 3,
   cudt                                = 0,
   ptop_requested                      = 5000,
   e_vert                              = 39,
   ```
c. Cold region 10 – 30 km grid sizes (e.g. used in NCAR’s Antarctic Mesoscale Prediction System):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp_physics</td>
<td>4,</td>
</tr>
<tr>
<td>ra_lw_physics</td>
<td>4,</td>
</tr>
<tr>
<td>ra_sw_physics</td>
<td>2,</td>
</tr>
<tr>
<td>radt</td>
<td>15,</td>
</tr>
<tr>
<td>sf_sfclay.physics</td>
<td>2,</td>
</tr>
<tr>
<td>sf_surface.physics</td>
<td>2,</td>
</tr>
<tr>
<td>bl_pbl.physics</td>
<td>2,</td>
</tr>
<tr>
<td>bldt</td>
<td>0,</td>
</tr>
<tr>
<td>cu.physics</td>
<td>1,</td>
</tr>
<tr>
<td>cudt</td>
<td>5,</td>
</tr>
<tr>
<td>fractional_seaice</td>
<td>1,</td>
</tr>
<tr>
<td>seaice_threshold</td>
<td>0.0,</td>
</tr>
<tr>
<td>ptop_requested</td>
<td>1000,</td>
</tr>
<tr>
<td>e_vert</td>
<td>44,</td>
</tr>
</tbody>
</table>

d. Hurricane applications (e.g. 36, 12, and 4 km nesting used by NCAR’s real-time hurricane runs in 2012):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp_physics</td>
<td>6,</td>
</tr>
<tr>
<td>ra_lw_physics</td>
<td>4,</td>
</tr>
<tr>
<td>ra_sw_physics</td>
<td>4,</td>
</tr>
<tr>
<td>radt</td>
<td>10,</td>
</tr>
<tr>
<td>sf_sfclay.physics</td>
<td>1,</td>
</tr>
<tr>
<td>sf_surface.physics</td>
<td>2,</td>
</tr>
<tr>
<td>bl_pbl.physics</td>
<td>1,</td>
</tr>
<tr>
<td>bldt</td>
<td>0,</td>
</tr>
<tr>
<td>cu.physics</td>
<td>6,    (only on 36/12 km grid)</td>
</tr>
<tr>
<td>cudt</td>
<td>0,</td>
</tr>
<tr>
<td>isftcflx</td>
<td>2,</td>
</tr>
<tr>
<td>ptop_requested</td>
<td>2000,</td>
</tr>
<tr>
<td>e_vert</td>
<td>36,</td>
</tr>
</tbody>
</table>

e. Regional climate case at 10 – 30 km grid sizes (e.g. used in NCAR’s regional climate runs):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp_physics</td>
<td>6,</td>
</tr>
<tr>
<td>ra_lw_physics</td>
<td>3,</td>
</tr>
<tr>
<td>ra_sw_physics</td>
<td>3,</td>
</tr>
<tr>
<td>radt</td>
<td>30,</td>
</tr>
<tr>
<td>sf_sfclay.physics</td>
<td>1,</td>
</tr>
<tr>
<td>sf_surface.physics</td>
<td>2,</td>
</tr>
<tr>
<td>bl_pbl.physics</td>
<td>1,</td>
</tr>
<tr>
<td>bldt</td>
<td>0,</td>
</tr>
<tr>
<td>cu.physics</td>
<td>1,</td>
</tr>
<tr>
<td>cudt</td>
<td>5,</td>
</tr>
</tbody>
</table>
sst_update = 1,  
tmn_update = 1,  
sst_skin = 1,  
bucket_mm = 100.0,  
bucket_J = 1.e9,  
ptop_requested = 1000,  
e_vert = 51,  

spec_bdy_width = 10,  
spec_zone = 1,  
relax_zone = 9,  
spec_exp = 0.33,

**Check Output**

Once a model run is completed, it is good practice to check a couple of things quickly.

If you have run the model on multiple processors using MPI, you should have a number of rsl.out.* and rsl.error.* files. Type `tail rsl.out.0000` to see if you get ‘SUCCESS COMPLETE WRF’. This is a good indication that the model has run successfully.

The namelist options are written to a separate file: namelist.output.

Check the output times written to the wrfout* file by using the netCDF command:

```
ncdump -v Times wrfout_d01_yyyy-mm-dd_hh:00:00
```

Take a look at either the rsl.out.0000 file or other standard-out files. This file logs the times taken to compute for one model time step, and to write one history and restart output file:

```
Timing for main: time 2006-01-21 23:56:00 on domain 2: 4.73350 elapsed seconds.
Timing for main: time 2006-01-22 00:00:00 on domain 1: 19.55880 elapsed seconds.
```

and

```
Timing for Writing wrfout_d02_2006-01-22_00:00:00 for domain 2: 1.17970 elapsed seconds.
Timing for main: time 2006-01-22 00:00:00 on domain 1: 27.66230 elapsed seconds.
Timing for Writing wrfout_d01_2006-01-22_00:00:00 for domain 1: 0.60250 elapsed seconds.
```
Trouble Shooting

- If the model aborts very quickly, it is likely that either the computer memory is not large enough to run the specific configuration, or the input data have some serious problems. For the first potential issue, try to type ‘unlimit’ or ‘ulimit –s unlimited’ to see if more memory and/or stack size can be obtained.
- For OpenMP (smpar-compiled code), the stack size needs to be set large, but not unlimited. Unlimited stack size may crash the computer.
- To check if the input data are the problem, use ncview or another netCDF file browser to check the fields in the wrfinput files.
- Another frequent error seen is ‘module_configure: initial_config: error reading namelist’. This is an error message from the model complaining about errors and typos in the namelist.input file. Edit the namelist.input file with caution. If unsure, always start with an available template. A namelist record where the namelist read error occurs is provided in the V3 error message, and it should help with identifying the error.
- If the model did not run to completion, one possibility is that the model may have become numerically unstable, which means the time step used for advancing the model in time is too large for a stable solution. Even if one observes the standard rule for setting the model time step (to be ~ 6*DX in kilometers in physical space), other configurations of the model domain may affect the outcome. For example, if one has thin model layers, or if one uses a very large domain and the corners of the domain may have a very large map-scale factor that reduces the equivalent earth distance to be a lot smaller than the model grid size. One can find out whether this is the case by searching for CFL prints in the standard output/error files (e.g. the rsl files):

```
grep cfl rsl.error.* or grep cfl wrf.out
```

you might see something like these:

```
5 points exceeded cfl=2 in domain
MAX AT i,j,k: 123 48 3 cfl,w,d(eta)= 4.165821
21 points exceeded cfl=2 in domain
MAX AT i,j,k: 123 49 4 cfl,w,d(eta)= 10.66290
```

When this happens, consider using the namelist option w_damping, and/or reducing the time step.

Physics and Dynamics Options

Physics Options

WRF offers multiple physics options that can be combined in any way. The options typically range from simple and efficient, to sophisticated and more computationally costly, and from newly developed schemes, to well-tried schemes such as those in current operational models.
The choices vary with each major WRF release, but here we will outline those available in WRF Version 3.

1. **Microphysics (mp-physics)**
   a. Kessler scheme: A warm-rain (i.e. no ice) scheme used commonly in idealized cloud modeling studies ($mp\text{-physics} = 1$).
   b. Lin et al. scheme: A sophisticated scheme that has ice, snow and graupel processes, suitable for real-data high-resolution simulations (2).
   c. WRF Single-Moment 3-class scheme: A simple, efficient scheme with ice and snow processes suitable for mesoscale grid sizes (3).
   d. WRF Single-Moment 5-class scheme: A slightly more sophisticated version of (c) that allows for mixed-phase processes and super-cooled water (4).
   e. Eta microphysics: The operational microphysics in NCEP models. A simple efficient scheme with diagnostic mixed-phase processes. For fine resolutions (< 5km) use option (5) and for coarse resolutions use option (95).
   f. WRF Single-Moment 6-class scheme: A scheme with ice, snow and graupel processes suitable for high-resolution simulations (6).
   h. New Thompson et al. scheme: A new scheme with ice, snow and graupel processes suitable for high-resolution simulations (8). This adds rain number concentration and updates the scheme from the one in Version 3.0. New in Version 3.1.
   i. Milbrandt-Yau Double-Moment 7-class scheme (9). This scheme includes separate categories for hail and graupel with double-moment cloud, rain, ice, snow, graupel and hail. New in Version 3.2. *(Note: Do not use this scheme in V3.6 and V3.6.1.)*
   k. WRF Double-Moment 5-class scheme (14). This scheme has double-moment rain. Cloud and CCN for warm processes, but is otherwise like WSM5. New in Version 3.1.
   l. WRF Double-Moment 6-class scheme (16). This scheme has double-moment rain. Cloud and CCN for warm processes, but is otherwise like WSM6. New in Version 3.1.
   m. Stony Brook University (Y. Lin) scheme (13). This is a 5-class scheme with riming intensity predicted to account for mixed-phase processes. New in Version 3.3.
   n. NSSL 2-moment scheme (17, 18). New since Version 3.4, this is a two-moment scheme for cloud droplets, rain drops, ice crystals, snow, graupel, and hail. It also predicts average graupel particle density, which allows graupel to span the range from frozen drops to low-density graupel. There is an additional option to predict cloud condensation nuclei (CCN, option 18) concentration (intended for idealized simulations). The scheme is intended for cloud-resolving simulations (dx <= 2km) in research applications. Since V3.5, two more one-moment schemes have been added (19 and 21). Option 19 is a single-moment version of the NSSL scheme, and option 21...
is similar to Gilmore et al. (2004). Option 22 (new in V3.7) is the two moment scheme (option 17) without hail.

o. CAM V5.1 2-moment 5-class scheme.

p. Thompson aerosol-aware (28). This scheme considers water- and ice-friendly aerosols. A climatology dataset may be used to specify initial and boundary conditions for the aerosol variables (Thompson and Eidhammer, 2014, JAS.) New in Version 3.6.

q. HUJI (Hebrew University of Jerusalem, Israel) spectral bin microphysics, full (32) and ‘fast’ (30) versions are available since Version 3.6.

2.1 Longwave Radiation (ra_lw_physics)

a. RRTM scheme (ra_lwPhysics = 1): Rapid Radiative Transfer Model. An accurate scheme using look-up tables for efficiency. Accounts for multiple bands, and microphysics species. For trace gases, the volume-mixing ratio values for CO$_2$=330e-6, N$_2$O=0. and CH$_4$=0. in pre-V3.5 code; in V3.5, CO$_2$=379e-6, N$_2$O=319e-9 and CH$_4$=1774e-9. See section 2.3 for time-varying option.

b. GFDL scheme (99): Eta operational radiation scheme. An older multi-band scheme with carbon dioxide, ozone and microphysics effects.

c. CAM scheme (3): from the CAM 3 climate model used in CCSM. Allows for aerosols and trace gases. It uses yearly CO$_2$, and constant N$_2$O (311e-9) and CH$_4$ (1714e-9). See section 2.3 for the time-varying option.

d. RRTMG scheme (4): A new version of RRTM added in Version 3.1. It includes the MCICA method of random cloud overlap. For major trace gases, CO$_2$=379e-6, N$_2$O=319e-9, CH$_4$=1774e-9. See section 2.3 for the time-varying option. In V3.7, a fast version is introduced as option 24.


2.2 Shortwave Radiation (ra_sw_physics)

a. Dudhia scheme: Simple downward integration allowing efficiently for clouds and clear-sky absorption and scattering (ra_swPhysics = 1).

b. Goddard shortwave: Two-stream multi-band scheme with ozone from climatology and cloud effects (2).

c. GFDL shortwave: Eta operational scheme. Two-stream multi-band scheme with ozone from climatology and cloud effects (99).

d. CAM scheme: from the CAM 3 climate model used in CCSM. Allows for aerosols and trace gases (3).


g. Fu-Liou-Gu scheme (7). multiple bands, cloud and cloud fraction effects, ozone profile from climatology, can allow for aerosols. New in Version 3.4.

h. Held-Suarez relaxation. A temperature relaxation scheme designed for idealized tests only (31).

Related options:

- Slope and shading effects. \textit{slope_rad} = 1 modifies surface solar radiation flux according to terrain slope. \textit{topo_shad} = 1 allows for shadowing of neighboring grid cells. Use only with high-resolution runs with grid size less than a few kilometers. Since Version 3.2, these are available for all shortwave options.

- \textit{swrad_scat}: scattering turning parameter for ra\_sw\_physics = 1. Default value is 1, which is equivalent to 1.e-5 m^2/kg. When the value is greater than 1, it increases the scattering.

- \textit{swint_opt}: Interpolation of short-wave radiation based on the updated solar zenith angle between SW calls. Available since V3.5.1.

2.3 Input to radiation options

a. CAM Green House Gases: Provides yearly green house gases from 1765 to 2500. The option is activated by compiling WRF with the macro –DCLWRFGHG added in configure.wrf. Once compiled, CAM, RRTM and RRTMG long-wave schemes will see these gases. Five scenario files are available: from IPCC AR5: CAMtr\_volume\_mixing\_ratio\_RCP4.5, CAMtr\_volume\_mixing\_ratio\_RCP6, and CAMtr\_volume\_mixing\_ratio\_RCP8.5; from IPCC AR4: CAMtr\_volume\_mixing\_ratio\_A1B, and CAMtr\_volume\_mixing\_ratio\_A2. The default points to the RCP8.5 file. New in Version 3.5.

b. Climatological ozone and aerosol data for RRTMG: The ozone data is adapted from CAM radiation (ra\_*\_physics=3), and it has latitudinal (2.82 degrees), height and temporal (monthly) variation, as opposed to the default ozone used in the scheme that only varies with height. This is activated by the namelist option o3input = 2, which becomes the default option in V3.7. The aerosol data is based on Tegen et al. (1997), which has 6 types: organic carbon, black carbon, sulfate, sea salt, dust and stratospheric aerosol (volcanic ash, which is zero). The data also has spatial (5 degrees in longitude and 4 degrees in latitudes) and temporal (monthly) variations. The option is activated by the namelist option aer\_opt = 1. New in Version 3.5.

c. Aerosol input for RRTMG and Goddard radiation options (aer\_opt = 2). Either AOD or AOD plus Angstrom exponent, single scattering albedo, and cloud asymmetry parameter can be provided via constant values from namelist or 2D input fields via auxiliary input stream 15. Aerosol type can be set too. New in V3.6.

e. Effective cloud water, ice and snow radii from Thompson (since 3.5.1), WSM, WDM and NSSL microphysics schemes (new in V3.7) are used in RRTMG.

### 2.4 Cloud fraction option

icloud: = 1, use Xu-Randall method; = 2, use threshold method which gives either 0 or 1 cloud fraction; = 3, use a RH-based method that follows Sundqvist et al. (1989). The threshold of RH depends on grid sizes (new in V3.7, fixed in V3.8).

### 3.1 Surface Layer (sf_sfclay_physics)

a. MM5 similarity: Based on Monin-Obukhov with Carlson-Boland viscous sub-layer and standard similarity functions from look-up tables (sf_sfclay_physics = 91). In V3.7, the thermal and moisture roughness lengths (or exchange coefficients for heat and moisture) over ocean are changed to COARE 3 formula (Fairall et al. 2003)

b. Eta similarity: Used in Eta model. Based on Monin-Obukhov with Zilitinkevich thermal roughness length and standard similarity functions from look-up tables (2).


g. Revised MM5 surface layer scheme (option 11 prior to V3.6, renamed to option 1 since V3.6): Remove limits and use updated stability functions. New in Version 3.4. (Jimenez et al. MWR 2012). In V3.7, the code is sped up to give similar timing as with the old MM5 scheme. The thermal and moisture roughness lengths (or exchange coefficients for heat and moisture) over ocean are changed to COARE 3 formula (Fairall et al. 2003) in V3.7.

h. Other: iz0tlnd = 1 (works with sf_sfclay_physics = 1, 91, and 5), Chen-Zhang thermal roughness length over land, which depends on vegetation height, 0 = original thermal roughness length in each sfclay option. New in Version 3.2.

### 3.2 Land Surface (sf_surface_physics)

a. (1)5-layer thermal diffusion: Soil temperature only scheme, using five layers.

b. (2) Noah Land Surface Model: Unified NCEP/NCAR/AFWA scheme with soil temperature and moisture in four layers, fractional snow cover and frozen soil physics. New modifications are added in Version 3.1 to better represent processes over ice sheets and snow covered area.
- In V3.6, a sub-tiling option is introduced, and it is activated by namelist 
  \textit{sf\_surface\_mosaic} = 1, and the number of tiles in a grid box is defined by namelist 
  \textit{mosaic\_cat}, with a default value of 3.

c. (3) RUC Land Surface Model: RUC operational scheme with soil temperature and 
  moisture in six layers, multi-layer snow and frozen soil physics (3).

d. (7) Pleim-Xiu Land Surface Model. For a more detailed description of the PX LSM, 
  including pros/cons, best practices, and recent improvements, see 
  \url{http://www2.mmm.ucar.edu/wrf/users/docs/PX-ACM.pdf}

Two-layer scheme with vegetation and sub-grid tiling (7). New in Version 3.0: The 
Pleim-Xiu land surface model (PX LSM; Pleim and Xiu 1995; Xiu and Pleim 2001) 
was developed and improved over the years to provide realistic ground temperature, 
soil moisture, and surface sensible and latent heat fluxes in mesoscale meteorological 
models. The PX LSM is based on the ISBA model (Noilhan and Planton 1989), and 
includes a 2-layer force-restore soil temperature and moisture model. The top layer 
is taken to be 1 cm thick, and the lower layer is 99 cm. Grid aggregate vegetation and 
soil parameters are derived from fractional coverage of land use categories and soil 
texture types. There are two indirect nudging schemes that correct biases in 2-m air 
temperature and moisture by dynamic adjustment of soil moisture (Pleim and Xiu, 
2003) and deep soil temperature (Pleim and Gilliam, 2009).

Users should recognize that the PX LSM was primarily developed for retrospective 
simulation, where surface-based observations are available to inform the indirect soil 
nudging. While soil nudging can be disabled using the FDDA namelist.input setting 
"pxlsm\_soil\_nudge," little testing has been done in this mode, although some users 
have reported reasonable results. Gilliam and Pleim (2010) discuss the 
implementation in the WRF model and provide typical configurations for retrospective 
applications. If soil nudging is activated, modelers must use the \texttt{Observgrid objective re-} 
\texttt{analysis utility} to produce a surface nudging file with the naming convention 
"wrfsfdda\_d0*." Obsgrid takes WPS "met\_em*" files and LittleR observation files 
and produces the "wrfsfdda\_d0*" file. The PX LSM uses 2-m temperature and mixing 
ratio re-analyses from this file for the deep soil moisture and temperature nudging. If 
modelers want to test PX LSM in forecast mode with soil nudging activated, 
forecasted 2-m temperature and mixing ratio can be used with empty observation files 
to produce the "wrfsfdda\_d0*" files, using Obsgrid, but results will be tied to the 
governing forecast model.

f. (4) Noah-MP (multi-physics) Land Surface Model: uses multiple options for key 
land-atmosphere interaction processes. Noah-MP contains a separate vegetation 
canopy defined by a canopy top and bottom with leaf physical and radiometric 
properties used in a two-stream canopy radiation transfer scheme that includes shading 
effects. Noah-MP contains a multi-layer snow pack with liquid water storage and 
melt/refreeze capability and a snow-interception model describing loading/unloading, 
melt/refreeze, and sublimation of the canopy-intercepted snow. Multiple options are 
available for surface water infiltration and runoff, and groundwater transfer and 
storage including water table depth to an unconfined aquifer. Horizontal and vertical 
vegetation density can be prescribed or predicted using prognostic photosynthesis and
dynamic vegetation models that allocate carbon to vegetation (leaf, stem, wood and root) and soil carbon pools (fast and slow). New in Version 3.4. (Niu et al. 2011)

g. (8) SSiB Land Surface Model: This is the third generation of the Simplified Simple Biosphere Model (Xue et al. 1991; Sun and Xue, 2001). SSiB is developed for land/atmosphere interaction studies in the climate model. The aerodynamic resistance values in SSiB are determined in terms of vegetation properties, ground conditions and bulk Richardson number according to the modified Monin–Obukhov similarity theory. SSiB-3 includes three snow layers to realistically simulate snow processes, including destructive metamorphism, densification process due to snow load, and snow melting, which substantially enhances the model’s ability for the cold season study. To use this option, ra_lw_physics and ra_sw_physics should be set to either 1, 3, or 4. The second full model level should be set to no larger than 0.982 so that the height of that level is higher than vegetation height. New in Version 3.4.

h. Fractional sea-ice (fractional_seaice = 1). Treat sea-ice as fractional field. Require fractional sea-ice as input data. Data sources may include those from GFS or the National Snow and Ice Data Center (http://nsidc.org/data/seaice/index.html). Use XICE for Vtable entry instead of SEAICE. This option works with sf_sfclay_physics = 1, 2, 5, and 7, and sf_surface_physics = 2, 3, and 7 in the present release. New in Version 3.1.

i. (5) CLM4 (Community Land Model Version 4, Oleson et al. 2010; Lawrence et al. 2010): CLM4 was developed at the National Center for Atmospheric Research with many external collaborators and represents a state-of-the-science land surface process model. It contains sophisticated treatment of biogeophysics, hydrology, biogeochemistry, and dynamic vegetation. In CLM4, the land surface in each model grid cell is characterized into five primary sub-grid land cover types (glacier, lake, wetland, urban, and vegetated). The vegetated sub-grid consists of up to 4 plant functional types (PFTs) that differ in physiology and structure. The WRF input land cover types are translated into the CLM4 PFTs through a look-up table. The CLM4 vertical structure includes a single-layer vegetation canopy, a five-layer snowpack, and a ten-layer soil column. An earlier version of CLM has been quantitatively evaluated within WRF in Jin and Wen (2012; JGR Atmosphere), Lu and Kueppers (2012; JGR Atmosphere), and Subin et al. (2011; Earth Interactions) (from Jin). New in Version 3.5. Updated for 20/21 category MODIS landuse data in V3.6.

3.3 Urban Surface (sf_urban_physics – replacing old switch ucmcall)

a. Urban canopy model (1): 3-category UCM option with surface effects for roofs, walls, and streets. In V3.7, a green roof option is added.

b. BEP (2). Building Environment Parameterization: Multi-layer urban canopy model that allows for buildings higher than the lowest model levels. Only works with Noah LSM and Boulac and MYJ PBL options. New in Version 3.1.

c. BEM (3). Building Energy Model. Adds to BEP, building energy budget with heating and cooling systems. Works with same options as BEP. New in Version 3.2.

3.4 Lake Physics (sf_lake_physics)
a. CLM 4.5 lake model (1). The lake scheme was obtained from the Community Land Model version 4.5 (Oleson et al. 2013) with some modifications by Gu et al. (2013). It is a one-dimensional mass and energy balance scheme with 20-25 model layers, including up to 5 snow layers on the lake ice, 10 water layers, and 10 soil layers on the lake bottom. The lake scheme is used with actual lake points and lake depth derived from the WPS, and it also can be used with user defined lake points and lake depth in WRF (lake_min_elev and lakedepth_default). The lake scheme is independent of a land surface scheme and therefore can be used with any land surface scheme embedded in WRF. The lake scheme developments and evaluations were included in Subin et al. (2012) and Gu et al. (2013) (Subin et al. 2012: Improved lake model for climate simulations, J. Adv. Model. Earth Syst., 4, M02001. DOI:10.1029/2011MS000072; Gu et al. 2013: Calibration and validation of lake surface temperature simulations with the coupled WRF-Lake model. Climatic Change, 1-13, 10.1007/s10584-013-0978-y).

4. Planetary Boundary layer (bl_pbl_physics)

a. Yonsei University scheme: Non-local-K scheme with explicit entrainment layer and parabolic K profile in unstable mixed layer (bl_pbl_physics = 1).
   - topo_wind: = 1: Topographic correction for surface winds to represent extra drag from sub-grid topography and enhanced flow at hill tops (Jimenez and Dudhia, JAMC 2012). Works with YSU PBL only. New in Version 3.4.


c. MRF scheme: Older version of (a) with implicit treatment of entrainment layer as part of non-local-K mixed layer (99).


e. Quasi-Normal Scale Elimination PBL (4). A TKE-prediction option that uses a new theory for stably stratified regions (Available since 3.1). Daytime part uses eddy diffusivity mass-flux method with shallow convection (mfshconv = 1) which is added in Version 3.4.

   - icloud_bl: = 1, option to couple subgrid-scale clouds from MYNN to radiation
   - bl_mynn_cloupdf: = 1, Kuwano et al (2010); = 2, Chaboureau and Bechtold (2002, JAS, with mods, default)
   - bl_mynn_cloudmix: = 1, mixing cloud water and ice (qnc and qni are mixed when scalar_pblmix = 1)

The above three options are new in V3.8.


i. UW (Bretherton and Park) scheme (9). TKE scheme from CESM climate model. New in Version 3.3.


k. LES PBL: A large-eddy-simulation (LES) boundary layer is available in Version 3. For this, bl_pbl_physic = 0, isflx = 1, and sf_sfclay_physics and sf_surface_physics are selected. This uses diffusion for vertical mixing and must use diff_opt = 2, and km_opt = 2 or 3, see below. Alternative idealized ways of running the LESPBL are chosen with isflx = 0 or 2. New in Version 3.0.

l. Grenier-Bretherton-McCaa scheme (12). This is a TKE scheme. Tested in cloud-topped PBL cases. New in Version 3.5.

m. Shin-Hong scheme (11): Include scale dependency for vertical transport in convective PBL. Vertical mixing in the stable PBL and free atmosphere follows YSU. This scheme also has diagnosed TKE and mixing length output. New in V3.7.

5. Cumulus Parameterization (cu_physics)

a. Kain-Fritsch scheme: Deep and shallow convection sub-grid scheme using a mass flux approach with downdrafts and CAPE removal time scale (cu_physics = 1).

- kfeta_trigger = 1 – default trigger; = 2 – moisture-advection modulated trigger function [based on Ma and Tan (2009, Atmospheric Research)]. May improve results in subtropical regions when large-scale forcing is weak.


c. Grell-Devenyi (GD) ensemble scheme: Multi-closure, multi-parameter, ensemble method with typically 144 sub-grid members (moved to option 93 in V3.5).


e. Grell 3D is an improved version of the GD scheme that may also be used on high resolution (in addition to coarser resolutions) if subsidence spreading (option cugd_avedx) is turned on (5). New in Version 3.0.


j. Grell-Freitas (GF) scheme (3): An improved GD scheme that tries to smooth the transition to cloud-resolving scales, as proposed by Arakawa et al. (2004). New in Version 3.5.

k. Old Kain-Fritsch scheme: Deep convection scheme using a mass flux approach with downdrafts and CAPE removal time scale (99).

l. Multi-scale Kain-Fritsch scheme (11): using scale-dependent dynamic adjustment timescale, LCC-based entrainment. Also uses new trigger function based on Bechtold.

m. New Tiedtke scheme (16): this version is similar to the Tiedtke scheme used in REGCM4 and ECMWF cy40r1. New in V3.7, updated in V3.8.

n. Kain-Fritsch-Cumulus Potential scheme (10): this option modifies the KF ad-hoc trigger function with one linked to boundary layer turbulence via probability density function (PDFs) using cumulus potential scheme. The scheme also computes the cumulus cloud fraction based on the time-scale relevant for shallow cumuli. (Berg et al. 2013.) New in V3.8.

6. Shallow convection option (shcu_physics)

   a. ishallow = 1, shallow convection option on. Works together with Grell 3D scheme (cu_physics = 5) – will move to shcu_physics category in the future.

   b. UW (Bretherton and Park) scheme (2). Shallow cumulus option from CESM climate model with momentum transport. New in Version 3.3.

   c. GRIMS (Global/Regional Integrated Modeling System) scheme: it represents the shallow convection process by using eddy-diffusion and the pal algorithm, and couples directly to the YSU PBL scheme. New in Version 3.5.

7. Other physics options

   a. Options to use for tropical storm and hurricane applications:

      - sf_ocean_physics = 1 (renamed from omllcall in previous versions): Simple ocean mixed layer model (1): 1-D ocean mixed layer model following that of Pollard, Rhines and Thompson (1972). Two other namelist options are available to specify the initial mixed layer depth (although one may ingest real mixed layer depth data) (oml_hml0) and a temperature lapse rate below the mixed layer (oml_gamma). Since V3.2, this option works with all sf_surface_physics options.

      - sf_ocean_physics = 2: New in V3.5. 3D Price-Weller-Pinkel (PWP) ocean model based on Price et al. (1994). This model predicts horizontal advection, pressure gradient force, as well as mixed layer processes. Only simple initialization via namelist variables ocean_z, ocean_t, and ocean_s is available in V3.5.

      - isftcflx: Modify surface bulk drag (Donelan) and enthalpy coefficients to be more in line with recent research results of those for tropical storms and hurricanes. This
option also includes dissipative heating term in heat flux. It is only available for
\( sf_sfcly_physics = 1 \). There are two options for computing enthalpy coefficients:
\( isftcflx = 1 \): constant \( Z_0q \) (since V3.2) for heat and moisture; \( isftcflx = 2 \) Garratt
formulation, slightly different forms for heat and moisture.

b. Other options for long simulations (new in Version 3.1):
- \( tmn_update \): update deep soil temperature (1).
- \( sst_skin \): calculate skin SST based on Zeng and Beljaars (2005) (1)
- \( bucket_mm \): bucket reset value for water equivalent precipitation accumulations
  (value in mm, -1 = inactive).
- \( bucket_J \): bucket reset value for energy accumulations (value in Joules, -1 =
  inactive). Only works with CAM and RRTMG radiation (\( ra_lw_physics = 3 \) and 4
  and \( ra_sw_physics = 3 \) and 4) options.
- To drive WRF model with climate data that does not have leap year, there is a
  compile option to do that. Edit configure.wrf and
  add -DNO_LEAP CALENDAR to the macro ARCH_LOCAL.

c. Land model input options:
- \( usemonalb \): When set to .true., it uses monthly albedo fields from geogrid,
  instead of table values
- \( rdlai2d \): When set to .true., it uses monthly LAI data from geogrid (new in V3.6)
  and the field will also go to wrflowinp file if \( sst_update \) is 1.

d. \( gwd_opt \): Gravity wave drag option. Can be activated when grid size is greater than
  10 km. May be beneficial for simulations longer than 5 days and over a large domain
  with mountain ranges. It is recommended that this option is used only with unrotated
  lat/long (e.g. global) or Mercator projections because the input orographic sub-grid

e. \( windfarm_opt \): Wind turbine drag parameterization scheme. It represents sub-grid
effects of specified turbines on wind and TKE fields. The physical characteristics of
the wind farm is read in from a file and use of the manufacturers’ specification is
recommended. An example of the file is provided in run/wind-turbine-1.tbl. The
location of the turbines are read in from a file, windturbines.txt. See
README.windturbine in WRFV3/ directory for more detail. New in Version 3.3, and
in this version it only works with 2.5 level MYNN PBL option (\( bl_pbl_physics = 5 \)),
and updated in V3.6.

8. Physics sensitivity options

a. \( no_mp_heating \): When set to 1, it turns off latent heating from microphysics. When
using this option, \( cu_physics \) should be set to 0.
b. \( icloud \): When set to 0, it turns off cloud effect on optical depth in shortwave
radiation options 1, 4 and longwave radiation option 1, 4. Note since V3.6, this
namelist also controls which cloud fraction method to use for radiation.
c. \( isftcflx \): When set to 0, it turns off both sensible and latent heat fluxes from the
surface. This option works for sf_sfclay_physics = 1, 5, 7, 11.
d. ifsnow: When set to 0, it turns off snow effect in sf_surface_physics = 1.

Diffusion and Damping Options

Diffusion in WRF is categorized under two parameters: the diffusion option and the K option. The diffusion option selects how the derivatives used in diffusion are calculated, and the K option selects how the K coefficients are calculated. Note that when a PBL option is selected, vertical diffusion is done by the PBL scheme, and not by the diffusion scheme. In Version 3, vertical diffusion is also linked to the surface fluxes.

1.1 Diffusion Option (diff_opt)
   a. Simple diffusion: Gradients are simply taken along coordinate surfaces (diff_opt = 1).
   b. Full diffusion: Gradients use full metric terms to more accurately compute horizontal gradients in sloped coordinates (diff_opt = 2).

1.2 K Option (km_opt)
   Note that when using a PBL scheme, only options (a) and (d) below make sense, because (b) and (c) are designed for 3d diffusion.
   a. Constant: K is specified by namelist values for horizontal and vertical diffusion (km_opt = 1).
   b. 3d TKE: A prognostic equation for turbulent kinetic energy is used, and K is based on TKE (km_opt = 2).
   c. 3d Deformation: K is diagnosed from 3d deformation and stability following a Smagorinsky approach (km_opt = 3).
   d. 2d Deformation: K for horizontal diffusion is diagnosed from just horizontal deformation. The vertical diffusion is assumed to be done by the PBL scheme (km_opt = 4).

1.3 6th Order Horizontal Diffusion (diff_6th_opt)
   6th-order horizontal hyper diffusion (del^6) on all variables to act as a selective short-wave numerical noise filter. Can be used in conjunction with diff_opt. = 1: simple; = 2: positive definite. Option 2 is recommended.

1.4 Nonlinear Backscatter Anisotropic (NBA) (sfs_opt)
   Sub-grid turbulent stress option for momentum in LES applications. New in Version 3.2. sfs_opt = 1 diagnostic sub-grid stress to be used with diff_opt = 2 and km_opt = 2 or 3. sfs_opt = TKE sub-grid stress to be used with diff_opt = 2 and km_opt = 2.

2. Damping Options
   These are independently activated choices.
a. Upper Damping: Either a layer of increased diffusion ($damp\_opt = 1$) or a Rayleigh relaxation layer (2) or an implicit gravity-wave damping layer (3, new in Version 3.0), can be added near the model top to control reflection from the upper boundary.

b. Vertical velocity damping ($w\_damping$): For operational robustness, vertical motion can be damped to prevent the model from becoming unstable with locally large vertical velocities. This only affects strong updraft cores, so has very little impact on results otherwise.

c. Divergence Damping ($sm\_div$): Controls horizontally-propagating sound waves.

d. External Mode Damping ($em\_div$): Controls upper-surface (external) waves.

e. Time Off-centering ($epssm$): Controls vertically-propagating sound waves.

Advection Options

a. Horizontal advection orders for momentum ($h\_mom\_adv\_order$) and scalar ($h\_sca\_adv\_order$) can be 2nd to 6th, with 5th order being the recommended one.

b. Vertical advection orders for momentum ($v\_mom\_adv\_order$) and scalar ($v\_sca\_adv\_order$) can be 2nd and 6th, with 3rd order being the recommended one.

c. Monotonic transport (option 2, new in Version 3.1) and positive-definite advection option (option 1) can be applied to moisture ($moist\_adv\_opt$), scalar ($scalar\_adv\_opt$), chemistry variables ($chem\_adv\_opt$) and tke ($tke\_adv\_opt$). Option 1 replaces $pd\_moist = .true.$ etc. in previous versions.

d. WENO (weighted essentially non-oscillatory) (option 3 for 5th order WENO; option 4 for 5th order WENO with positive definite limiter): for moisture ($moist\_adv\_opt$), scalar ($scalar\_adv\_opt$), chemistry variables ($chem\_adv\_opt$) and TKE ($tke\_adv\_opt$). For momentum, $momentum\_adv\_opt = 3$.

Some notes about using monotonic and positive-definite advection options:

The positive-definite and monotonic options are available for moisture, scalars, chemical scalars and TKE in the ARW solver. Both the monotonic and positive-definite transport options conserve scalar mass locally and globally and are consistent with the ARW mass conservation equation. We recommend using the positive-definite option for moisture variables on all real-data simulations. The monotonic option may be beneficial in chemistry applications and for moisture and scalars in some instances.

When using these options there are certain aspects of the ARW integration scheme that should be considered in the simulation configuration.

(1) The integration sequence in ARW changes when the positive-definite or monotonic options are used. When the options are not activated, the timestep tendencies from the physics (excluding microphysics) are used to update the scalar mixing ratio at the same time as the transport (advection). The microphysics is computed, and moisture is updated, based on the transport+physics update. When the monotonic or positive definite options are activated, the scalar mixing ratio is first
updated with the physics tendency, and the new updated values are used as the
starting values for the transport scheme. The microphysics update occurs after
the transport update using these latest values as its starting point. It is important to
remember that for any scalars, the local and global conservation properties, positive
definiteness and monotonicity depend upon each update possessing these properties.

(2) Some model filters may not be positive definite.
   i. $\text{diff}_6\text{th}\_\text{opt} = 1$ is not positive definite nor monotonic. Use $\text{diff}_6\text{th}\_\text{opt} = 2$
      if you need this diffusion option ($\text{diff}_6\text{th}\_\text{opt} = 2$ is monotonic and positive-
definite). We have encountered cases where the departures from
      monotonicity and positive-definiteness have been very noticeable.
   ii. $\text{diff}\_\text{opt} = 1$ and $\text{km}\_\text{opt} = 4$ (a commonly-used real-data case mixing option)
      is not guaranteed to be positive-definite nor monotonic due to the variable
      eddy diffusivity, $K$. We have not observed significant departures from
      positive-definiteness or monotonicity when this filter is used with these
      transport options.
   iii. The diffusion option that uses a user-specified constant eddy viscosity is
        positive definite and monotonic.
   iv. Other filter options that use variable eddy viscosity are not positive definite or
        monotonic.

(3) Most of the model physics are not monotonic nor should they be - they represent
sources and sinks in the system. All should be positive definite, although we have not
examined and tested all options for this property.

(4) The monotonic option adds significant smoothing to the transport in regions
where it is active. You may want to consider turning off the other model filters for
variables using monotonic transport (filters such as the second and sixth order
horizontal filters). At present it is not possible to turn off the filters for the scalars but
not for the dynamics using the namelist - one must manually comment out the calls in
the solver.

Other Dynamics Options

a. The model can be run hydrostatically by setting the $\text{non}_\text{hydrostatic}$ switch to .false.
b. The Coriolis term can be applied to wind perturbation ($\text{pert}_\text{coriolis} = .true.$) only
   (idealized only).
c. For $\text{diff}\_\text{opt} = 2$ only, vertical diffusion may act on full fields (not just on
   perturbation from the 1D base profile ($\text{mix}_\text{full}\_\text{fields} = .true.$; idealized only).
d. To obtain more accurate solution with moisture, one can add
   
   $use\_q\_\text{diabatic}$: which considers moisture tendency from microphysics in small
   steps. This option could make time-step more restrictive.
   $use\_\text{theta}_\text{m}$: which considers moisture effect on pressure in small steps. The
   current implementation may cost a bit more to run.
Lateral Boundary Condition Options

a. Periodic (periodic_x / periodic_y): for idealized cases.

b. Open (open_xs, open_xe, open_ys, open_ye): for idealized cases.

c. Symmetric (symmetric_xs, symmetric_xe, symmetric_ys, symmetric_ye): for idealized cases.

d. Specified (specified): for real-data cases. The first row and column are specified with external model values (spec_zone = 1, and it should not change). The rows and columns in relax_zone have values blended from an external model and WRF. The value of relax_zone may be changed, as long as spec_bdy_width = spec_zone + relax_zone. This can be used with periodic_x in tropical channel simulations.

spec_exp: exponential multiplier for the relaxation zone ramp, used with a specified boundary condition. 0. = linear ramp, default; 0.33 = ~3*dx exp decay factor. This may be useful for long simulations.

e. Nested (nested): for real and idealized cases.

Summary of PBL Physics Options

<table>
<thead>
<tr>
<th>bl_pblPhysics</th>
<th>Scheme</th>
<th>Reference</th>
<th>Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>YSU</td>
<td>Hong, Noh and Dudhia (2006, MWR)</td>
<td>2004</td>
</tr>
<tr>
<td>2</td>
<td>MYJ</td>
<td>Janjic (1994, MWR)</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>GFS</td>
<td>Hong and Pan (1996, MWR)</td>
<td>2005</td>
</tr>
<tr>
<td>4</td>
<td>QNSE</td>
<td>Sukoriansky, Galperin and Perov (2005, BLM)</td>
<td>2009</td>
</tr>
<tr>
<td>5</td>
<td>MYNN2</td>
<td>Nakanishi and Niino (2006, BLM)</td>
<td>2009</td>
</tr>
<tr>
<td>6</td>
<td>MYNN3</td>
<td>Nakanishi and Niino (2006, BLM)</td>
<td>2009</td>
</tr>
<tr>
<td>7</td>
<td>ACM2</td>
<td>Pleim (2007, JAMC)</td>
<td>2008</td>
</tr>
<tr>
<td>8</td>
<td>BouLac</td>
<td>Bougeault and Lacarrere (1989, MWR)</td>
<td>2009</td>
</tr>
<tr>
<td>9</td>
<td>UW</td>
<td>Bretherton and Park (2009, JC)</td>
<td>2011</td>
</tr>
<tr>
<td>10</td>
<td>TEMF</td>
<td>Angevine, Jiang and Mauriten (2010, MWR)</td>
<td>2011</td>
</tr>
<tr>
<td>12</td>
<td>GBM</td>
<td>Grenier and Bretherton (2001, MWR)</td>
<td>2013</td>
</tr>
<tr>
<td>99</td>
<td>MRF</td>
<td>Hong and Pan (1996, MWR)</td>
<td>2000</td>
</tr>
<tr>
<td>11</td>
<td>Shin-Hong</td>
<td>Shin and Hong (2015, MWR)</td>
<td>2015</td>
</tr>
<tr>
<td>bl_pbl_physics</td>
<td>Scheme</td>
<td>Cores</td>
<td>sf_sfclay_physics</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>-------------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>YSU</td>
<td>ARW/NMM</td>
<td>1, (91)*</td>
</tr>
<tr>
<td>2</td>
<td>MYJ</td>
<td>ARW/NMM</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>GFS (hwrf)</td>
<td>NMM</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>QNSE-EDMF</td>
<td>ARW/NMM</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>MYNN2</td>
<td>ARW</td>
<td>1,2,5,(91)</td>
</tr>
<tr>
<td>6</td>
<td>MYNN3</td>
<td>ARW</td>
<td>1,2,5,(91)</td>
</tr>
<tr>
<td>7</td>
<td>ACM2</td>
<td>ARW</td>
<td>1,7,(91)</td>
</tr>
<tr>
<td>8</td>
<td>BouLac</td>
<td>ARW</td>
<td>1,2,(91)</td>
</tr>
<tr>
<td>9</td>
<td>UW</td>
<td>ARW</td>
<td>1,2,(91)</td>
</tr>
<tr>
<td>10</td>
<td>TEMF</td>
<td>ARW</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>GBM</td>
<td>ARW</td>
<td>1,(91)</td>
</tr>
<tr>
<td>99</td>
<td>MRF</td>
<td>ARW/NMM</td>
<td>1,(91)</td>
</tr>
<tr>
<td>11</td>
<td>Shin-Hong</td>
<td>ARW</td>
<td>1,(91)</td>
</tr>
</tbody>
</table>

* sfclay option 11 was renamed to 1, and original option 1 to 91 in Version 3.6.
### Summary of Microphysics Options

<table>
<thead>
<tr>
<th>mp_physics Scheme</th>
<th>Reference</th>
<th>Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kessler</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>Lin (Purdue)</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>WSM3</td>
<td>2004</td>
</tr>
<tr>
<td>4</td>
<td>WSM5</td>
<td>2004</td>
</tr>
<tr>
<td>5</td>
<td>Eta (Ferrier)</td>
<td>2000</td>
</tr>
<tr>
<td>6</td>
<td>WSM6</td>
<td>2004</td>
</tr>
<tr>
<td>7</td>
<td>Goddard</td>
<td>2008</td>
</tr>
<tr>
<td>8</td>
<td>Thompson</td>
<td>2009</td>
</tr>
<tr>
<td>9</td>
<td>Milbrandt 2-mom</td>
<td>2010</td>
</tr>
<tr>
<td>10</td>
<td>Morrison 2-mom</td>
<td>2008</td>
</tr>
<tr>
<td>11</td>
<td>CAM 5.1</td>
<td>2013</td>
</tr>
<tr>
<td>13</td>
<td>SBU-YLin</td>
<td>2011</td>
</tr>
<tr>
<td>14</td>
<td>WDM5</td>
<td>2009</td>
</tr>
<tr>
<td>16</td>
<td>WDM6</td>
<td>2009</td>
</tr>
<tr>
<td>17</td>
<td>NSSL 2-mom</td>
<td>2012</td>
</tr>
<tr>
<td>18</td>
<td>NSSL 2-mom w/ CCN prediction</td>
<td>2012</td>
</tr>
<tr>
<td>19</td>
<td>NSSL 1-mom</td>
<td>2013</td>
</tr>
<tr>
<td>21</td>
<td>NSSL 1-momlfo</td>
<td>2013</td>
</tr>
<tr>
<td>22</td>
<td>NSSL 2-mom w/o hail</td>
<td>2015</td>
</tr>
<tr>
<td>28</td>
<td>Thompson aerosol-aware</td>
<td>2014</td>
</tr>
<tr>
<td>30</td>
<td>HUJI SBM ‘fast’</td>
<td>2014</td>
</tr>
<tr>
<td>32</td>
<td>HUJI SBM full</td>
<td>2014</td>
</tr>
<tr>
<td>mp_physics</td>
<td>Scheme</td>
<td>Cores</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>Kessler</td>
<td>ARW</td>
</tr>
<tr>
<td>2</td>
<td>Lin (Purdue)</td>
<td>ARW (Chem)</td>
</tr>
<tr>
<td>3</td>
<td>WSM3</td>
<td>ARW</td>
</tr>
<tr>
<td>4</td>
<td>WSM5</td>
<td>ARW/NMM</td>
</tr>
<tr>
<td>5</td>
<td>Eta (Ferrier)</td>
<td>ARW/NMM</td>
</tr>
<tr>
<td>6</td>
<td>WSM6</td>
<td>ARW/NMM</td>
</tr>
<tr>
<td>8</td>
<td>Thompson</td>
<td>ARW/NMM</td>
</tr>
<tr>
<td>9</td>
<td>Milbrandt 2-mom</td>
<td>ARW</td>
</tr>
<tr>
<td>10</td>
<td>Morrison 2-mom</td>
<td>ARW (Chem)</td>
</tr>
<tr>
<td>11</td>
<td>CAM 5.1</td>
<td>ARW</td>
</tr>
<tr>
<td>13</td>
<td>SBU-YLin</td>
<td>ARW</td>
</tr>
<tr>
<td>14</td>
<td>WDM5</td>
<td>ARW</td>
</tr>
<tr>
<td>16</td>
<td>WDM6</td>
<td>ARW</td>
</tr>
<tr>
<td>17</td>
<td>NSSL 2-mom</td>
<td>ARW</td>
</tr>
<tr>
<td>18</td>
<td>NSSL 2-mom +CCN</td>
<td>ARW</td>
</tr>
<tr>
<td>19</td>
<td>NSSL 1-mom</td>
<td>ARW</td>
</tr>
<tr>
<td>21</td>
<td>NSSL 1-momlflo</td>
<td>ARW</td>
</tr>
<tr>
<td>22</td>
<td>/nssl 2-mom</td>
<td>ARW</td>
</tr>
<tr>
<td>28</td>
<td>Thompson aerosol-aware</td>
<td>ARW/NMM</td>
</tr>
<tr>
<td>30</td>
<td>HUJI fast</td>
<td>ARW</td>
</tr>
<tr>
<td>32</td>
<td>HUJI full</td>
<td>ARW</td>
</tr>
</tbody>
</table>

* Adveacts only total condensates  ** Nn = CCN number  *** Vg: graupel volume
## Summary of Cumulus Parameterization Options

<table>
<thead>
<tr>
<th>cu_physics</th>
<th>Scheme</th>
<th>Reference</th>
<th>Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kain-Fritsch</td>
<td>Kain (2004, JAM)</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>Grell-Freitas</td>
<td>Grell et al. (2013)</td>
<td>2013</td>
</tr>
<tr>
<td></td>
<td>Old Simplified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arakawa-Schubert</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Grell-3</td>
<td></td>
<td>2008</td>
</tr>
<tr>
<td>6</td>
<td>Tiedtke</td>
<td>Tiedtke (1989, MWR), Zhang et al. (2011, MWR)</td>
<td>2011</td>
</tr>
<tr>
<td>7</td>
<td>Zhang-McFarlane</td>
<td>Zhang and McFarlane (1995, AO)</td>
<td>2011</td>
</tr>
<tr>
<td>10</td>
<td>KF-CuP</td>
<td>Berg et al. (2013, MWR)</td>
<td>2016</td>
</tr>
<tr>
<td>11</td>
<td>Multi-scale KF</td>
<td>Zheng et al. (2015, MWR)</td>
<td>2015</td>
</tr>
<tr>
<td>14</td>
<td>New SAS</td>
<td>Han and Pan (2011, Wea. Forecasting)</td>
<td>2011</td>
</tr>
<tr>
<td>84</td>
<td>New SAS (HWRF)</td>
<td>Han and Pan (2011, Wea. Forecasting)</td>
<td>2012</td>
</tr>
<tr>
<td>93</td>
<td>Grell-Devenyi</td>
<td>Grell and Devenyi (2002, GRL)</td>
<td>2002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cu_physics</th>
<th>Scheme</th>
<th>Cores</th>
<th>Moisture Tendencies</th>
<th>Momentum Tendencies</th>
<th>Shallow Convection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kain-Fritsch</td>
<td>ARW / NMM</td>
<td>Qc Qr Qi Qs</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>BMJ</td>
<td>ARW / NMM</td>
<td>-</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>GF</td>
<td>ARW</td>
<td>Qc Qi</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>OSAS</td>
<td>ARW / NMM</td>
<td>Qc Qi</td>
<td>yes (NMM)</td>
<td>yes (ARW)</td>
</tr>
</tbody>
</table>
### Summary of Radiation Physics Options

<table>
<thead>
<tr>
<th>ra_sw Physics</th>
<th>Scheme</th>
<th>Reference</th>
<th>Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dudhia</td>
<td>Dudhia (1989, JAS)</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>CAM</td>
<td>Collins et al. (2004, NCAR Tech Note)</td>
<td>2006</td>
</tr>
<tr>
<td>4</td>
<td>RRTMG</td>
<td>Iacono et al. (2008, JGR)</td>
<td>2009</td>
</tr>
<tr>
<td>24</td>
<td>RRTMG</td>
<td>Fast version</td>
<td>2015</td>
</tr>
<tr>
<td>7</td>
<td>FLG</td>
<td>Fu et al. (2011, JGR), Fu and Liou (1992, JAS)</td>
<td>2012</td>
</tr>
<tr>
<td>99</td>
<td>GFDL</td>
<td>Fels and Schwarzkopf (1981, JGR)</td>
<td>2004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ra_sw_ physics</th>
<th>Scheme</th>
<th>Cores+Chem</th>
<th>Microphysics Interaction</th>
<th>Cloud Fraction</th>
<th>Ozone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dudhia ARW NMM + Chem(PM2.5)</td>
<td>Qc Qr Qi Qs Qg</td>
<td>1/0</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>GSFC ARW+Chem(τ)</td>
<td>Qc Qi</td>
<td>1/0</td>
<td>5 profiles</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CAM ARW</td>
<td>Qc Qi Qs</td>
<td>max-rand</td>
<td>lat/month</td>
<td></td>
</tr>
</tbody>
</table>
### ra_lw_physics

<table>
<thead>
<tr>
<th>ra_lw_physics</th>
<th>Scheme</th>
<th>Cores+Chem</th>
<th>Microphysics Interaction</th>
<th>Cloud Fraction</th>
<th>Ozone</th>
<th>GHG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RRTM</td>
<td>ARW NMM</td>
<td>Qc Qr Qi Qs Qg</td>
<td>1/0</td>
<td>1 profile</td>
<td>constant or yearly GHG</td>
</tr>
<tr>
<td>3</td>
<td>CAM</td>
<td>ARW</td>
<td>Qc Qi Qs</td>
<td>max-rand overlap</td>
<td>lat/month</td>
<td>yearly CO2 or yearly GHG</td>
</tr>
<tr>
<td>4</td>
<td>RRTMG</td>
<td>ARW + Chem (τ), NMM</td>
<td>Qc Qr Qi Qs</td>
<td>max-rand overlap</td>
<td>1 profile</td>
<td>constant or yearly GHG</td>
</tr>
<tr>
<td>24</td>
<td>RRTMG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>New Goddard</td>
<td>ARW</td>
<td>Qc Qr Qi Qs Qg</td>
<td>1/0</td>
<td>5 profiles</td>
<td>constant</td>
</tr>
<tr>
<td>7</td>
<td>FLG</td>
<td>ARW</td>
<td>Qc Qr Qi Qs Qg</td>
<td>1/0</td>
<td>5 profiles</td>
<td>constant</td>
</tr>
<tr>
<td>31</td>
<td>Held-Suarez</td>
<td>ARW</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>GFDL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suarez</td>
<td>GFDL</td>
<td>ARW NMM</td>
<td>Qc Qr Qi Qs</td>
<td>max-rand overlap</td>
<td>lat/date</td>
<td>constant</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>---------</td>
<td>-------------</td>
<td>-----------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Description of Namelist Variables

The following is a description of the namelist variables. The variables that are a function of nests are indicated by \((\text{max}\_\text{dom})\) following the variable. Also see the Registry/Registry.EM and run/README.namelist files in the WRFV3/ for more detailed information.

<table>
<thead>
<tr>
<th>Variable Names</th>
<th>Input Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;time_control options for time control run_days</td>
<td>1</td>
<td>run time in days</td>
</tr>
<tr>
<td>run_hours</td>
<td>0</td>
<td>run time in hours *note: if it is more than 1 day, you may use both run_days and run_hours or just run_hours. e.g. if the total run length is 36 hrs, you may set run_days = 1, and run_hours = 12, or run_days = 0, and run_hours = 36</td>
</tr>
<tr>
<td>run_minutes</td>
<td>0</td>
<td>run time in minutes</td>
</tr>
<tr>
<td>run_seconds</td>
<td>0</td>
<td>run time in seconds</td>
</tr>
<tr>
<td>start_year (max_dom)</td>
<td>2012</td>
<td>4 digit year of starting time</td>
</tr>
<tr>
<td>start_month (max_dom)</td>
<td>06</td>
<td>2 digit month of starting time</td>
</tr>
<tr>
<td>start_day (max_dom)</td>
<td>11</td>
<td>2 digit day of starting time</td>
</tr>
<tr>
<td>start_hour (max_dom)</td>
<td>12</td>
<td>2 digit hour of starting time</td>
</tr>
<tr>
<td>start_minute (max_dom)</td>
<td>00</td>
<td>2 digit minute of starting time</td>
</tr>
<tr>
<td>start_second (max_dom)</td>
<td>00</td>
<td>2 digit second of starting time *note: the start time is used to name the first wrfout file. It also controls the start time for nest domains, and the time to restart</td>
</tr>
<tr>
<td>end_year (max_dom)</td>
<td>2012</td>
<td>4 digit year of ending time</td>
</tr>
<tr>
<td>end_month (max_dom)</td>
<td>06</td>
<td>2 digit month of ending time</td>
</tr>
<tr>
<td>end_day (max_dom)</td>
<td>12</td>
<td>2 digit day of ending time</td>
</tr>
<tr>
<td>end_hour (max_dom)</td>
<td>12</td>
<td>2 digit hour of ending time</td>
</tr>
<tr>
<td>end_minute (max_dom)</td>
<td>00</td>
<td>2 digit minute of ending time</td>
</tr>
<tr>
<td>end_second (max_dom)</td>
<td>00</td>
<td>2 digit second of ending time *note: all end times also control when the nest domain integrations end. All start and end times are used by real.exe. You may use either run_days/run_hours/etc. or end_year/month/day/hour/etc. to control the length of model</td>
</tr>
</tbody>
</table>
The program `real.exe` uses start and end times only for the program `real.exe`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>interval_seconds</code></td>
<td>10800</td>
<td>Time interval between the incoming real data, which will be the interval between the lateral boundary condition file (for <code>real</code> only)</td>
</tr>
<tr>
<td><code>input_from_file</code></td>
<td>.true.</td>
<td>Logical; whether the nested run will have input files for domains other than domain 1</td>
</tr>
<tr>
<td><code>fine_input_stream</code></td>
<td></td>
<td>Selected fields from nest input</td>
</tr>
<tr>
<td><code>history_interval</code></td>
<td>60</td>
<td>History output file interval in minutes (integer only)</td>
</tr>
<tr>
<td><code>history_interval_d</code></td>
<td>1</td>
<td>History output file interval in days (integer only); used as an alternative to <code>history_interval</code></td>
</tr>
<tr>
<td><code>history_interval_h</code></td>
<td>1</td>
<td>History output file interval in hours (integer only); used as an alternative to <code>history_interval</code></td>
</tr>
<tr>
<td><code>history_interval_m</code></td>
<td>1</td>
<td>History output file interval in minutes (integer only); used as an alternative to <code>history_interval</code> and is equivalent to <code>history_interval</code></td>
</tr>
<tr>
<td><code>history_interval_s</code></td>
<td>1</td>
<td>History output file interval in seconds (integer only); used as an alternative to <code>history_interval</code></td>
</tr>
<tr>
<td><code>frames_per_outfile</code></td>
<td>1</td>
<td>Number of output times bulked into each history file; used to split output files into smaller pieces</td>
</tr>
<tr>
<td><code>restart</code></td>
<td>.false.</td>
<td>Logical; whether this run is a restart</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>restart_interval</td>
<td>1440</td>
<td>restart output file interval in minutes</td>
</tr>
<tr>
<td>override_restart_timers</td>
<td>.false.</td>
<td>(default) uses all output intervals (including history) given by the wrfrst files</td>
</tr>
<tr>
<td></td>
<td>.true.</td>
<td>uses restart output intervals given by the namelist</td>
</tr>
<tr>
<td>write_hist_at_0h_rst</td>
<td>.false.</td>
<td>(default) does not give a history file at the initial time of restart (prevents overwriting original history file at this time)</td>
</tr>
<tr>
<td></td>
<td>.true.</td>
<td>gives a history file at the initial time of restart</td>
</tr>
<tr>
<td>output_ready_flag</td>
<td>.true.</td>
<td>asks the model to write-out an empty file with the name 'wrfoutReady_d&lt;domain&gt;_&lt;date&gt;'; Useful in production runs so that post-processing code can check on the completeness of this file.</td>
</tr>
<tr>
<td>reset_simulation_start</td>
<td>.false.</td>
<td>whether to overwrite the simulation start date with the forecast start time</td>
</tr>
<tr>
<td>auxinput1_inname</td>
<td>&quot;met_em.d&lt;domain&gt;_&lt;date&gt;&quot;</td>
<td>(default); name of input file from WPS</td>
</tr>
<tr>
<td>auxinput4_inname</td>
<td>&quot;wrflowinp_d&lt;domain&gt;&quot;</td>
<td>name of input file for lower boundary file; works with sst_update = 1</td>
</tr>
<tr>
<td>auxinput4_interval</td>
<td>360</td>
<td>file interval in minutes for lower boundary file; works with sst_update = 1</td>
</tr>
<tr>
<td>io_form_auxinput4</td>
<td>2</td>
<td>IO format for wrflowinp files; required for V3.2; works with sst_update = 1</td>
</tr>
<tr>
<td>io_form_history</td>
<td>2</td>
<td>the format in which the history output file will be</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>split netCDF files, one per processor *note: no supported post-processing software for split files</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>binary format *note: no supported post-processing software available</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>PHDF5 format *note: no supported post-processing software available</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>GRIB1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>10</td>
<td>GRIB2</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>parallel netCDF</td>
<td></td>
</tr>
<tr>
<td>io_form_restart</td>
<td>the format in which the restart output files will be</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>nedCDF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>split netCDF files, one per processor (must restart with the same number of processors)</td>
<td></td>
</tr>
<tr>
<td>io_form_input</td>
<td>the format of the input files</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>netCDF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>allows the program <em>real.exe</em> to read in split <em>met_em</em> files, and write split <em>wrfinput</em> files. No split file for the <em>wrfbdy</em> file.</td>
<td></td>
</tr>
<tr>
<td>io_form_boundary</td>
<td>the format for the <em>wrfbdy</em> file</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>netCDF format</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PHD5 format</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GRIB1 format</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>GRIB2 format</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>pnetCDF format</td>
<td></td>
</tr>
<tr>
<td>ncd_nofill</td>
<td>.true.</td>
<td>(default) only a single write, not the write/read/write sequence (new in V3.6)</td>
</tr>
<tr>
<td>io_form_auxinput2</td>
<td>IO format for input stream 2 data</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>netCDF format</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PHD5 format</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GRIB1 format</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>GRIB2 format</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>pnetCDF format</td>
<td></td>
</tr>
<tr>
<td>diag_print</td>
<td>0</td>
<td>(default) When set to 1 or 2, it allows some simple diagnostic fields to be output</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>domain-averaged 3-hourly hydrostatic surface pressure tendency (Dpsfc/Dt), and dry-hydrostatic column pressure tendency (Dmu/Dt) will appear in stdout file.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>in addition to those listed above, domain-averaged rainfall, surface evaporation, and sensible and latent heat fluxes will be output in stdout file.</td>
</tr>
<tr>
<td>debug_level</td>
<td>0</td>
<td>giving this a larger value (50, 100, 200, etc.) increases the debugging</td>
</tr>
<tr>
<td>Variable</td>
<td>Value/Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>auxhist2_outname</td>
<td>&quot;rainfall_d&lt;domain&gt;&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>print-outs when running WRF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>file name to write additional output to a different unit or output stream. If not specified, auxhist2_d&lt;domain&gt;_&lt;date&gt; will be used. Also note that to write variables in output other than the history file requires either a change in the Registry.EM_COMMON file, or the use of the option iofields_filename option.</td>
<td></td>
</tr>
<tr>
<td>auxhist2_interval</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>(max_dom)</td>
<td>the interval in minutes for the output</td>
<td></td>
</tr>
<tr>
<td>io_form_auxhist2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>output format for using auxhist2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 netCDF format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 PHD5 format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 GRIB1 format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 GRIB2 format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11 pnetCDF format</td>
<td></td>
</tr>
<tr>
<td>frames_per_auxhist2</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>(max_dom)</td>
<td>how many output times will be in each output file</td>
<td></td>
</tr>
<tr>
<td>auxinput11_interval</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>interval in minutes for obs nudging input. It should be set as the same (or more) frequency as obs_ionf (with the unit of the coarse domain time step)</td>
<td></td>
</tr>
<tr>
<td>auxinput11_end_h</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>end of the observation time (in hours), when using the diag_print option</td>
<td></td>
</tr>
<tr>
<td>nocolons</td>
<td>.false.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>when set to .true. this replaces the colons with underscores in the output file names</td>
<td></td>
</tr>
<tr>
<td>write_input</td>
<td>.true.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>write input-formatted data as output for 3DVAR application</td>
<td></td>
</tr>
<tr>
<td>inputout_interval</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>interval in minutes when using the write_input option</td>
<td></td>
</tr>
<tr>
<td>input_outname</td>
<td>&quot;wrf_3dvar_input_d&lt;domain&gt;_&lt;date&gt;&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output file name from 3DVAR</td>
<td></td>
</tr>
<tr>
<td>inputout_begin_y</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>beginning year to write 3DVAR data</td>
<td></td>
</tr>
<tr>
<td>inputout_begin_d</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>beginning day to write 3DVAR data</td>
<td></td>
</tr>
<tr>
<td>inputout_begin_h</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>beginning hour to write 3DVAR data</td>
<td></td>
</tr>
<tr>
<td>inputout_begin_m</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>beginning minute to write 3DVAR data</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>inputout_begin_s</td>
<td>0</td>
<td>beginning second to write 3DVAR data</td>
</tr>
<tr>
<td>inputout_end_y</td>
<td>0</td>
<td>ending year to write 3DVAR data</td>
</tr>
<tr>
<td>inputout_end_d</td>
<td>0</td>
<td>ending day to write 3DVAR data</td>
</tr>
<tr>
<td>inputout_end_h</td>
<td>12</td>
<td>ending hour to write 3DVAR data</td>
</tr>
<tr>
<td>inputout_end_m</td>
<td>0</td>
<td>ending minute to write 3DVAR data</td>
</tr>
<tr>
<td>inputout_end_s</td>
<td>0</td>
<td>ending second to write 3DVAR data</td>
</tr>
</tbody>
</table>

*NOTE: The above example shows that the input-formatted data are output starting from hour 3 to hour 12 in a 180-min interval.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_ic_times</td>
<td>.false.</td>
<td>when set to .true., allows you to output a wrfinput file for all time periods</td>
</tr>
<tr>
<td>adjust_output_times</td>
<td>.false.</td>
<td>(default); adjust output times to the nearest hour</td>
</tr>
<tr>
<td>output_ready_flag (new since V3.6.1)</td>
<td>.true.</td>
<td>(default = .false.); when turned on, the model will write out an empty file with the name wrfoutReady_d&lt;domain&gt;_&lt;date&gt;. This is useful in production runs so that post-processing code can check on the existence of this file to start doing processing.</td>
</tr>
<tr>
<td>output_diagnostics (new since V3.3.1)</td>
<td>0</td>
<td>turned off 36 surface diagnostic arrays (max/min/mean/std) in the time interval are specified. The output goes to auxiliary history output stream 3 with default file name 'wrfxtrm_d&lt;domain&gt;_&lt;date&gt;'. You must also set io_form_auxhist3 = 2, auxhist3_interval = 1440, 1440, and frames_per_auxhist3 = 1000, 1000. <strong>Note:</strong> do restart only at multiple of auxhist3 intervals</td>
</tr>
<tr>
<td>nwp_diagnostics (new)</td>
<td>0</td>
<td>turned off</td>
</tr>
</tbody>
</table>
For automatic moving nests

**input_from_hires**

*(max_dom) (new since V3.6)*

| .false. | When set to .true., high-resolution terrain and landuse will be used in the nests (requires special input data, and environment variable TERRAIN_AND_LANDUSE set at compile time). This optin will overwrite input_from_file option for nest domains. |

**rsmas_data_path**

| “high-res-data-directory” | Directory path where the high-res data is stored. |

**iofields_filename**

*(max_dom)*

| "my_iofields_list.txt" | an option to request particular variables to appear in output, if they are not already, or to not appear if they do and you do not want them to. You must also create a text file (my_iofields_list.txt) in which you will declare the variables to be output. It will be a single line of text, e.g.: +:h:7:RAINC,RAINNC or -:h:0:RAINC,RAINNC |

**ignore_iofields_warning**

| .true. | tells the model to continue if an error is encountered in the user-specified files |

| .false. | tells the model to abort if an error is encountered in the user-specified files |

**&domains**

| **dimensions, nesting, parameters** |

| **time_step** | 60 | time step for integration seconds (recommended 6*dx in km for a typical case) |

| **time_step_fract_num** | 0 | numerator for fractional time step |

<p>| <strong>time_step_fract_den</strong> | 1 | denominator for fractional time step. E.g., if you want to use 60.3 sec as your time step, set time_step = 60, time_step_fract_num = 3, |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_step_dfi</td>
<td>60</td>
<td>time step when setting dfi_opt = 1, may be different from the regular time step</td>
</tr>
<tr>
<td>max_dom</td>
<td>1</td>
<td>the number of domains over which you are running</td>
</tr>
<tr>
<td>s_we (max_dom)</td>
<td>1</td>
<td>start index in x (west-east) direction (leave as is)</td>
</tr>
<tr>
<td>e_we (max_dom)</td>
<td>91</td>
<td>end index in x (west-east) direction (staggered dimension)</td>
</tr>
<tr>
<td>s_sn (max_dom)</td>
<td>1</td>
<td>start index in y (south-north) direction (leave as is)</td>
</tr>
<tr>
<td>e_sn (max_dom)</td>
<td>82</td>
<td>end index in y (south-north) direction (staggered dimension)</td>
</tr>
<tr>
<td>s_vert (max_dom)</td>
<td>1</td>
<td>start index in z (vertical) direction (leave as is)</td>
</tr>
<tr>
<td>e_vert (max_dom)</td>
<td>30</td>
<td>end index in z (vertical) direction (staggered dimension -- this refers to full levels). Most variables are on unstaggered levels. *Note: Vertical dimensions need to be the same for all nests</td>
</tr>
<tr>
<td>dx (max_dom)</td>
<td>30000</td>
<td>grid length in x-direction (in meters)</td>
</tr>
<tr>
<td>dy (max_dom)</td>
<td>30000</td>
<td>grid length in y-direction (in meters)</td>
</tr>
<tr>
<td>ztop (max_dom)</td>
<td>19000</td>
<td>height in meters; used to define model top for idealized cases</td>
</tr>
<tr>
<td>grid_id (max_dom)</td>
<td>1</td>
<td>domain identifier</td>
</tr>
<tr>
<td>parent_id (max_dom)</td>
<td>0</td>
<td>ID of the parent domain</td>
</tr>
<tr>
<td>i_parent_start (max_dom)</td>
<td>1</td>
<td>the starting lower-left corner i-index from the parent domain</td>
</tr>
<tr>
<td>j_parent_start (max_dom)</td>
<td>1</td>
<td>the starting lower-left corner j-index from the parent domain</td>
</tr>
<tr>
<td>parent_grid_ratio (max_dom)</td>
<td>1</td>
<td>parent-to-nest domain grid size ratio. *Note: for real data cases the ratio must be odd; for ideal data cases, the ratio can be even if feedback is set to 0.</td>
</tr>
<tr>
<td>parent_time_step_ratio (max_dom)</td>
<td>1</td>
<td>parent-to-nest time step ratio; this can be different from the parent_grid_ratio</td>
</tr>
<tr>
<td>feedback</td>
<td>0</td>
<td>no feedback</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>feedback from nest to its parent domain</td>
</tr>
<tr>
<td>smooth_option</td>
<td>0</td>
<td>no smoothing</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1-2-1 smoothing option for parent domain; used only with feedback=1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>(default) smoothing-desmoothing option for parent domain; used only with feedback=1</td>
</tr>
<tr>
<td>hypsometric_opt</td>
<td>2</td>
<td>(default) computes height in program real.exe and pressure in the model (ARW only) by using an alternative method (less biased when compared against input data)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>original method</td>
</tr>
<tr>
<td>max_ts_locs</td>
<td>5</td>
<td>maximum number of time series locations</td>
</tr>
<tr>
<td>max_ts_level</td>
<td>15</td>
<td>highest model level for profile output</td>
</tr>
</tbody>
</table>

**Options for Program real.exe**

<table>
<thead>
<tr>
<th>num_metgrid_levels</th>
<th>40</th>
<th>number of vertical levels in WPS output (type ncdump -h on one of the met_em* files to find out this number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>num_metgrid_soil_levels</td>
<td>4</td>
<td>number of soil levels or layers in WPS output (type ncdump -h on one of the met_em* files to find out this number)</td>
</tr>
<tr>
<td>eta_levels</td>
<td>1.0, 0.99, ...0.0</td>
<td>model eta levels from 1 to 0. If not given, real will provide a set of levels</td>
</tr>
<tr>
<td>ideal_init_method</td>
<td>1</td>
<td>method to compute albedo in idealized cases in start_em</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>(default) albedo from phb</td>
</tr>
<tr>
<td>interp_method_type</td>
<td></td>
<td>The default is to use the Smolarkiewicz &quot;SINT&quot; method; however, this is known to break with the implementation inside of WRF for large refinement ratios (such as 15:1). For those extreme and rare occurrences, other schemes are available. For options 1, 3, 4, and 12, the FG lateral</td>
</tr>
</tbody>
</table>
boundaries use the same horizontal scheme for the lateral BC computations

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bi-linear interpolation</td>
</tr>
<tr>
<td>2</td>
<td>(default) SINT</td>
</tr>
<tr>
<td>3</td>
<td>nearest-neighbor - only to be used for testing purposes</td>
</tr>
<tr>
<td>4</td>
<td>overlapping quadratic</td>
</tr>
<tr>
<td>12</td>
<td>for testing only, uses SINT horizontal interpolation, and same scheme for computation of FG lateral boundaries</td>
</tr>
</tbody>
</table>

**Vertical interpolation options**

<table>
<thead>
<tr>
<th>force_sfc_in_vinterp</th>
<th>1</th>
<th>(default) use the surface level as the lower boundary when interpolating through this many eta levels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>perform traditional trapping interpolation</td>
</tr>
</tbody>
</table>

| maxw_horiz_pres_diff (new since V3.6.1) | 5000 | (default) Pressure threshold (Pa). For using the level of max winds when the pressure difference between neighboring values exceeds this maximum, the variable is NOT inserted into the column for vertical interpolation. ARW real only. |

| trop_horiz_pres_diff (new since V3.6.1) | 5000 | (default) Pressure threshold (Pa). For using the tropopause level when the pressure difference between neighboring values exceeds this maximum, the variable is NOT inserted into the column for vertical interpolation. ARW real only. |

| maxw_above_this_level (new since V3.6.1) | 30000 | (default) minimum height (it is actually pressure in Pa) to allow using the level of max wind information in real. With a value of 300 hPa, then a max wind value at 500 hPa will be ignored. ARW real only. |

| use_maxw_level (new since V3.7.1) | 1 | use max wind speed level in vertical interpolation inside of the ARW real program (default = 0; do
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_trop_level (new since V3.7.1)</td>
<td>1</td>
<td>same as above, but with tropopause level data (default = 0; do not use)</td>
</tr>
<tr>
<td>interp_theta (new since V3.3.1)</td>
<td>.false.</td>
<td>(default) vertically interpolates temperature (which may reduce bias when compared with input data)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(default changed to .false. beginning V3.4)</td>
</tr>
<tr>
<td>p_top_requested</td>
<td>5000</td>
<td>pressure top (in Pa) to use in the model; must be available in WPS data</td>
</tr>
<tr>
<td>interp_type</td>
<td>2</td>
<td>(default) vertical interpolation that is linear in log(pressure)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>vertical interpolation that is linear in pressure</td>
</tr>
<tr>
<td>extrap_type</td>
<td>2</td>
<td>(default) vertical extrapolation of non-temperature variables, using the lowest level as constant below ground</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>vertical extrapolation of non-temperature variables, using the 2 lowest levels</td>
</tr>
<tr>
<td>t_extrap_type</td>
<td></td>
<td>vertical extrapolation for potential temp:</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>(default) -6.5 K/km lapse rate for temperature</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>isothermal</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>constant theta</td>
</tr>
<tr>
<td>use_levels_below_ground</td>
<td></td>
<td>in vertical interpolation, whether to use levels below input surface level</td>
</tr>
<tr>
<td></td>
<td>.true.</td>
<td>(default) use input isobaric levels below input surface</td>
</tr>
<tr>
<td></td>
<td>.false.</td>
<td>extrapolate when WRF location is below input surface level</td>
</tr>
<tr>
<td>use_surface</td>
<td>.true.</td>
<td>(default) uses input surface level data in vertical interpolation</td>
</tr>
<tr>
<td></td>
<td>.false.</td>
<td>do not use input surface data</td>
</tr>
<tr>
<td>lagrange_order</td>
<td>2</td>
<td>(default) quadratic vertical interpolation order</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(default changed to 2 beginning V3.4)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>linear vertical interpolation order</td>
</tr>
<tr>
<td>Option</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lowest_lev_from_sfc</td>
<td>.false.</td>
<td>(default) use traditional interpolation</td>
</tr>
<tr>
<td></td>
<td>.true.</td>
<td>use surface values for the lowest eta (u,v,t,q)</td>
</tr>
<tr>
<td>sfcp_to_sfcp</td>
<td>.true.</td>
<td>optional method to compute model's surface pressure when incoming data only has surface pressure and terrain, but not sea-level pressure (default is .false.)</td>
</tr>
</tbody>
</table>
| use_tavg_for_tsk                   | .true. | uses diurnally-averaged surface temp as skin temp. The diurnally-
<p>|                                    |       | averaged surface temp can be computed using WPS utility avg_tsfc.exe. May use this option when SKINTEMP is not present (default is .false.) |
| rh2qv_wrt_liquid (new since V3.3)  | .true. | (default) computes qv with respect to liquid water                         |
|                                    | .false. | computes qv with respect to ice                                            |
| rh2qv_method (new since V3.3)       | 1     | (default) old MM5 method                                                    |
|                                    | 2     | uses a WMO recommended method (WMO-No. 49, corrigendum, August 2000)        |
| smooth_cg_topo                     | .true. | smooths the outer rows and columns of the domain 1 topography with respect to the input data (default is .false.) |
| vert_refine_fact                   | 1     | vertical refinement factor for ndown (1 = same number of vertical levels as the coarse domain, 2 = double the vertical resolution, and so on); not used for current vertical grid refinement |
| vert_refine_method (max_dom) (new since V3.3) | 0     | (default) no vertical refinement                                           |
|                                    | 1     | integer vertical refinement                                                |
|                                    | 2     | use specified or computed eta levels for vertical refinement               |
| Options for Preset Moving Nest     |       |                                                                             |
| num_moves                          | 2     | total # of moves for all domains                                           |
| move_id (max_moves)                | 2, 2, | a list of nest domain ID's, one per                                         |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>move</strong> interval (max_moves)</td>
<td>60, 120</td>
<td>time in minutes since the start of this domain</td>
</tr>
<tr>
<td><strong>move_cd_x</strong> (max_moves)</td>
<td>1, -1,</td>
<td>the # of parent domain grid cells to move in the i-direction</td>
</tr>
<tr>
<td><strong>move_cd_y</strong> (max_moves)</td>
<td>-1, 1,</td>
<td>the # of parent domain grid cells to move in the j-direction (positive in increasing i/j directions, and negative in decreasing i/j directions. Only 1, 0, and -1 is permitted.</td>
</tr>
<tr>
<td><strong>Options for Automatic Moving Nest</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vortex_interval (max_dom)</td>
<td>15</td>
<td>how often the new vortex position is computed (in mins)</td>
</tr>
<tr>
<td>max_vortex_speed (max_dom)</td>
<td>40</td>
<td>used to compute the search radius for the new vortex position (in m/s)</td>
</tr>
<tr>
<td>corral_dist (max_dom)</td>
<td>8</td>
<td>how close the moving nest is allowed to get to the coarse grid boundary. This # sets the minimum limit of grid cells allowed between them.</td>
</tr>
<tr>
<td>track_level</td>
<td>50000</td>
<td>pressure level value (Pa) at which the tropical storm vortex is tracked</td>
</tr>
<tr>
<td>time_to_move (max_dom)</td>
<td>0</td>
<td>time (in mins) to start moving nest</td>
</tr>
<tr>
<td><strong>Options for Adaptive Time Step</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use_adaptive_time_step</td>
<td>.true.</td>
<td>use adaptive time step (default is .false.)</td>
</tr>
<tr>
<td>step_to_output_time</td>
<td>.true.</td>
<td>modifies the time step so that the exact history time is reached</td>
</tr>
<tr>
<td>target_cfl (max_dom)</td>
<td>1.2, 1.2, 1.2,</td>
<td>if vertical CFL $\leq$ this value, then time step is increased</td>
</tr>
<tr>
<td>target_hcfl (max_dom) (new since V3.3)</td>
<td>0.84, 0.84, 0.84,</td>
<td>if horizontal CFL $\leq$ this value, the time step is increased</td>
</tr>
<tr>
<td>max_step_increase_pct (max_dom)</td>
<td>5, 51, 51</td>
<td>percentage of previous time step to increase if the max CFL is $\leq$ target_cfl</td>
</tr>
<tr>
<td>starting_time_step (max_dom)</td>
<td>-1, -1, -1</td>
<td>flag -1 implies 4*dx is used to start the model. Any positive integer specifies the time step the model will use to start (in seconds).</td>
</tr>
</tbody>
</table>

*Note: when use_adaptive_time_step = .true., the value specified for
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>starting_time_step_den (max_dom) * (new since V3.6)</td>
<td>0</td>
<td>denominator for starting_time_step (so that fractional time step can be used)</td>
</tr>
<tr>
<td>max_time_step (max_dom)</td>
<td>-1, -1, -1</td>
<td>flag -1 implies the maximum time step is $8 \times dx$. Any positive integer specifies the maximum time step (in seconds).</td>
</tr>
<tr>
<td>min_time_step (max_dom)</td>
<td>-1, -1, -1</td>
<td>flag -1 implies the minimum time step is $3 \times dx$. Any positive integer specifies the minimum time step (in seconds).</td>
</tr>
<tr>
<td>min_time_step_den (max_dom) * (new since V3.6)</td>
<td>0</td>
<td>denominator for min_time_step</td>
</tr>
<tr>
<td>adaptation_domain</td>
<td>1</td>
<td>(default) specifies which domain to use to drive adaptive time stepping</td>
</tr>
<tr>
<td><strong>Options to Control Parallel Computing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tile_sz_x</td>
<td>0</td>
<td>number of points in tile x direction (open MP only)</td>
</tr>
<tr>
<td>tile_sz_y</td>
<td>0</td>
<td>number of points in tile y direction; can be determined automatically (open MP only)</td>
</tr>
<tr>
<td>numtiles</td>
<td>1</td>
<td>number of tiles per patch (alternative to above 2 items; open MP only)</td>
</tr>
<tr>
<td>nproc_x</td>
<td>-1</td>
<td>(default) turned off; code will do automatic decomposition (MPI only)</td>
</tr>
<tr>
<td></td>
<td>&gt;1</td>
<td>number of processors in x for decomposition (MPI only)</td>
</tr>
<tr>
<td>nproc_y</td>
<td>-1</td>
<td>(default) turned off; code will do automatic decomposition (MPI only)</td>
</tr>
<tr>
<td></td>
<td>&gt;1</td>
<td>number of processors in y for decomposition (MPI only)</td>
</tr>
<tr>
<td><strong>Options for 3D Ocean Model</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ocean_levels</td>
<td>30</td>
<td>(default) number of ocean levels when using $\text{sf_ocean_physics} = 2$</td>
</tr>
<tr>
<td>ocean_z</td>
<td></td>
<td>(values for # of vertical profile of layer depth (m))</td>
</tr>
<tr>
<td><strong>ocean_levels</strong></td>
<td>for number of ocean_levels. See /run/README.namelist for more details.</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>ocean_t</strong></td>
<td>(values for # of ocean_levels) vertical profile of ocean temps (K) for number of ocean_levels.</td>
<td></td>
</tr>
<tr>
<td><strong>ocean_s</strong></td>
<td>(values for # of ocean_levels) vertical profile of salinity.</td>
<td></td>
</tr>
</tbody>
</table>

**&physics**

<table>
<thead>
<tr>
<th>mp_physics (max_dom)</th>
<th>0</th>
<th>(default) no microphysics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Kessler scheme</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Lin et al. scheme</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>WSM 3-class simple ice scheme</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>WSM 5-class scheme</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Ferrier (new Eta) microphysics, operational High-Resolution Window</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>WSM 6-class graupel scheme</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Goddard GCE scheme (also uses gsfcgce_hail and gsfcgce_2ice)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Thompson graupel scheme (2-moment scheme in V3.1)</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Milbrandt-Yau 2-moment scheme</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Morrison 2-moment scheme</td>
</tr>
<tr>
<td><strong>(new since V3.5)</strong></td>
<td>11</td>
<td>CAM 5.1 5-class scheme</td>
</tr>
<tr>
<td><strong>(new since V3.3)</strong></td>
<td>13</td>
<td>SBU_YLin, 5-class scheme</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>WRF double moment, 5-class scheme</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>High-resolution Ferrier microphysics, with advection</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>WRF double moment, 6-class scheme</td>
</tr>
<tr>
<td><strong>(new since V3.4)</strong></td>
<td>17</td>
<td>NSSL 2-moment 4-ice scheme (steady background CCN)</td>
</tr>
<tr>
<td><strong>(new since V3.4)</strong></td>
<td>18</td>
<td>NSSL 2-moment 4-ice scheme with predicted CCN (better for idealized than real cases); to set a global CCN value, use nssl_cccn = 0.7e9 (CCN for NSSL scheme 18). Also sets same value to ccn_conc for mp_physics = 18.</td>
</tr>
<tr>
<td><strong>(new since V3.5)</strong></td>
<td>19</td>
<td>NSSL 1-moment, 6-class scheme</td>
</tr>
<tr>
<td><strong>(new since V3.5)</strong></td>
<td>21</td>
<td>NSSL-LFO 1-moment, 6-class;</td>
</tr>
</tbody>
</table>
very similar to Gilmore et al. 2004; can set intercepts and particle densities in physics namelist, e.g., nssl_cnor for NSSL 1-moment schemes, intercept and particle densities can be set for snow, graupel, hail, and rain. For the 1- and 2-moment schemes, the shape parameters for graupel and hail can be set. See /WRFV3/run/README.namelist file for specifics.

(new since V3.7) 22 NSSL 2-moment 3-ice scheme, no hail.

(new since V3.6) 28 aerosol-aware Thompson scheme with water- and ice-friendly aerosol climatology (new for V3.6); this option has 2 climatological aerosol input options: use_aero_icbs = .F. (use constant values), and use_aero_icbc = .T. (use input from WPS)

(new since V3.6) 30 HUJI (Hebrew University of Jerusalem, Israel) spectral bin microphysics, fast version

(new since V3.6) 32 HUJI spectral bin microphysics, full version

95 Ferrier (old Eta), operational NAM (WRF NMM)

98 Thompson scheme in V3.0

do_radar_ref (new since V3.4.1) 0 allows radar reflectivity to be computed using mp-scheme-specific parameters. Currently works for mp_physics = 2,4,6,7,8,10,14,16 0: off 1: on

mp_zero_out for non-zero mp_physics options, this keeps moisture variables above a threshold value ≥0. An alternative (and better) way to keep moisture variables positive is to use the moist_adv_opt.

0 (default) no action taken; no adjustment to any moisture field
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>except for Qv, all other moisture arrays are set to zero if they fall below a critical value</td>
<td>1</td>
<td>Qv ≥ 0 and all other moisture arrays are set to zero if they fall below a critical value</td>
</tr>
<tr>
<td>mp_zero_out_thresh</td>
<td>1.e-8</td>
<td>critical value for moisture variable threshold, below which moisture arrays (except for Qv) are set to zero (unit: kg/kg)</td>
</tr>
<tr>
<td>mp_tend_lim</td>
<td>10.</td>
<td>limit on temp tendency from microphysics latent heating when radar data assimilation is used</td>
</tr>
<tr>
<td>gsf cgce hail</td>
<td>0</td>
<td>(default) running gsf cgce scheme with graupel</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>running gsf cgce scheme with hail</td>
</tr>
<tr>
<td>gsf cgce 2 ice</td>
<td>0</td>
<td>(default) running gsf cgce scheme with snow, ice, and graupel/hail</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>running gsf cgce scheme with only ice and snow (gsfcgce_hail is ignored)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>running gsf cgce scheme with only ice and graupel (used only in very extreme situation; gsf cgce_hail is ignored)</td>
</tr>
<tr>
<td>ccn_conc (new name since V3.7)</td>
<td>1.0E8</td>
<td>(default) CCN concentration; used by WDM schemes (previously afwa_ccn_conc, new in V3.6.1)</td>
</tr>
<tr>
<td>hail_opt (new name since V3.7)</td>
<td>1</td>
<td>hail/graupel switch for WSM6, WDM6, and Morrison schemes (previously afwa_hail_opt, new in V3.6.1)</td>
</tr>
</tbody>
</table>

The following 9 namelists are for the NSSL 1-moment scheme. For the 1- and 2-moment schemes, the shape parameters for graupel and hail can also be set.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nssl alpha h</td>
<td>0</td>
<td>shape parameter for graupel</td>
</tr>
<tr>
<td>nssl alpha hl</td>
<td>2</td>
<td>shape parameter for hail</td>
</tr>
<tr>
<td>nssl cnoh</td>
<td>4.e5</td>
<td>graupel intercept</td>
</tr>
<tr>
<td>nssl cnohl</td>
<td>4.e4</td>
<td>hail intercept</td>
</tr>
<tr>
<td>nssl cnor</td>
<td>8.e5</td>
<td>rain intercept</td>
</tr>
<tr>
<td>nssl cnos</td>
<td>3.e6</td>
<td>snow intercept</td>
</tr>
<tr>
<td>nssl rho qh</td>
<td>500.</td>
<td>graupel density</td>
</tr>
<tr>
<td>nssl rho ghl</td>
<td>900.</td>
<td>hail density</td>
</tr>
<tr>
<td>nssl rho qs</td>
<td>100.</td>
<td>snow density</td>
</tr>
</tbody>
</table>
| no_mp_heating | 1   | turn off latent heating from a microphysics scheme (0 is off and
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_mp_re</td>
<td>0</td>
<td>whether to use effective radii computed in mp schemes in RRTMG (default)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>use effective radii</td>
</tr>
<tr>
<td>ra_lw_physics</td>
<td>0</td>
<td>no longwave radiation</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>use effective radii</td>
</tr>
<tr>
<td>ra_sw_physics</td>
<td>0</td>
<td>no shortwave radiation</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Dudhia scheme (ptop &gt; 50 mb)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>(old) Goddard shortwave scheme</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>CAM scheme (restart must be at 6-hourly interval; also requires levsiz, paerlev, cam_abs_dim1(2); see below</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>rrtmg scheme (Default values for GHG in V3.5: co2vmr=379.e-6, n2ovmr=319.e-9, ch4vmr=1774.e-9; Values used in previous versions: co2vmr=330.e-6, n2ovmr=0., ch4vmr=0.)</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>fast rrtmg scheme for GPU and MIC</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Goddard scheme</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>FLG (UCLA) scheme</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>Earth Held-Suarez forcing</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>GFDL (Eta) longwave (semi-supported); also must use co2tf = 1 for ARW</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) no shortwave radiation</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Dudhia scheme (ptop &gt; 50 mb)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>(old) Goddard shortwave scheme</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>CAM scheme (restart must be at 6-hourly interval); must set levsiz, paerlev, cam_abs_dim1/2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>rrtmg scheme</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>fast rrtmg scheme for GPU and MIC</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Goddard scheme</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>FLG (UCLA) scheme</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>GFDL (Eta) longwave (semi-supported); also must use co2tf = 1 for ARW</td>
</tr>
</tbody>
</table>
supported); must use `co2tf = 1` for ARW

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>radt (max_dom)</td>
<td>30</td>
<td>minutes between radiation physics calls. Recommended 1 minute per km of dx (e.g. 10 for 10 km grid); use the same value for all nests</td>
</tr>
<tr>
<td>swint_opt (new since V3.5.1)</td>
<td></td>
<td>Interpolation of shortwave radiation based on the updated solar zenith angle between radiation calls</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>no interpolation</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>use interpolation</td>
</tr>
<tr>
<td>ra_call_offset</td>
<td>0</td>
<td>default; call radiation after output time</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>may call radiation just before output time.</td>
</tr>
<tr>
<td>co2tf</td>
<td>1</td>
<td>CO2 transmission function flag for GFDL radiation only. Set it to 1 for ARW, which allows generation of CO2 function internally</td>
</tr>
</tbody>
</table>

*Note: The following 5 variables for CAM are automatically set since V3.2*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cam_abs_freq_s</td>
<td>21600</td>
<td>default CAM clear sky longwave absorption calculation frequency (recommended minimum value to speed scheme up)</td>
</tr>
<tr>
<td>levsiz</td>
<td>59</td>
<td>(default) number of ozone data levels for CAM radiation</td>
</tr>
<tr>
<td>paerlev</td>
<td>29</td>
<td>(default) number of aerosol data levels for CAM radiation</td>
</tr>
<tr>
<td>cam_abs_dim1</td>
<td>4</td>
<td>(default) dimension for absnxt (absorption save array) in CAM radiation</td>
</tr>
<tr>
<td>cam_abs_dim2</td>
<td>same as e_vert</td>
<td>(default) dimension for abstot (2nd absorption save array) in CAM radiation</td>
</tr>
<tr>
<td>o3input (new since V3.5)</td>
<td></td>
<td>ozone input option (RRTMG only)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>using_profile inside the scheme</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>using CAM ozone data (ozone.formatted)</td>
</tr>
<tr>
<td></td>
<td>(became default in V3.7)</td>
<td></td>
</tr>
<tr>
<td>aer_opt</td>
<td></td>
<td>aerosol input option (RRTMG only)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>off</td>
</tr>
<tr>
<td></td>
<td>(new since V3.5)</td>
<td>using Tegen climatology</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>using J. A. Ruiz-Arias method (see other aer* options)</td>
</tr>
</tbody>
</table>
(new since V3.8) 3 using G. Thompson's water/ice-friendly climatological aerosol

alevsiz 12 no of vertical levels in aerosol data. Value set automatically.

no_src_types 6 no of aerosol types: organic and black carbon, sea salt, sulfate, dust and stratospheric aerosol (volcanic ash – currently 0). Value set automatically.

0 do not interpolate (default)
1 interpolate

*The following aerosol options allow RRTMG and new Goddard radiation schemes to see it, but the aerosols are constant during the model integration*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aer_aod550_opt</td>
<td>1</td>
<td>(default) input constant value for AOD at 550 nm from namelist; in this case, the value is read from aer_aod550_val</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>input value from auxiliary input 5; it is a time-varying 2D grid in netcdf wrf-compatible format.</td>
</tr>
<tr>
<td>aer_aod550_val</td>
<td>0.12</td>
<td>(default) value to be used with aer_aod550_opt = 1</td>
</tr>
<tr>
<td>aer_angexp_opt</td>
<td>1</td>
<td>(default) input constant value for Angstrom exponent from namelist. In this case, the value is read from aer_angexp_val</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>input value from auxiliary input 5, as in aer_aod550_opt</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Angstrom exponent value estimated from the aerosol type defined in aer_type, and modulated with the RH in WRF.</td>
</tr>
<tr>
<td>aer_angexp_val</td>
<td>1.3</td>
<td>(default) value to be used with aer_angexp_opt = 1</td>
</tr>
<tr>
<td>aer_ssa_opt</td>
<td>1</td>
<td>(default) input constant value for single scattering albedo from namelist. In this case, the value is read from aer_ssa_val</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>input value from auxiliary input 5, as in aer_aod550_opt</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>single scattering albedo value estimated from the aerosol type defined in aer_type, and modulated with the RH in WRF.</td>
</tr>
<tr>
<td>aer_ssa_val</td>
<td>0.85</td>
<td>(default) value to be used with aer_ssa_opt = 1</td>
</tr>
</tbody>
</table>
### aer_asy_opt
- **Value**: 1
- **Description**: (default) input constant value for asymmetry parameter from namelist. In this case, the value is read from aer_asy_val.

### aer_asy_val
- **Value**: 0.9
- **Description**: (default) value to be used with aer_asy_opt = 1.

### aer_type
- **Value**: 1 (default)
- **Description**: rural
- **Value**: 2
- **Description**: urban
- **Value**: 3
- **Description**: maritime

### sf_sfclay_physics
- **Value**: 0 (default)
- **Description**: no surface-layer
- **Value**: 1
- **Description**: Revised MM5 Monin-Obukhov scheme (Jimenez, renamed in v3.6)
- **Value**: 2
- **Description**: Monin-Obukhov (Janjic Eta) scheme
- **Value**: 3
- **Description**: NCEP GFS scheme (NMM only)
- **Value**: 4
- **Description**: QNSE
- **Value**: 5
- **Description**: MYNN
- **Value**: 7
- **Description**: Pleim-Xiu (ARW only), only tested with Pleim-Xiu surface and ACM2 PBL (new since V3.3)
- **Value**: 91
- **Description**: old MM5 surface layer scheme (previously option 1) (since V3.6; option 1 in earlier versions)

### iz0tlnlnd (new since V3.2)
- **Value**: 0 (default)
- **Description**: old, or non-vegetation dependent thermal roughness length over land
- **Value**: 1
- **Description**: veg dependent Chen-Zhang Czil

### sf_surface_physics
- **Value**: 0 (default)
- **Description**: no surface temp prediction
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>thermal diffusion scheme</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>unified Noah land-surface model</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RUC land-surface model</td>
<td></td>
</tr>
<tr>
<td>(new since V3.4)</td>
<td>4</td>
<td>Noah-MP land-surface model (additional options under the &amp;noah_mp section)</td>
</tr>
<tr>
<td>(new since V3.5)</td>
<td>5</td>
<td>CLM4 (Community Land Model Version 4)</td>
</tr>
<tr>
<td>(new since V3.4)</td>
<td>7</td>
<td>Pleim-Xiu scheme (ARW only)</td>
</tr>
<tr>
<td>sf_urban_physics</td>
<td>8</td>
<td>SSiB land-surface model (ARW only). Works with ra_lw_physics = 1, 3, or 4, and ra_sw_physics = 1, 3, or 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>activate urban canopy model (in Noah LSM only)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Single-layer, UCM</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Multi-layer, Building Environment Parameterization (BEP) scheme (works only with the MYJ and BouLac PBL)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Multi-layer, Building Environment Model (BEM) scheme (works only with MYJ and BouLac PBL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ua_phys (new since V3.5)</td>
</tr>
<tr>
<td></td>
<td>.false.</td>
<td>Option to activate UA Noah LSM changes to use a different snow-cover physics. Aimed toward improving treatment of snow as it relates to the vegetation canopy.</td>
</tr>
<tr>
<td>num_soil_layers</td>
<td></td>
<td>number of soil layers in land surface model (set before running real.exe)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>(default) thermal diffusion scheme for temp only</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Noah land-surface model</td>
</tr>
<tr>
<td></td>
<td>6 or 9</td>
<td>RUC land-surface model</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>CLM4 land-surface model</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Pleim-Xiu land-surface model</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>SSiB land-surface model</td>
</tr>
<tr>
<td>bl_pbl_physics (max_dom)</td>
<td></td>
<td>boundary layer option</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) no boundary-layer</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>YSU scheme; use sf_sfclay_physics =1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Mellor-Yamada-Janjic (Eta) TKE</td>
</tr>
<tr>
<td>Scheme</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NCEP GFS scheme (NMM only); use <code>sf_sfclay_physics=3</code></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>QNSE-EDMF; use <code>sf_sfclay_physics=4</code></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MYNN 2.5 level TKE; use <code>sf_sfclay_physics=1, 2, or 5</code></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MYNN 3rd level TKE; use <code>sf_sfclay_physics=5</code></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ACM2 (Pleim) scheme (ARW only); use <code>sf_sfclay_physics=1</code> or <code>7</code></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Bougeault and Lacarrere (BouLac) TKE; use <code>sf_sfclay_physics=1</code> or <code>2</code> (new since V3.3)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Bretherton-Park/UW TKE scheme; use <code>sf_sfclay_physics=1</code> or <code>2</code> (new since V3.3)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TEMF scheme (ARW only); use <code>sf_sfclay_physics=10</code> (new since V3.7)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Shin-Hong 'scale-aware' PBL scheme (new since V3.5)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GBM TKE-type scheme (ARW only); use <code>sf_sfclay_physics=1</code></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>MRF scheme (to be removed in the future)</td>
<td></td>
</tr>
</tbody>
</table>

**Variables**

- `mfshconv (max_dom)` turns on day-time EDMF for QNSE (0=off)
- `bldt (max_dom)` minutes between boundary-layer physics calls (0=call every time step)
- `topo_wind (max_dom)` (new since V3.4) turns on topographic surface wind correction, and requires extra input from geogrid. YSU PBL only
  - 0 off
  - 1 Jimenez method
  - 2 UW method
- `bl_mynn_tkebudget (max_dom)` (new since V3.4.1) (default) off
  - 1 adds MYNN tke budget terms to output
### bl_mynn_tkeadvect (max_dom) *(new since V3.5)*
- **false.** (default) off; does not advect tke in MYNN scheme (default)
- **true.** do MYNN tke advection

### icloud_bl *(new since V3.8)*
- option to couple the subgrid-scale clouds from the PBL scheme (MYNN only) to the radiation scheme
- **0** no coupling
- **1** (default) activate coupling to radiation

### bl_mynn_cloudmix *(new since V3.8)*
- option to activate mixing of qc and qi in MYNN (NOTE: qnc and qni are mixed when scalar_pblmix = 1)
- **0** no mixing of qc and qi
- **1** (default) mixing activated

### bl_mynn_mixlength *(new since V3.8)*
- option to change mixing length formulation in MYNN
- **0** original, as in Nakanishi and Niino 2009
- **1** (default) RAP/HRRR (including BouLac in free atmosphere)
- **2** experimental (includes cloud-specific mixing length and a scale-aware mixing length; following Ito et al. 2015, BLM); this option has been well-tested with the edmf options

### bl_mynn_cloudpdf
- option to switch to different cloud PDFs to represent subgrid clouds
- **0** original (Sommeria and Deardorf 1977)
- **1** Kuwano et al. 2010; similar to option 0, but uses resolved scale gradients, as opposed to higher order moments
- **2** (default) from Chaboureau and Bechtold 2002 (JAS, with mods)

### bl_mynn_edmf *(new since V3.8)*
- option to activate mass-flux scheme in MYNN
- **0** (default) regular MYNN
- **1** for StEM
- **2** for TEMF

### bl_mynn_edmf_mom *(new since V3.8)*
- option to activate momentum transport in MYNN mass-flux scheme

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheme (assuming bl_mynn_edmf &gt; 0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>no momentum transport</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>(default) momentum transport activated</td>
</tr>
<tr>
<td>bl_mynn_edmf_tke (new since V3.8)</td>
<td></td>
<td>option to activate TKE transport in MYNN mass-flux scheme</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>(default) no TKE transport</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>activate TKE transport</td>
</tr>
<tr>
<td>scalar_pblmix (new since V3.6)</td>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>mix scalar fields consistent with PBL option (exch_h)</td>
</tr>
<tr>
<td>tracer_pblmix (new since V3.6)</td>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>mix tracer fields consistent with PBL option (exch_h)</td>
</tr>
<tr>
<td>shinhong_tke_diag (max_dom) (new since V3.7)</td>
<td>0</td>
<td>diagnostic TKE and mixing length from Shin-Hong PBL</td>
</tr>
<tr>
<td>opt_thcnd (new since V3.8)</td>
<td>1</td>
<td>option to treat thermal conductivity in Noah LSM</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>McCumber and Pielke for silt loam and sandy loam</td>
</tr>
<tr>
<td>sf_surface_mosaic (new since V3.6)</td>
<td>0</td>
<td>option to mosaic landuse categories for Noah LSM</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>use mosaic landuse categories</td>
</tr>
<tr>
<td>mosaic_lu (new since V3.4)</td>
<td>1</td>
<td>option to specify landuse parameters based on a mosaic approach, when using the RUC land surface model; default is 0 (off)</td>
</tr>
<tr>
<td>mosaic_soil (new since V3.4)</td>
<td>1</td>
<td>option to specify soil parameters based on a mosaic approach, when using the RUC land surface model; default is 0 (off)</td>
</tr>
<tr>
<td>mosaic_cat (new since V3.6)</td>
<td>3</td>
<td>(default) number of mosaic landuse categories in a grid cell</td>
</tr>
<tr>
<td>grav_settling (max_dom) (new since V3.5.1)</td>
<td>0</td>
<td>gravitational settling of fog/cloud droplets (Now works for any PBL scheme, since V3.5.1)</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>(default) no settling of cloud</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>droplets</td>
<td>1</td>
<td>settling from Dyunkerke 1991 (in atmosphere at at surface)</td>
</tr>
<tr>
<td>Fogdes (vegetation and wind speed dependent; Katata et al. 2008) at surface, and Dyunkerke in the atmosphere</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ysu_topdown_pblmix (new since V3.7)</td>
<td>1</td>
<td>turns on top-down radiation-driven mixing (default is 0=no)</td>
</tr>
<tr>
<td>cu_physics (max _dom)</td>
<td>0</td>
<td>(default) no cumulus parameterization</td>
</tr>
<tr>
<td>1</td>
<td>Kain-Fritsch (new Eta) scheme</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Betts-Miller-Janjic scheme</td>
<td></td>
</tr>
<tr>
<td>(new since V3.5, replacing Grell-Devenyi scheme)</td>
<td>3</td>
<td>Grell-Freitas ensemble scheme</td>
</tr>
<tr>
<td>(new to ARW since V3.3)</td>
<td>4</td>
<td>Old GFS Simplified Arakawa-Schubert (SAS)</td>
</tr>
<tr>
<td>5</td>
<td>New Grell scheme (G3)</td>
<td></td>
</tr>
<tr>
<td>(new since V3.3)</td>
<td>6</td>
<td>Tiedtke scheme (ARW only)</td>
</tr>
<tr>
<td>(new since V3.3)</td>
<td>7</td>
<td>Zhang-McFarlane from CESM (works with MYJ and UW PBL)</td>
</tr>
<tr>
<td>(new since V3.7)</td>
<td>10</td>
<td>Modified Kain-Fritsch scheme with trigger function based on PDFs (ARW-only)</td>
</tr>
<tr>
<td>(new since V3.7)</td>
<td>11</td>
<td>Multi-scale Kain-Fritsch scheme</td>
</tr>
<tr>
<td>(new since V3.3)</td>
<td>14</td>
<td>New GFS SAS from YSU (ARW only)</td>
</tr>
<tr>
<td>(new since V3.7)</td>
<td>16</td>
<td>A newer Tiedtke scheme</td>
</tr>
<tr>
<td>84</td>
<td>New SAS (HWRF)</td>
<td></td>
</tr>
<tr>
<td>(option 3 before V3.5)</td>
<td>93</td>
<td>Grell-Devenyi ensemble scheme</td>
</tr>
<tr>
<td>99</td>
<td>previous Kain-Fritsch scheme</td>
<td></td>
</tr>
<tr>
<td>cudt</td>
<td>0</td>
<td>minutes between cumulus physics calls; should be set to 0 when using all cu_physics except Kain-Fritsch (0 = call every time step)</td>
</tr>
<tr>
<td>kfeta_trigger</td>
<td>1</td>
<td>The way to determines whether a grid point is convective; used only with cu_physics=1. = 1, default, original.</td>
</tr>
<tr>
<td>(new since V3.3)</td>
<td>2</td>
<td>moisture-advection based trigger (Ma and Tan 2009; ARW only)</td>
</tr>
<tr>
<td>3</td>
<td>relative humidity-dependent</td>
<td></td>
</tr>
<tr>
<td>ishallow</td>
<td>1</td>
<td>shallow convection used with</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cu_physics</td>
<td>3 or</td>
<td>independent shallow cumulus option (not tied to deep convection)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>(default is 0 = off)</td>
</tr>
<tr>
<td>shcu_physics (max_dom)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>no independent shallow cumulus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Park and Bretherton shallow cumulus from CAM5</td>
</tr>
<tr>
<td>(new since V3.3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>GRIMS scheme</td>
</tr>
<tr>
<td>(new since V3.5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Note:</em> The following 5 options show recommended #'s. If you would like to use any other number, consult the code to understand what you are doing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxiens</td>
<td>1</td>
<td>Grell-Devenyi and G3 only</td>
</tr>
<tr>
<td>maxens</td>
<td>3</td>
<td>Grell-Devenyi only</td>
</tr>
<tr>
<td>maxens2</td>
<td>3</td>
<td>Grell-Devenyi only</td>
</tr>
<tr>
<td>maxens3</td>
<td>16</td>
<td>Grell-Devenyi only</td>
</tr>
<tr>
<td>ensdim</td>
<td>144</td>
<td>Grell-Devenyi only</td>
</tr>
<tr>
<td>cugd_avedx</td>
<td>1</td>
<td>(default) number of grid boxes over which subsidence is spread, for large grid distances</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>for small grid distances (DX &lt; 5 km)</td>
</tr>
<tr>
<td>nsas_dx_factor (New since V3.6)</td>
<td>0</td>
<td>(default); off</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>nsas grid distance dependent option</td>
</tr>
<tr>
<td>For the KF-CuP Scheme:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shallowcu_forced_ra (max_dom) (new since V3.8)</td>
<td>.false.</td>
<td>radiative impact of shallow Cu by a prescribed maximum cloud fraction [cu_physics = 10 only; default = .false. (off); if =.true., radiative impact of shallow cu with a cloud fraction value of 0.36 ]</td>
</tr>
<tr>
<td>numbins (max_dom) (new since V3.8)</td>
<td>1</td>
<td>number of perturbations for potential temperature and mixing ratio in the CuP PDF (cu_physics = 10 only; default is 1; should be an odd number - recommended value is 21)</td>
</tr>
<tr>
<td>thBinSize (max_dom) (new since V3.8)</td>
<td>1</td>
<td>bin size of potential temperature perturbation increment: 0.01 K (cu_physics = 10 only; default is 1)</td>
</tr>
<tr>
<td>rBinSize (max_dom) (new since V3.8)</td>
<td>1</td>
<td>bin size of mixing ratio perturbation increment: 1.0e-4 kg/kg (cu_physics = 10 only; default is 1)</td>
</tr>
<tr>
<td>minDeepFreq (max_dom)</td>
<td>1</td>
<td>minimum frequency required</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>(new since V3.8)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>minShallowFreq</strong> (max_dom) <strong>(new since V3.8)</strong></td>
<td>1</td>
<td>minimum frequency required before shallow convection is allowed: 1.0e-2 (cu_physics = 10 only; default is 1)</td>
</tr>
<tr>
<td><strong>shcu_aerosols_opt</strong> (max_dom) <strong>(new since V3.8)</strong></td>
<td></td>
<td>whether to include aerosols in shcu (cu_physics = 10 only; run with WRF-Chem)</td>
</tr>
<tr>
<td><strong>cu_diag</strong> (max_dom)</td>
<td>0</td>
<td>(default) none</td>
</tr>
<tr>
<td><strong>cu_diag</strong> (max_dom)</td>
<td>2</td>
<td>prognostic</td>
</tr>
<tr>
<td><strong>kf_edrates</strong> (max_dom) <strong>(new since V3.8)</strong></td>
<td></td>
<td>option to add entrainment/detrainment rates and convective timescale output variables for KF-based cumulus schemes (cu_physics = 1, 11, and 99 only)</td>
</tr>
<tr>
<td><strong>convtrans_avglen_m</strong></td>
<td>30</td>
<td>averaging time for convective transport output variables (in minutes; only use with cu_physics=3, 5 and 93)</td>
</tr>
<tr>
<td><strong>cu_rad_feedback</strong> (max dom)</td>
<td>.true.</td>
<td>sub-grid cloud effect to the optical depth in radiation currently it works only for GF, G3, GD, and KF schemes; also need to set cu_diag = 1 for GF, G3, and GD schemes (default is .false. = off)</td>
</tr>
<tr>
<td><strong>isfflx</strong></td>
<td></td>
<td>heat and moisture fluxes from the surface for real-data cases and when a PBL is used (only works with sf_sfclay_physics=1, 5, 7, or 11)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = fluxes are on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = fluxes are off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It also controls surface fluxes when diff_opt = 2 and km_opt = 3, and a PBL isn’t used</td>
</tr>
</tbody>
</table>
### Studio Xland (new since V3.7)

<table>
<thead>
<tr>
<th></th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Constant fluxes defined by tke_drag_coefficient and tke_heat_flux</td>
</tr>
<tr>
<td>1</td>
<td>Use model-computed u* and heat and moisture fluxes</td>
</tr>
<tr>
<td>2</td>
<td>Use model-computed u* and specified heat flux by tke_heat_flux</td>
</tr>
</tbody>
</table>

### ifsnow

- **1**: (default) with snow-cover effect
- **0**: Without snow-cover effect

**Description**: Snow-cover effects (only works for sf_surface_physics=1)

### icloud

- **1**: (default) with cloud effect, and use cloud fraction option 1 (Xu-Randall method)
- **0**: Without cloud effect
- **2**: With cloud effect, and use cloud fraction option 2, 0/1 based on threshold

**Description**: Cloud effect to the optical depth in radiation (only works with ra_sw_physics=1,4 and ra_lw_physics=1,4); since V3.6 this also controls the cloud fraction options (new since V3.7)

### swrad_scat

- **1**: Scattering tuning parameter; default 1 is 1.e-5 m^2 kg^{-1} (only for ra_sw_physics=1). Increase for more scattering.

**Description**: Where landuse and soil category data come from (new since V3.7)

### surface_input_source

- **1**: (default prior to V3.8) WPS/geogrid, but with dominant categories recomputed in real
- **2**: GRIB data from another model (only if arrays VEGCAT/SOILCAT exist)
<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pxlsm_smois_init</td>
<td>3</td>
<td>(default beginning in V3.8) use dominant land and soil categories from WPS/geogrid</td>
</tr>
<tr>
<td>(max_dom)</td>
<td></td>
<td>Pleim-Xu land-surface model soil moisture initialization option</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>from analysis</td>
</tr>
<tr>
<td>pxlsm_smois_init</td>
<td>1</td>
<td>(default) from LANDUSE.TBL (SLMO, or moisture availability)</td>
</tr>
<tr>
<td>(max_dom)</td>
<td></td>
<td>number of land categories in input data</td>
</tr>
<tr>
<td>num_land_cat</td>
<td>24</td>
<td>(default prior to V3.8) for USGS</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>for MODIS</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>for USGS if including lake category</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>(default beginning with V3.8) for MODIS if including lake category</td>
</tr>
<tr>
<td>num_soil_cat</td>
<td>40</td>
<td>NLCD2006 (North America only)</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>number of soil categories in input data</td>
</tr>
<tr>
<td>usemonalb</td>
<td>.true.</td>
<td>use monthly albedo map instead of table values (recommended for sst_update=1)</td>
</tr>
<tr>
<td></td>
<td>.false.</td>
<td>(default) use table values</td>
</tr>
<tr>
<td>rdmaxalb</td>
<td>.true.</td>
<td>(default) use snow albedo from geogrid</td>
</tr>
<tr>
<td></td>
<td>.false.</td>
<td>use snow albedo from table</td>
</tr>
<tr>
<td>rdlai2d (data available to use this option since V3.6)</td>
<td>.true.</td>
<td>use LAI (Leaf Area Index) from input data. If sst_update is 1, then LAI will also appear in wrflowinwp file</td>
</tr>
<tr>
<td></td>
<td>.false.</td>
<td>(default) use LAI from table</td>
</tr>
<tr>
<td>seaice_threshold</td>
<td>100</td>
<td>If skin temp (TSK) is less than this value, water points are changed to sea ice. If water point + 5-layer slab scheme, set to land point and permanent ice; if water point + Noah scheme, set to land point, permanent ice, set temps from 2 m to surface, and set smois and sh2o. The default value was changed to 100. From 271 in 3.5.1 to avoid mixed-up use with fractional seaice input. Used by sf_surface_physics = 1, 2, 3, 4, 8</td>
</tr>
<tr>
<td>sst_update</td>
<td></td>
<td>option to use time-varying SST,</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>seaice, vegetation fraction, and albedo during a model simulation (set before running <code>real.exe</code>)</td>
<td>0</td>
<td>(default) no SST update</td>
</tr>
<tr>
<td><code>real.exe</code> will create <code>wrfflowinp</code> file(s) at the same time interval as the available input data. These files contain SST, XICE, ALBEDO, and VEGFRA. Also set <code>auxinput4_inname = &quot;wrfflowinp_d&lt;domain&gt;&quot;</code>, <code>auxinput4_interval</code> and (in V3.2) <code>io_form_auxinput4</code> in namelist section &amp;time_control</td>
<td>1</td>
<td>update deep layer soil temperature, useful for long simulations (multi-year runs; default is 0 = off)</td>
</tr>
<tr>
<td>tmn_update</td>
<td>1</td>
<td>days over which tmn (deep layer soil temp) is computed using skin temperature</td>
</tr>
<tr>
<td>lagday</td>
<td>150</td>
<td>calculate skin SST, useful for long simulations (multi-year runs; default is 0 = off)</td>
</tr>
<tr>
<td>bucket_mm</td>
<td>-1</td>
<td>bucket reset values for water accumulation (unit in mm), useful for long simulations (multi-year runs)</td>
</tr>
<tr>
<td>bucket_j</td>
<td>-1</td>
<td>bucket reset value for energy accumulations (unit in Joules); useful for long simulations (multi-year runs)</td>
</tr>
<tr>
<td>slope_rad (max_dom)</td>
<td>1</td>
<td>use slope-dependent radiation; for <code>ra_sw_physics</code></td>
</tr>
<tr>
<td>topo_shading (max_dom)</td>
<td>1</td>
<td>applies neighboring-point shadow effects for <code>ra_sw_physics</code></td>
</tr>
<tr>
<td>shadlen</td>
<td>25000</td>
<td>maximum length of orographic shadow (in meters); use with <code>topo_shading=1</code></td>
</tr>
<tr>
<td>sf_ocean_physics (replacing omlcall beginning with V3.5)</td>
<td></td>
<td>activate ocean model</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>off</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>activate a simple ocean mixed layer (oml) model</td>
</tr>
<tr>
<td><em>(new since V3.5)</em></td>
<td>2</td>
<td>activate a 3D PWP ocean model</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>omdt</td>
<td>1</td>
<td>3D PWP time step (minutes). It can be set to the same as the WRF time step in corresponding nested grids, but omdt should be no less than 1.0 minute.</td>
</tr>
<tr>
<td>oml_hml0 (for sf_ocean_physics=1)</td>
<td>≥ 0</td>
<td>initial ocean mixed layer depth value (m); constant everywhere (50 is default)</td>
</tr>
<tr>
<td></td>
<td>&lt; 0</td>
<td>use input</td>
</tr>
<tr>
<td>oml_gamma (for sf_ocean_physics=1)</td>
<td>0.14</td>
<td>(K m⁻¹) lapse rate in deep water (below the mixed layer) for oml</td>
</tr>
<tr>
<td>oml_relaxation_time <em>(new since V3.8)</em></td>
<td>0.</td>
<td>relaxation time (seconds) of mixed layer ocean model back to original values (e.g. value: 259200 sec - 3 days)</td>
</tr>
<tr>
<td>ocean_levels (for sf_ocean_physics=2)</td>
<td>30</td>
<td>number of vertical levels in 3D ocean model</td>
</tr>
<tr>
<td>isftcflx</td>
<td>alternative Ck (exchange coefficient for temp and moisture), Cd (drag coefficient for momentum) formulation for tropical storm application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(default) off for Ck</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Donelan Cd + constant ( Z_{0q} ) for Ck</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Donelan Cd + Garratt Ck</td>
</tr>
<tr>
<td>fractional_seaice</td>
<td>1</td>
<td>treats seaice as a fractional field; works with ( \text{sf_sfclay_physics} = 1, 2, 4, 5, \text{or} \ 7 ) Also set ( \text{seaice_threshold} = 0 ).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) either ice or no ice flag</td>
</tr>
<tr>
<td>seaice_albedo_opt <em>(new since V3.4)</em></td>
<td>option to set albedo over sea ice</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>seaice albedo is a constant value from namelist option ( \text{seaice_albedo_default} )</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>seaice albedo is a function of air temp, skin temp, and snow</td>
</tr>
</tbody>
</table>
### seaice_snowdepth_opt

*(new since V3.5)*

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>snow depth on sea ice is bounded by seaice_snowdepth_min and seaice_snowdepth_max</td>
<td>0</td>
<td>snow depth on sea ice and is read in from input array SNOWSI (bounded by seaice_snowdepth_min and seaice_snowdepth_max)</td>
</tr>
<tr>
<td>snow depth on sea ice read in from input array SNOWSI (bounded by seaice_snowdepth_min and seaice_snowdepth_max)</td>
<td>1</td>
<td>snow depth on sea ice read in from input array SNOWSI (bounded by seaice_snowdepth_min and seaice_snowdepth_max)</td>
</tr>
</tbody>
</table>

### seaice_snowdepth_max

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximum allowed accumulation of snow (m) on sea ice</td>
<td>1.e10</td>
<td>maximum allowed accumulation of snow (m) on sea ice</td>
</tr>
</tbody>
</table>

### seaice_snowdepth_min

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum snow depth (m) on sea ice</td>
<td>0.001</td>
<td>minimum snow depth (m) on sea ice</td>
</tr>
</tbody>
</table>

### seaice_thickness_opt

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>option for treating seaice thickness</td>
<td>0</td>
<td>seaice thickness is uniform value taken from namelist variable seaice_thickness_default</td>
</tr>
<tr>
<td>seaice thickness is read in from input variable ICEDEPTH</td>
<td>1</td>
<td>seaice thickness is read in from input variable ICEDEPTH</td>
</tr>
</tbody>
</table>

### seaice_thickness_default

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default value of seaice thickness for seaice_thickness_opt=0</td>
<td>3.0</td>
<td>default value of seaice thickness for seaice_thickness_opt=0</td>
</tr>
</tbody>
</table>

### prec_acc_dt (max_dom)

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket reset time interval between outputs for cumulus or grid-scale precipitation (in minutes). If set &gt;0, this will output 3 new 2d fields: prec_acc_c, prec_acc_nc, and snow_acc_nc (descriptions of these can be found in the Registry.EM_COMMON file)</td>
<td>0.</td>
<td>bucket reset time interval between outputs for cumulus or grid-scale precipitation (in minutes). If set &gt;0, this will output 3 new 2d fields: prec_acc_c, prec_acc_nc, and snow_acc_nc (descriptions of these can be found in the Registry.EM_COMMON file)</td>
</tr>
</tbody>
</table>

### traj_opt (new since V3.5)

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate forward trajectories (default 0)</td>
<td>1</td>
<td>activate forward trajectories (default 0)</td>
</tr>
</tbody>
</table>

### num_traj

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of trajectories to be released</td>
<td>0</td>
<td>number of trajectories to be released</td>
</tr>
</tbody>
</table>

*The following are options for the lake model*

### sf_lake_physics (max_dom) (new since V3.6)

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lake model on (default is 0 = off)</td>
<td>1</td>
<td>lake model on (default is 0 = off)</td>
</tr>
</tbody>
</table>

### lakedepth_default (max_dom)

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(default) lake depth (in meters). If there is no lake depth information in the input data, then lake depth is assumed to be 50m</td>
<td>50</td>
<td>(default) lake depth (in meters). If there is no lake depth information in the input data, then lake depth is assumed to be 50m</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>lake_min_elev (max_dom)</code></td>
<td>5</td>
<td>(default) minimum elevation of lakes; may be used to determine whether a water point is a lake in the absence of a lake category. If the landuse type includes 'lake' (i.e., Modis_lake and USGS_LAKE); this variable is of no effects</td>
</tr>
<tr>
<td><code>use_lakedepth</code></td>
<td>1</td>
<td>(default) option to use lake depth data. Lake depth data is available beginning in the V3.6 geogrid program. If the lake depth data was not processed, but this switch is set to 1, the program will stop and tell the user to go back to geogrid program.</td>
</tr>
<tr>
<td><code>lightning_option (max_dom) (new since V3.5)</code></td>
<td>0</td>
<td>Lightening parameterization option to allow flash rate prediction without chemistry. Requires do_radar_ref on.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>PR92 based on maximum w, redistributes flashes within dBZ &gt; 20 (for convection resolved runs)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>PR92 based on 20 dBZ top, redistributes flashes within dBZ &gt; 20 (for convection resolved runs)</td>
</tr>
<tr>
<td>(New since V3.6.1)</td>
<td>3</td>
<td>Predicting the potential for lightning activity (based on Yair et al., 2010)</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>PR92 based on level of neutral buoyancy from convective parameterization (for scale where a CPS is used, intended for use at 10 &lt; dx &lt; 50 km)</td>
</tr>
<tr>
<td><code>lightning_dt (max_dom) (new since V3.5)</code></td>
<td>0.</td>
<td>time interval (seconds) for calling lightning parameterization. Default uses model time step</td>
</tr>
<tr>
<td><code>lightning_start_seconds (max_dom) (new since V3.5)</code></td>
<td>0.</td>
<td>start time for calling lightning parameterization. Recommends at least 10 minutes for spin-up</td>
</tr>
<tr>
<td><code>flashrate_factor (max_dom) (new since V3.5)</code></td>
<td>1.0</td>
<td>Factor to adjust the predicted number of flashes. Recommends 1.0 for lightning_option = 11 between dx=10 and 50 km. Manual tuning recommended for all other</td>
</tr>
<tr>
<td>Parameter</td>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>cellcount_method</strong> (max_dom)</td>
<td>Options independently for each nest.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Method for counting storm cells. Used by CRM options (lightning_options=1,2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Model determines method used</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Tile-wide, appropriate for large domains</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Domain-wide, appropriate for single-storm domains</td>
</tr>
<tr>
<td><strong>cldtop_adjustment</strong> (max_dom)</td>
<td>0.</td>
<td>Adjustment from LNB in km. Used by lightning_option=11. Default is 0, but recommends 2 km</td>
</tr>
<tr>
<td><strong>iccg_method</strong> (max_dom)</td>
<td>0</td>
<td>Default method depending on lightning option, currently all options use iccg_method=2 by default</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Constant everywhere, set with namelist options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iccg_prescribed (num</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Coarsely prescribed 1995-1999 NLDN/OTD climatology based on Boccippio et al. (2001)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Parameterization by Price and Rind (1993) based on cold-cloud depth</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Gridded input via arrays iccg_in (num</td>
</tr>
<tr>
<td><strong>iccg_prescribed_num</strong> (max_dom)</td>
<td>0.</td>
<td>Numerator of user-specified prescribed IC:CG</td>
</tr>
<tr>
<td><strong>iccg_prescribed_den</strong> (max_dom)</td>
<td>1.</td>
<td>Denominator of user-specified prescribed IC:CG</td>
</tr>
<tr>
<td><strong>For Wind Turbine Drag Parameterization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>windfarm_opt</strong> (new since V3.3)</td>
<td>1</td>
<td>Simulates the effects of wind turbines in the atmospheric evolution (default is 0 = off)</td>
</tr>
<tr>
<td><strong>windfarm_ij</strong> (new since V3.3)</td>
<td></td>
<td>Whether to use lat-lon or i-j coordinate as wind turbine</td>
</tr>
<tr>
<td>locations</td>
<td>0</td>
<td>(default) the coordinates of the turbines are defined in terms of lat-lon</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>the coordinates of the turbines are defined in terms of grid points</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&amp;stoch</th>
<th>1</th>
<th>Stochastic kinetic-energy backscatter scheme turned on (0=off) (previously stoch_force_opt; new since V3.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>skebs (max_dom) (name change new since V3.7)</td>
<td>1</td>
<td>Stochastic kinetic-energy backscatter scheme turned on (0=off) (previously stoch_force_opt; new since V3.3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&amp;noah_mp</th>
<th>Options for NoahMP LSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>dveg</td>
<td>dynamic vegetation option</td>
</tr>
<tr>
<td>1</td>
<td>off [LAI (Leaf Area Index) from table; FVEG (veg fraction) = shdfac (model variable for veg fraction)]</td>
</tr>
<tr>
<td>2</td>
<td>on (LAI predicted; FVEG calculated)</td>
</tr>
<tr>
<td>3</td>
<td>off (LAI from table; FVEG calculated)</td>
</tr>
<tr>
<td>4</td>
<td>(default) off (LAI from table; FVEG = maximum veg. fraction)</td>
</tr>
<tr>
<td>(new since V3.7)</td>
<td>5</td>
</tr>
<tr>
<td>opt_crs</td>
<td>stomatal resistance option</td>
</tr>
<tr>
<td>1</td>
<td>(default) Ball-Berry</td>
</tr>
<tr>
<td>2</td>
<td>Jarvis</td>
</tr>
<tr>
<td>opt_sfc</td>
<td>surface layer drag coefficient calculation</td>
</tr>
<tr>
<td>1</td>
<td>(default) Monin-Obukhov</td>
</tr>
<tr>
<td>2</td>
<td>original Noah</td>
</tr>
<tr>
<td>(option removed in 3.7)</td>
<td>3</td>
</tr>
<tr>
<td>(option removed in 3.7)</td>
<td>4</td>
</tr>
<tr>
<td>opt_btr</td>
<td>soil moisture factor for stomatal resistance</td>
</tr>
<tr>
<td>1</td>
<td>Noah</td>
</tr>
<tr>
<td>2</td>
<td>CLM</td>
</tr>
<tr>
<td>3</td>
<td>SSiB</td>
</tr>
<tr>
<td>opt_run</td>
<td>1</td>
</tr>
<tr>
<td>------------------</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>opt_frz</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>opt_inf</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>opt_rad</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>opt_alb</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>opt_snf</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>opt_tbot</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>opt_stc</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>opt_gla (new since V3.8)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>opt_rsf (new since V3.8)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>Sakaguchi and Zeng 2009</td>
</tr>
<tr>
<td>2</td>
<td>Sellers 1992</td>
</tr>
<tr>
<td>3</td>
<td>adjusted Sellers to decrease RSURF for wet soil</td>
</tr>
<tr>
<td>4</td>
<td>option 1 for non-snow; rsurf = rsurf_snow for snow (set in MPTABLE -added in V3.8)</td>
</tr>
</tbody>
</table>

*(new since V3.7)*

3 semi-implicit where Ts uses snow cover fraction

### &fdda

Options for grid, obs and spectral nudging

**For Grid Nudging**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid_fdda (max_dom)</td>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td>1</td>
<td>grid analysis nudging on</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>spectral analysis nudging option</td>
<td></td>
</tr>
<tr>
<td>gfdda_inname</td>
<td>&quot;wrf_fdda_d&lt;domain&gt;&quot;</td>
<td>name of fdda input file that will be produced when running real</td>
</tr>
<tr>
<td>gfdda_interval_m (max_dom)</td>
<td>360</td>
<td>time interval (in mins) between analysis times</td>
</tr>
<tr>
<td>gfdda_end_h (max_dom)</td>
<td>6</td>
<td>time (hr) to stop nudging after the start of the forecast</td>
</tr>
<tr>
<td>io_form_gfdda</td>
<td>2</td>
<td>netCDF format</td>
</tr>
<tr>
<td>4</td>
<td>PHD5 format</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GRIB1 format</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>GRIB2 format</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>pnetCDF format</td>
<td></td>
</tr>
<tr>
<td>fgdt (max_dom)</td>
<td>0</td>
<td>calculation frequency (in mins) for analysis nudging; 0=every time step (which is recommended)</td>
</tr>
<tr>
<td>if_no_pbl_nudging_uv (max_dom)</td>
<td>0</td>
<td>(default) nudging in the PBL</td>
</tr>
<tr>
<td>1</td>
<td>no nudging of u and v in the PBL</td>
<td></td>
</tr>
<tr>
<td>if_no_pbl_nudging_t (max_dom)</td>
<td>0</td>
<td>(default) nudging in the PBL</td>
</tr>
<tr>
<td>1</td>
<td>no nudging of temp in the PBL</td>
<td></td>
</tr>
<tr>
<td>if_no_pbl_nudging_q (max_dom)</td>
<td>0</td>
<td>(default) nudging in the PBL</td>
</tr>
<tr>
<td>1</td>
<td>no nudging of qvapor in the PBL</td>
<td></td>
</tr>
<tr>
<td>if_zfac_uv (max_dom)</td>
<td>0</td>
<td>(default) nudge u and v in all layers</td>
</tr>
<tr>
<td>1</td>
<td>limit nudging to levels above k_zfac_uv</td>
<td></td>
</tr>
</tbody>
</table>
| k_zfac_uv | 10 | model level below which nudging
### if_zfac_t (max_dom)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(default) nudge temp in all layers</td>
</tr>
<tr>
<td>1</td>
<td>limit nudging to levels above k_zfac_t</td>
</tr>
</tbody>
</table>

### k_zfac_t

- model level below which nudging is switched off for temp

### if_zfac_q (max_dom)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(default) nudge qvapor in all layers</td>
</tr>
<tr>
<td>1</td>
<td>limit nudging to levels above k_zfac_q</td>
</tr>
</tbody>
</table>

### k_zfac_q

- model level below which nudging is switched off for qvapor

### guv (max_dom)

- nudging coefficient for u and v (s^-1)

### gt (max_dom)

- nudging coefficient for temp (s^-1)

### gq (max_dom)

- nudging coefficient for qvapor (s^-1)

### if_ramping

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(default) nudging ends as a step function</td>
</tr>
<tr>
<td>1</td>
<td>ramping nudging down at the end of the period</td>
</tr>
</tbody>
</table>

### dtramp_min

- time (min) for ramping function; 60.0 = ramping starts at last analysis time, -60.0 = ramping ends at last analysis time

### grid_sfdda (max_dom)

- surface fdda switch

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td>1</td>
<td>nudging selected surface fields</td>
</tr>
</tbody>
</table>

### sgfdda_inname

- "wrsfdda_d<domain >"

### sgfdda_interval_m (max_dom)

- time interval (in mins) between surface analysis times

### sgfdda_end_h (max_dom)

- time (in hours) to stop surface nudging after start of the forecast

### io_form_sgfdda

- surface analysis format

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(2=netCDF)</td>
</tr>
</tbody>
</table>

### guv_sfc (max_dom)

- nudging coefficient for u and v (s^-1)

### gt_sfc (max_dom)

- nudging coefficient for temp (s^-1)

### gq_sfc (max_dom)

- nudging coefficient for qvapor (s^-1)

### rinblw

- radius of influence used to determine the confidence (or weights) for the analysis, which is based on the distance between the grid point to the nearest obs. The analysis without nearby
<table>
<thead>
<tr>
<th><strong>(For Spectral Nudging)</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fgdtdzero (max_dom)</td>
<td>1</td>
<td>nudging tendencies are set to zero in between fdda calls</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) not active</td>
</tr>
<tr>
<td>if_no_pbl_nudging_ph (max_dom)</td>
<td>1</td>
<td>no nudging of ph in the PBL</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) nudging of ph in the PBL</td>
</tr>
<tr>
<td>if_zfac_ph (max_dom)</td>
<td>0</td>
<td>(default) nudge ph in all layers</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>limit nudging to levels above k_zfac_ph</td>
</tr>
<tr>
<td>k_zfac_ph</td>
<td>10</td>
<td>model level below which nudging is switched off for water ph</td>
</tr>
<tr>
<td>gph (max_dom)</td>
<td>0.0003</td>
<td>nudging coefficient for ph ($s^{-1}$)</td>
</tr>
<tr>
<td>dk_zfac_uv (max_dom)</td>
<td>1</td>
<td>depth in k between k_zfac_uv to dk_zfac_uv where nuding increases linearly to full strength</td>
</tr>
<tr>
<td>dk_zfac_t (max_dom)</td>
<td>1</td>
<td>depth in k between k_zfac_t to dk_zfac_t where nuding increases linearly to full strength</td>
</tr>
<tr>
<td>dk_zfac_ph (max_dom)</td>
<td>1</td>
<td>depth in k between k_zfac_ph to dk_zfac_ph where nuding increases linearly to full strength</td>
</tr>
<tr>
<td>xwavenum</td>
<td>3</td>
<td>top wave number to nudge in x-direction (0 is default)</td>
</tr>
<tr>
<td>ywavenum</td>
<td>3</td>
<td>top wave number to nudge in y-direction (0 is default)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>(For Obs Nudging)</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>obs_nudge_opt (max_dom)</td>
<td>1</td>
<td>obs-nudging fdda on for each domain (default is 0=off); also must set auxinput11_interval and auxinput11_end_h under &amp;time_control</td>
</tr>
<tr>
<td>max_obs</td>
<td>150000</td>
<td>max number of observations used on a domain during any given time window (default is 0)</td>
</tr>
<tr>
<td>fdda_start (max_dom)</td>
<td>0.</td>
<td>obs nudging start time (min)</td>
</tr>
<tr>
<td>fdda_end (max_dom)</td>
<td>180.</td>
<td>obs nudging end time (min)</td>
</tr>
<tr>
<td>obs_nudge_wind (max_dom)</td>
<td>1</td>
<td>nudge wind on</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td>obs_coef_wind (max_dom)</td>
<td>6.e-4</td>
<td>nudging coefficient for wind ($s^{-1}$)</td>
</tr>
<tr>
<td>obs_nudge_temp</td>
<td>1</td>
<td>nudge temperature on</td>
</tr>
<tr>
<td>(max_dom)</td>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>obs_coef_temp (max_dom)</td>
<td>6.e-4</td>
<td>nudging coefficient for temp ($s^{-1}$)</td>
</tr>
<tr>
<td>obs_nudge_mois (max_dom)</td>
<td>1</td>
<td>nudge water vapor mixing ratio</td>
</tr>
<tr>
<td>obs_coef_mois (max_dom)</td>
<td>0</td>
<td>(default) off</td>
</tr>
<tr>
<td>obs_coef_pstr</td>
<td>6.e-4</td>
<td>nudging coefficient for water vapor mixing ratio ($s^{-1}$)</td>
</tr>
<tr>
<td>obs_coef_pstr</td>
<td>0</td>
<td>nudging coefficient for surface pressure ($s^{-1}$) (not used)</td>
</tr>
<tr>
<td>obs_rinxy</td>
<td>200.</td>
<td>horizontal radius of influence (km; 200 is a reasonable value, but should be adjusted, based on data density)</td>
</tr>
<tr>
<td>obs_rinsig</td>
<td>0.1</td>
<td>vertical radius of influence in eta (0.1 is a reasonable value, but should be adjusted, based on data density)</td>
</tr>
<tr>
<td>obs_twindo (max_dom)</td>
<td>0.666667</td>
<td>half-period time window over which an observation will be used for nudging (hrs)</td>
</tr>
<tr>
<td>obs_npfi</td>
<td>10</td>
<td>frequency in coarse grid timesteps for diagnostic prints</td>
</tr>
<tr>
<td>obs_ionf (max_dom)</td>
<td>1</td>
<td>frequency in coarse grid timesteps for obs input and err calc</td>
</tr>
<tr>
<td>obs_idynin</td>
<td>1</td>
<td>for dynamic initialization using a ramp-down function to gradually turn off the FDDA before the pure forecast (default is 0=off)</td>
</tr>
<tr>
<td>obs_dtramp</td>
<td>40.</td>
<td>time period (mins) over which the nudging is ramped down from one to zero</td>
</tr>
<tr>
<td>obs_prt_max</td>
<td>1000</td>
<td>maximum allowed obs entries in diagnostic printout</td>
</tr>
<tr>
<td>obs_prt_freq (max_dom)</td>
<td>1000</td>
<td>frequency in obs index for diagnostic printout</td>
</tr>
<tr>
<td>obs_ipf_in4dob</td>
<td>.true.</td>
<td>print obs input diagnostics (default is .false.=off)</td>
</tr>
<tr>
<td>obs_ipf_errrob</td>
<td>.true.</td>
<td>print obs error diagnostics (default is .false.=off)</td>
</tr>
<tr>
<td>obs_ipf_nudob</td>
<td>.true.</td>
<td>print obs nudge diagnostics (default is .false.=off)</td>
</tr>
<tr>
<td>obs_ipf_init</td>
<td>.true.</td>
<td>(default) enable obs printed warning messages</td>
</tr>
<tr>
<td>obs_no_pbl_nudge_uv (max_dom)</td>
<td>1</td>
<td>no wind-nudging within the PBL</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>obs_no_pbl_nudge_t (max_dom)</td>
<td>0</td>
<td>(default) wind-nudging within the PBL</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>no temperature-nudging within the PBL</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) temperature-nudging within the PBL</td>
</tr>
<tr>
<td>obs_no_pbl_nudge_q (max_dom)</td>
<td>1</td>
<td>no moisture-nudging within the PBL</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(default) no moisture-nudging within the PBL</td>
</tr>
<tr>
<td>obs_nudgezfullr1_uv</td>
<td>50</td>
<td>Vertical influence full weight height for LML obs, regime 1, winds</td>
</tr>
<tr>
<td>obs_nudgezrampr1_uv</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 1, winds</td>
</tr>
<tr>
<td>obs_nudgezfullr2_uv</td>
<td>50</td>
<td>Vertical influence full weight height for LML obs, regime 2, winds</td>
</tr>
<tr>
<td>obs_nudgezrampr2_uv</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 2, winds</td>
</tr>
<tr>
<td>obs_nudgezfullr4_uv</td>
<td>-5000</td>
<td>Vertical influence full weight height for LML obs, regime 4, winds</td>
</tr>
<tr>
<td>obs_nudgezrampr4_uv</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 4, winds</td>
</tr>
<tr>
<td>obs_nudgezfullr1_t</td>
<td>50</td>
<td>Vertical influence full weight height for LML obs, regime 1, temperature</td>
</tr>
<tr>
<td>obs_nudgezrampr1_t</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 1, temperature</td>
</tr>
<tr>
<td>obs_nudgezfullr2_t</td>
<td>50</td>
<td>Vertical influence full weight height for LML obs, regime 2, temperature</td>
</tr>
<tr>
<td>obs_nudgezrampr2_t</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 2, temperature</td>
</tr>
<tr>
<td>obs_nudgezfullr4_t</td>
<td>-5000</td>
<td>Vertical influence full weight height for LML obs, regime 4, temperature</td>
</tr>
<tr>
<td>obs_nudgezrampr4_t</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 4, temperature</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>obs_nudgezfullr1_q</td>
<td>50</td>
<td>Vertical influence full weight height for LML obs, regime 1, temperature</td>
</tr>
<tr>
<td>obs_nudgezrampr1_q</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 1, temperature</td>
</tr>
<tr>
<td>obs_nudgezfullr2_q</td>
<td>50</td>
<td>Vertical influence full weight height for LML obs, regime 2, temperature</td>
</tr>
<tr>
<td>obs_nudgezrampr2_q</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 2, temperature</td>
</tr>
<tr>
<td>obs_nudgezfullr4_q</td>
<td>-5000</td>
<td>Vertical influence full weight height for LML obs, regime 4, temperature</td>
</tr>
<tr>
<td>obs_nudgezrampr4_q</td>
<td>50</td>
<td>Vertical influence ramp-to-zero height for LML obs, regime 4, temperature</td>
</tr>
<tr>
<td>obs_nudgefullmin</td>
<td>50</td>
<td>Minimum depth (m) through which vertical influence function remains 1.0</td>
</tr>
<tr>
<td>obs_nudgezrampmin</td>
<td>50</td>
<td>Minimum depth (m) through which vert infl fcn decreases from 1 to 0</td>
</tr>
<tr>
<td>obs_nudgezmax</td>
<td>3000</td>
<td>Max depth (m) in which vert infl function is non-zero</td>
</tr>
<tr>
<td>obs_sfcfact</td>
<td>1.0</td>
<td>Scale factor applied to time window for surface obs</td>
</tr>
<tr>
<td>obs_sfcfacr</td>
<td>1.0</td>
<td>Scale factor applied to horiz radius of influence for surface obs</td>
</tr>
<tr>
<td>obs_dpsmx</td>
<td>7.5</td>
<td>Max pressure change (cb) allowed within horiz radius of influence</td>
</tr>
<tr>
<td>obs_sfc_scheme_horiz</td>
<td></td>
<td>Horizontal spreading scheme for surface obs</td>
</tr>
<tr>
<td>obs_sfc_scheme_horiz</td>
<td>0</td>
<td>(Default) WRF scheme</td>
</tr>
<tr>
<td>obs_sfc_scheme_horiz</td>
<td>1</td>
<td>Original MM5 scheme</td>
</tr>
<tr>
<td>obs_sfc_scheme_vert</td>
<td></td>
<td>Vertical spreading scheme for surface obs</td>
</tr>
<tr>
<td>obs_sfc_scheme_vert</td>
<td>0</td>
<td>(Default) regime vif scheme</td>
</tr>
<tr>
<td>obs_sfc_scheme_vert</td>
<td>1</td>
<td>Original scheme (simple scheme)</td>
</tr>
<tr>
<td>obs_max_sndng_gap</td>
<td>20</td>
<td>Max allowed pressure gap between soundings for interpolation (cb)</td>
</tr>
<tr>
<td>obs_scl_neg_qv_innov</td>
<td></td>
<td>Diffusion, damping options</td>
</tr>
<tr>
<td>(new since V3.6)</td>
<td>0</td>
<td>0: default behavior</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1: prevent nudging toward negative Qv</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td><strong>Value</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>advection options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rk_ord</td>
<td></td>
<td>time-integration scheme option</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Runge-Kutta 2nd order</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>(default/recommended) Runge-Kutta 3rd order</td>
</tr>
<tr>
<td>diff_opt (max_dom)</td>
<td></td>
<td>turbulence and mixing option</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>no turbulence or explicit spatial numerical filters (km_opt is ignored)</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>(default) evaluates 2nd order diffusion term on coordinate surfaces, uses kvdif for vertical diffusion unless PBL option is used, may be used with km_opt = 1 (recommended for real-data case) and 4</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>evaluates mixing terms in physical space (stress form) (x,y,z); turbulence parameterization is chosen by specifying km_opt</td>
</tr>
<tr>
<td>km_opt (max_dom)</td>
<td></td>
<td>eddy coefficient option</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>(default) constant (use khdif and kvdif)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1.5 order TKE closure (3D) ** Not recommended for DX &gt; 2 km</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Smagorinsky first order closure (3D) **Not recommended for DX &gt; 2 km</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>horizontal Smagorinsky first order closure (recommended for real-data case)</td>
</tr>
<tr>
<td>diff_6th_opt (max_dom)</td>
<td></td>
<td>6th-order numerical diffusion</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>(default) no 6th-order diffusion</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>6th-order numerical diffusion</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>6th-order numerical diffusion, but prohibit up-gradient diffusion</td>
</tr>
<tr>
<td>diff_6th_factor</td>
<td>0.12</td>
<td>6th-order numerical diffusion non-dimensional rate (max value 1.0 corresponds to complete removal of 2dx wave in one timestep)</td>
</tr>
<tr>
<td>damp_opt</td>
<td></td>
<td>upper-level damping flag</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>(default) no damping</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>with diffusive damping; maybe used for real-data cases (dampcoeff nondimensional ~</td>
</tr>
</tbody>
</table>
### Table: Model Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.01 to 0.1</td>
<td>with Rayleigh damping (dampcoef inverse time scale [1/s], e.g. 0.003)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>with Rayleigh damping (dampcoef inverse time scale [1/s], e.g. 0.2; for real-data cases)</td>
</tr>
<tr>
<td>use_theta_m (new since V3.7)</td>
<td>1</td>
<td>uses theta(1+1.61Qv); (default 0=off)</td>
</tr>
<tr>
<td>use_q_diabatic (new since V3.7)</td>
<td>1</td>
<td>includes QV and QC tendencies in advection; this helps to produce correct solution in an idealized ‘moist benchmark’ test case (Bryan, 2014). In real data testing, time step needs to be reduced to maintain a stable solution (default 0=off)</td>
</tr>
<tr>
<td>c_s (new since V3.7)</td>
<td>0.25</td>
<td>(default) Smagorinsky coeff</td>
</tr>
<tr>
<td>c_k (new since V3.7)</td>
<td>0.15</td>
<td>(default) TKE coeff</td>
</tr>
<tr>
<td>zdamp (max_dom)</td>
<td>5000</td>
<td>damping depth (m) from model top</td>
</tr>
<tr>
<td>dampcoef (max_dom)</td>
<td>0.</td>
<td>damping coefficient (see damp_opt)</td>
</tr>
<tr>
<td>w_damping</td>
<td>0</td>
<td>vertical velocity damping flag (for operational use)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>with damping</td>
</tr>
<tr>
<td>base_pres</td>
<td>100000</td>
<td>base state surface pressure (Pa); real only., not recommended to change.</td>
</tr>
<tr>
<td>base_temp</td>
<td>290.</td>
<td>base state temperature (K); real only</td>
</tr>
<tr>
<td>base_lapse</td>
<td>50.</td>
<td>real-data ONLY, lapse rate (K), not recommended to change</td>
</tr>
<tr>
<td>iso_temp</td>
<td>200.</td>
<td>isothermal temperature in stratosphere; enables model to be extended to 5 mb; real only. Default value changed to 200 since V3.5</td>
</tr>
<tr>
<td>base_pres_strat (New since V3.6.1)</td>
<td>0.</td>
<td>real data, em ONLY, base state pressure (Pa) at bottom of the stratosphere, US Standard atmosphere 55 hPa. (default value set to 0 mb in V3.7)</td>
</tr>
<tr>
<td>base_lapse_strat (new since V3.6.1)</td>
<td>-11.</td>
<td>(default) real-data; em ONLY, base state lapse rate (dT/d(lnP)) in stratosphere, approx to US standard atmosphere -12K</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>use_baseparm_fr_nml</td>
<td>.false.</td>
<td>for backward compatibility; to use with old wrfinput file produced prior to V3.4</td>
</tr>
<tr>
<td>use_input_w (new since V3.3.1)</td>
<td>.false.</td>
<td>whether to use vertical velocity from input file</td>
</tr>
<tr>
<td>khdif (max_dom)</td>
<td>0.0</td>
<td>horizontal diffusion constant (m$^2$/s)</td>
</tr>
<tr>
<td>kvdif (max_dom)</td>
<td>0.0</td>
<td>vertical diffusion constant (m$^2$/s)</td>
</tr>
<tr>
<td>smdiv (max_dom)</td>
<td>0.1</td>
<td>divergence damping (0.1 is typical)</td>
</tr>
<tr>
<td>emdiv (max_dom)</td>
<td>0.01</td>
<td>external-mode filter coef for mass coordinate model (0.01 is typical for real-data cases)</td>
</tr>
<tr>
<td>epssm (max_dom)</td>
<td>0.1</td>
<td>time off-centering for vertical sound waves</td>
</tr>
<tr>
<td>non-hydrostatic (max_dom)</td>
<td>.true.</td>
<td>(default) running the model in non-hydrostatic mode</td>
</tr>
<tr>
<td>pert_coriolis (max_dom)</td>
<td>.false.</td>
<td>running the model in hydrostatic mode</td>
</tr>
<tr>
<td>top_lid (max_dom)</td>
<td>.false.</td>
<td>zero vertical motion at top of domain (only for idealized)</td>
</tr>
<tr>
<td>mix_full_fields</td>
<td>.false.</td>
<td>used with diff_opt = 2; value of .true. is recommended, except for highly idealized numerical tests; damp_opt must not be =1 if .true. is chosen; .false. means subtract 1D base-state profile before mixing (only for idealized)</td>
</tr>
<tr>
<td>mix_isotropic (max_dom)</td>
<td>0</td>
<td>(default) anisotropic vertical/horizontal diffusion</td>
</tr>
<tr>
<td>mix_upper_bound (max_dom)</td>
<td>0.1</td>
<td>non-dimensional upper limit for diffusion coefficients; for km_opt = 2, 3</td>
</tr>
<tr>
<td>h_mom_adv_order (max_dom)</td>
<td>5</td>
<td>horizontal momentum advection order; 5 (default) = 5th, etc.</td>
</tr>
<tr>
<td>v_mom_adv_order (max_dom)</td>
<td>3</td>
<td>vertical momentum advection order; 3 (default) = 3rd, etc.</td>
</tr>
<tr>
<td>h_sca_adv_order (max_dom)</td>
<td>5</td>
<td>horizontal scalar advection order; 5 (default) = 5th, etc.</td>
</tr>
<tr>
<td>v_sca_adv_order (max_dom)</td>
<td>3</td>
<td>vertical scalar advection order; 3 (default) = 3rd, etc.</td>
</tr>
<tr>
<td>time_step_sound</td>
<td>4</td>
<td>number of sound steps per</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>(max_dom)</td>
<td>timestep (if using a time step much larger than 6*DX (in km), increase number of sound steps (default is 0)</td>
<td></td>
</tr>
<tr>
<td>moist_adv_opt (max_dom)</td>
<td>advection options for moisture</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>simple</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(default) positive-definite</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>monotonic</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5th-order WENO (Weighted Essentially Non-Oscillatory)</td>
<td></td>
</tr>
<tr>
<td>(new since V3.4)</td>
<td>4</td>
<td>5th-order WENO with positive definite</td>
</tr>
<tr>
<td>scalar_adv_opt (max_dom)</td>
<td>advection options for scalars</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>simple</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(default) positive-definite</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>monotonic</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5th-order WENO</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5th-order WENO with positive definite</td>
<td></td>
</tr>
<tr>
<td>tke_adv_opt (max_dom)</td>
<td>advection options for TKE</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>simple</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(default) positive-definite</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>monotonic</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5th-order WENO</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5th-order WENO with positive definite</td>
<td></td>
</tr>
<tr>
<td>chem_adv_opt (max_dom)</td>
<td>advection options for chem variables</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>simple</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(default) positive definite</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>monotonic</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5th-order WENO</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5th-order WENO with positive definite</td>
<td></td>
</tr>
<tr>
<td>tracer_adv_opt (max_dom)</td>
<td>advection options for tracer variables</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>simple</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(default) positive definite</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>monotonic</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5th-order WENO</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5th-order WENO with positive definite</td>
<td></td>
</tr>
<tr>
<td>momentum_adv_opt</td>
<td>advection options for momentum</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(default) standard</td>
<td></td>
</tr>
</tbody>
</table>
| (new since V3.4)           | 3                                                                           | 5th-order WENO
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tke_drag_coefficient (max_dom)</td>
<td>0</td>
<td>Surface drag coefficient (Cd, dimensionless) for <code>diff\_opt = 2</code> only</td>
</tr>
<tr>
<td>tke_heat_flux (max_dom)</td>
<td>0</td>
<td>Surface thermal flux (H/rho*cp), Km/s, for <code>diff\_opt = 2</code> only</td>
</tr>
<tr>
<td>fft_filter_lat</td>
<td>45.</td>
<td>The latitude above which the polar filter is turned on for global model</td>
</tr>
<tr>
<td>coupled_filtering (new since V3.7)</td>
<td>.true.</td>
<td>(default) Run coupled scalar arrays are run through the polar filters</td>
</tr>
<tr>
<td>pos_def (new since V3.7)</td>
<td>.false.</td>
<td>(default) Remove negative values of scalar arrays by setting minimum value to zero</td>
</tr>
<tr>
<td>swap_pole_with_next_j (new since V3.7)</td>
<td>.false.</td>
<td>(default) Replace the entire j=1 (jds-1) with the values from j=2 (jds-2)</td>
</tr>
<tr>
<td>actual_distance_average (new since V3.7)</td>
<td>.false.</td>
<td>(default) Average the field at each i location in the j-loop with a number of grid points based on a map-factor ratio</td>
</tr>
<tr>
<td>gwd_opt</td>
<td>1</td>
<td>Gravity wave drag option; use when grid size &gt; 10 km (default is 0 = off)</td>
</tr>
<tr>
<td>do_avgflx_em (max_dom)</td>
<td>1</td>
<td>Outputs time-averaged mass-coupled advective velocities (default is 0 = off)</td>
</tr>
<tr>
<td>do_avgflx_cugd (max_dom)</td>
<td>1</td>
<td>Outputs time-averaged convective mass-fluxes from the Grell-Devenyi ensemble scheme (default is 0 = off; only takes effect if <code>do\_avgflx\_em =1, and cu\_physics = 93</code>)</td>
</tr>
<tr>
<td>sfs_opt (max_dom)</td>
<td>0</td>
<td>Nonlinear backscatter and anisotropy (NBA) (default)</td>
</tr>
<tr>
<td>m_opt (max_dom)</td>
<td>1</td>
<td>Adds output of Mij stress terms when NBA is not used (default is 0 = off)</td>
</tr>
<tr>
<td>tracer_opt (max_dom)</td>
<td>2</td>
<td>Activates 8 pre-defined tracers in the Registry (default is 0 = off)</td>
</tr>
<tr>
<td>rad_nudge</td>
<td>1</td>
<td>Option to nudge toward initial sounding in idealized TC case</td>
</tr>
</tbody>
</table>
**&bdy_control**  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec_bdy_width</td>
<td>5</td>
<td>total number of rows for specified boundary value nudging (real only)</td>
</tr>
<tr>
<td>spec_zone</td>
<td>1</td>
<td>number of points in specified zone (specified b.c. option; real only)</td>
</tr>
<tr>
<td>relax_zone</td>
<td>4</td>
<td>number of points in relaxation zone (spec b.c. option; real only)</td>
</tr>
<tr>
<td>specified</td>
<td>.true.</td>
<td>specified boundary condition; only can be used for domain 1 (default is .false.; real only)</td>
</tr>
<tr>
<td>spec_exp</td>
<td>0.</td>
<td>exponential multiplier for relaxation zone ramp for specified = .true.; default is 0. = linear ramp; 0.33 = ~3*DX exp decay factor (real only)</td>
</tr>
<tr>
<td>periodic_x (max_dom)</td>
<td>.true.</td>
<td>periodic boundary conditions in x-direction (default is .false.)</td>
</tr>
<tr>
<td>symmetric_xs (max_dom)</td>
<td>.true.</td>
<td>symmetric boundary conditions at x start (west; default is .false.)</td>
</tr>
<tr>
<td>symmetric_xe (max_dom)</td>
<td>.true.</td>
<td>symmetric boundary conditions at x end (east; default is .false.)</td>
</tr>
<tr>
<td>open_xs (max_dom)</td>
<td>.true.</td>
<td>open boundary conditions at x start (west; default is .false.)</td>
</tr>
<tr>
<td>open_xe (max_dom)</td>
<td>.true.</td>
<td>open boundary conditions at x end (east; default is .false.)</td>
</tr>
<tr>
<td>periodic_y (max_dom)</td>
<td>.true.</td>
<td>periodic boundary conditions in y-direction (default is .false.)</td>
</tr>
<tr>
<td>symmetric_ys (max_dom)</td>
<td>.true.</td>
<td>symmetric boundary conditions at y start (south; default is .false.)</td>
</tr>
<tr>
<td>symmetric_ye (max_dom)</td>
<td>.true.</td>
<td>symmetric boundary conditions at y end (north; default is .false.)</td>
</tr>
<tr>
<td>open_ys (max_dom)</td>
<td>.true.</td>
<td>open boundary conditions at y start (south; default is .false.)</td>
</tr>
<tr>
<td>open_ye (max_dom)</td>
<td>.true.</td>
<td>open boundary conditions at y end (north; default is .false.)</td>
</tr>
<tr>
<td>nested (max_dom)</td>
<td>.false., .true., .true.</td>
<td>nested boundary conditions (must be set to .true for nests)</td>
</tr>
<tr>
<td>polar (max_dom)</td>
<td>.true.</td>
<td>polar boundary condition (v=0 at polarward-most v-point) for global application (default is .false.)</td>
</tr>
<tr>
<td>constant_bc</td>
<td>.true.</td>
<td>constant boundary condition used with DFI (default is .false.)</td>
</tr>
<tr>
<td>spec_bdy_final_mu (new since V3.7)</td>
<td>1</td>
<td>call spec_bdy_final for mu (default is 0=off)</td>
</tr>
<tr>
<td>have_bcs_moist (new since V3.5.1)</td>
<td>.false.</td>
<td>do not use microphysics variables in boundary file in model run after ndown (default)</td>
</tr>
<tr>
<td>have_bcs_scalar (new since V3.5.1)</td>
<td>.false.</td>
<td>do not use scalar variables in boundary file in model run after ndown (default)</td>
</tr>
<tr>
<td>have_bcs_scalar (new since V3.5.1)</td>
<td>.true.</td>
<td>use scalar variables in boundary file</td>
</tr>
<tr>
<td>&amp;namelist_quilt</td>
<td>options for asynchronized I/O for MPI applications</td>
<td></td>
</tr>
<tr>
<td>nio_tasks_per_group</td>
<td>0</td>
<td>(default) no quilting</td>
</tr>
<tr>
<td>nio_groups</td>
<td>1</td>
<td>default; may be set to higher value for nesting IO or history and restart IO</td>
</tr>
<tr>
<td>&amp;grib2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>background_proc_id</td>
<td>255</td>
<td>(default); background generating process identifier, typically defined by the originating center to identify the background data that was used in creating the data; this is octet 13 of Section 4 in the grib2 message</td>
</tr>
<tr>
<td>forecast_proc_id</td>
<td>255</td>
<td>(default) analysis or generating forecast process identifier, typically defined by the originating center to identify the forecast process that was used to generate the data; this is octet 14 of Section 4 in the grib2 message</td>
</tr>
<tr>
<td>production_status</td>
<td>255</td>
<td>(default) production status of processed data in the grib2 message; see Code Table 1.3 of the grib2 manual; this is octect 20 of Section 1 in the grib2 record.</td>
</tr>
<tr>
<td>compression</td>
<td>40</td>
<td>(default) for jpeg2000</td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>PNG</td>
</tr>
<tr>
<td>&amp;dfi_control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>dfi_opt</td>
<td>0</td>
<td>(default) no digital filter initialization</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>digital filter launch (DFL)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>diabatic DFI (DDFI)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>(recommended) twice DFI (TDFI)</td>
</tr>
<tr>
<td>dfi_nfilter</td>
<td>0</td>
<td>uniform filter</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Lanczos filter</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hamming filter</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Blackman filter</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Kaiser filter</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Potter filter</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Dolph window filter</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>(default; recommended) Dolph filter</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>recursive high-order filter</td>
</tr>
<tr>
<td>dfi_write_filtered_input</td>
<td>.true.</td>
<td>whether to write wrfinput file with filtered model state before beginning forecast</td>
</tr>
<tr>
<td>dfi_write_dfi_history</td>
<td>.false.</td>
<td>whether to write wrfout files during filtering integration</td>
</tr>
<tr>
<td>dfi_cutoff_seconds</td>
<td>3600</td>
<td>cutoff period (s) for the filter; should not be longer than the filter window</td>
</tr>
<tr>
<td>dfi_time_dim</td>
<td>1000</td>
<td>maximum number of time steps for filtering period; this value can be larger than necessary</td>
</tr>
<tr>
<td>dfi_bckstop_year</td>
<td>2001</td>
<td>4-digit year of stop time for backward DFI integration</td>
</tr>
<tr>
<td>dfi_bckstop_month</td>
<td>06</td>
<td>2-digit month of stop time for backward DFI integration</td>
</tr>
<tr>
<td>dfi_bckstop_day</td>
<td>11</td>
<td>2-digit day of stop time for backward DFI integration</td>
</tr>
<tr>
<td>dfi_bckstop_hour</td>
<td>11</td>
<td>2-digit hour of stop time for backward DFI integration</td>
</tr>
<tr>
<td>dfi_bckstop_minute</td>
<td>00</td>
<td>2-digit minute of stop time for backward DFI integration</td>
</tr>
<tr>
<td>dfi_bckstop_second</td>
<td>00</td>
<td>2-digit second of stop time for backward DFI integration</td>
</tr>
</tbody>
</table>

*for a model that starts from 2001061112, the below setup specifies 1 hour backward integration*

*for a model that starts at*
2001061112, the below setup specifies 30 minutes of forward integration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfi_fwdstop_year</td>
<td>2001</td>
<td>4-digit year of stop time for forward DFI integration</td>
</tr>
<tr>
<td>dfi_fwdstop_month</td>
<td>06</td>
<td>2-digit month of stop time for forward DFI integration</td>
</tr>
<tr>
<td>dfi_fwdstop_day</td>
<td>11</td>
<td>2-digit day of stop time for forward DFI integration</td>
</tr>
<tr>
<td>dfi_fwdstop_hour</td>
<td>12</td>
<td>2-digit hour of stop time for forward DFI integration</td>
</tr>
<tr>
<td>dfi_fwdstop_minute</td>
<td>30</td>
<td>2-digit minute of stop time for forward DFI integration</td>
</tr>
<tr>
<td>dfi_fwdstop_second</td>
<td>00</td>
<td>2-digit second of stop time for forward DFI integration</td>
</tr>
<tr>
<td>dfi_radar</td>
<td>0</td>
<td>DFI radar data assimilation switch</td>
</tr>
</tbody>
</table>

&scm

for the single-column model (SCM) option only

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scm_force</td>
<td>0</td>
<td>(default) single column forcing turned off</td>
</tr>
<tr>
<td>scm_force</td>
<td>1</td>
<td>single column forcing on</td>
</tr>
<tr>
<td>scm_force_dx</td>
<td>4000.</td>
<td>DX for SCM forcing (m)</td>
</tr>
<tr>
<td>num_force_layers</td>
<td>8</td>
<td>number of SCM input forcing layers</td>
</tr>
<tr>
<td>scm_lu_index</td>
<td>2</td>
<td>SCM land use category (2 = dryland, cropland, and pasture; others can be found in the LANDUSE.TBL)</td>
</tr>
<tr>
<td>scm_isltyp</td>
<td>4</td>
<td>SCM soil category (4 = silt loam; others can be found in the SOILPARM.TBL)</td>
</tr>
<tr>
<td>scm_vegfra</td>
<td>0.5</td>
<td>SCM vegetation fraction</td>
</tr>
<tr>
<td>scm_canwat</td>
<td>0.0</td>
<td>SCM canopy water (kg m^-2)</td>
</tr>
<tr>
<td>scm_lat</td>
<td>36.605</td>
<td>SCM latitude</td>
</tr>
<tr>
<td>scm_lon</td>
<td>-97.485</td>
<td>SCM longitude</td>
</tr>
<tr>
<td>scm_th_adv</td>
<td>.true.</td>
<td>turn on theta advection in SCM</td>
</tr>
<tr>
<td>scm_wind_adv</td>
<td>.true.</td>
<td>turn on wind advection in SCM</td>
</tr>
<tr>
<td>scm_qv_adv</td>
<td>.true.</td>
<td>turn on moisture advection in SCM</td>
</tr>
<tr>
<td>scm_vert_adv</td>
<td>.true.</td>
<td>turn on vertical advection in SCM</td>
</tr>
<tr>
<td>scm ql_adv</td>
<td>.true.</td>
<td>turn on liquid advection in SCM</td>
</tr>
<tr>
<td>num_force_soil_layers</td>
<td>5</td>
<td>number of SCM soil forcing layers</td>
</tr>
<tr>
<td>scm_soilt_force</td>
<td>.true.</td>
<td>turn on soil temperature forcing in SCM (default is .false. = off)</td>
</tr>
</tbody>
</table>
### scm_soilq_force
- **Value:** .true.
- **Description:** Turn on soil moisture forcing in SCM (default is .false. = off)

### scm_force_th_largescale
- **Value:** .true.
- **Description:** Turn on large-scale theta forcing in SCM (default is .false. = off)

### scm_force_qv_largescale
- **Value:** .true.
- **Description:** Turn on large-scale qv forcing in SCM (default is .false. = off)

### scm_force_ql_largescale
- **Value:** .true.
- **Description:** Turn on large-scale ql forcing in SCM (default is .false. = off)

### scm_force_wind_largescale
- **Value:** .true.
- **Description:** Turn on large-scale wind forcing in SCM (default is .false. = off)

### &tc (controls for tc_em.exe only)
- **insert_bogus_storm**
  - **Value:** .false.
  - **Description:** T/F for inserting a bogus tropical storm
- **remove_storm**
  - **Value:** .false.
  - **Description:** T/F for only removing the original TC
- **num_storm**
  - **Value:** 1
  - **Description:** Number of bogus TC
- **latc_loc**
  - **Value:** -999.
  - **Description:** Center latitude of the bogus TC
- **lonc_loc**
  - **Value:** -999.
  - **Description:** Center longitude of the bogus TC
- **vmax_meters_per_second (max_dom)**
  - **Value:** -999.
  - **Description:** Wind max of bogus storm (m s⁻¹)
- **rmmax**
  - **Value:** -999.
  - **Description:** Maximum radius outward from storm center of bogus TC
- **vmax_ratio (max_dom)**
  - **Value:** -999.
  - **Description:** Ratio for representative maximum winds, 0.75 for 45 km grid, and 0.9 for 15 km grid
- **rankine_lid**
  - **Value:** -999.
  - **Description:** Top pressure limit for the TC bogus scheme

### &diags
*output fields on pressure levels*
*Also need to set auxhist23_outname="wrfpress_d<domain>_<date>"*
- **io_form_auxhist23**
  - **Value:** 2
- **auxhist23_interval**
  - **Value:** 180, 180,
- **frame_per_auxhist23**
  - **Value:** 100, 100,
- **p_lev_diags**
  - **Value:** 0
  - **Description:** 0/1 whether to output pressure level diagnostics
- **num_press_levels**
  - **Value:** 4
  - **Description:** Number of pressure levels
- **press_levels (max_plevs)**
  - **Value:** 0
  - **Description:** Pressure levels in Pa
- **use_tot_or_hyd_p**
  - **Value:** 2
  - **Description:** 1: use total pressure 2: use hydrostatic pressure
- **z_lev_diags**
  - **Value:** 0
  - **Description:** Switch to vertically interpolate diagnostics to z-levels; (default is off)
- **num_z_levels**
  - **Value:** 0
  - **Description:** Number of height levels to
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z_levels</td>
<td>0</td>
<td>list of height values (m) to interpolate data to; positive numbers are for height above mean sea level (i.e., a flight level), negative numbers are for level above ground</td>
</tr>
<tr>
<td>p_lev_missing</td>
<td>-999.</td>
<td>Missing value below ground</td>
</tr>
<tr>
<td>afwa_diag_opt</td>
<td>0</td>
<td>(default) AFWA diagnostic option (1 = on)</td>
</tr>
<tr>
<td>afwa_ptype_opt</td>
<td>0</td>
<td>(default) precipitation type option (1 = on)</td>
</tr>
<tr>
<td>afwa_vil_opt</td>
<td>0</td>
<td>(default) vertical integral liquid option (1 = on)</td>
</tr>
<tr>
<td>afwa_radar_opt</td>
<td>0</td>
<td>(default) radar option (1 = on)</td>
</tr>
<tr>
<td>afwa_severe_opt</td>
<td>0</td>
<td>(default) severe weather option (1 = on)</td>
</tr>
<tr>
<td>afwa_icing_opt</td>
<td>0</td>
<td>(default) icing option (1 = on)</td>
</tr>
<tr>
<td>afwa_vis_opt</td>
<td>0</td>
<td>(default) visibility option (1 = on)</td>
</tr>
<tr>
<td>afwa_cloud_opt</td>
<td>0</td>
<td>(default) cloud option (1 = on)</td>
</tr>
<tr>
<td>afwa_therm_opt</td>
<td>0</td>
<td>thermal indices option (default is 0=off)</td>
</tr>
<tr>
<td>afwa_turb_opt</td>
<td>0</td>
<td>turbulence option (default is 0=off)</td>
</tr>
<tr>
<td>afwa_buoy_opt</td>
<td>0</td>
<td>buoyancy option (default is 0=off)</td>
</tr>
<tr>
<td>afwa_hailcast_opt</td>
<td>0</td>
<td>hailcast option (default is 0=off)</td>
</tr>
<tr>
<td>afwa_ptype_ccn_tmp</td>
<td>264.15</td>
<td>(default) CCN temperature for precipitation type calculation</td>
</tr>
<tr>
<td>afwa_ptype_tot_melt</td>
<td>50</td>
<td>(default) total melting energy for precipitation type calculation</td>
</tr>
<tr>
<td>progn</td>
<td>0</td>
<td>(default) switch to use mix-activate scheme (only for Morrison, WDM6, WDM5, and NSSL_2MOMCCN/NSSL_2MOM)</td>
</tr>
</tbody>
</table>
WRF Output Fields

List of Fields

The following is an edited output list from the netCDF command `ncdump -h`. Note that valid output fields will depend on the model options used. If the fields have zero values, then the fields are not computed by the model options selected.

```
ncdump -h wrfout_d<domain>_<date>
```

```
netcdf wrfout_d01_2000-01-24_12:00:00

dimensions:
  Time = UNLIMITED ; // (1 currently)
  DateStrLen = 19 ;
  west_east = 73 ;
  south_north = 60 ;
  bottom_top = 29 ;
  bottom_top_stag = 30 ;
  soil_layers_stag = 4 ;
  west_east_stag = 74 ;
  south_north_stag = 61 ;

variables:
  char Times(Time, DateStrLen) ;
  float LU_INDEX(Time, south_north, west_east) ;
    LU_INDEX:description = "LAND USE CATEGORY" ;
    LU_INDEX:units = "" ;
  float ZNU(Time, bottom_top) ;
    ZNU:description = "eta values on half (mass) levels" ;
    ZNU:units = "" ;
  float ZNW(Time, bottom_top_stag) ;
    ZNW:description = "eta values on full (w) levels" ;
    ZNW:units = "" ;
  float ZS(Time, soil_layers_stag) ;
    ZS:description = "DEPTHS OF CENTERS OF SOIL LAYERS" ;
    ZS:units = "m" ;
  float DZS(Time, soil_layers_stag) ;
    DZS:description = "THICKNESSES OF SOIL LAYERS" ;
    DZS:units = "m" ;
  float U(Time, bottom_top, south_north, west_east_stag) ;
    U:units = "m s-1" ;
  float V(Time, bottom_top, south_north_stag, west_east) ;
    V:units = "m s-1" ;
  float W(Time, bottom_top_stag, south_north, west_east) ;
    W:units = "m s-1" ;
  float PH(Time, bottom_top_stag, south_north, west_east) ;
    PH:units = "perturbation geopotential" ;
  float PHB(Time, bottom_top_stag, south_north, west_east) ;
    PHB:units = "base-state geopotential" ;
  float T(Time, bottom_top, south_north, west_east) ;
    T:units = "perturbation potential temperature (theta-t0)" ;
    T:units = "K" ;
  float MU(Time, south_north, west_east) ;
    MU:description = "perturbation dry air mass in column" ;
```
MU: units = "Pa";
float MUB(Time, south_north, west_east);
MUB: description = "base state dry air mass in column";
MUB: units = "Pa";
float NEST_POS(Time, south_north, west_east);
NEST_POS: description = "-";
NEST_POS: units = "-";
float P(Time, bottom_top, south_north, west_east);
P: description = "perturbation pressure";
P: units = "Pa";
float PB(Time, bottom_top, south_north, west_east);
PB: description = "BASE STATE PRESSURE";
PB: units = "Pa";
float FNM(Time, bottom_top);
FNM: description = "upper weight for vertical stretching";
FNM: units = "";
float FNP(Time, bottom_top);
FNP: description = "lower weight for vertical stretching";
FNP: units = "";
float RDNW(Time, bottom_top);
RDNW: description = "inverse d(eta) values between full (w) levels";
RDNW: units = "";
float RDN(Time, bottom_top);
RDN: description = "inverse d(eta) values between half (mass) levels";
RDN: units = "";
float DNW(Time, bottom_top);
DNW: description = "d(eta) values between full (w) levels";
DNW: units = "";
float DN(Time, bottom_top);
DN: description = "d(eta) values between half (mass) levels";
DN: units = "";
float CFN(Time);
CFN: description = "extrapolation constant";
CFN: units = "";
float CFN1(Time);
CFN1: description = "extrapolation constant";
CFN1: units = "";
float P_HYD(Time, bottom_top, south_north, west_east);
P_HYD: description = "hydrostatic pressure";
P_HYD: units = "Pa";
float Q2(Time, south_north, west_east);
Q2: description = "QV at 2 M";
Q2: units = "kg kg\(^{-1}\)";
float T2(Time, south_north, west_east);
T2: description = "TEMP at 2 M";
T2: units = "K";
float TH2(Time, south_north, west_east);
TH2: description = "POT TEMP at 2 M";
TH2: units = "K";
float PSFC(Time, south_north, west_east);
PSFC: description = "SFC PRESSURE";
PSFC: units = "Pa";
float U10(Time, south_north, west_east);
U10: description = "U at 10 M";
U10: units = "m s\(^{-1}\)";
float V10(Time, south_north, west_east);
V10: description = "V at 10 M";
V10: units = "m s\(^{-1}\)";
float RDX(Time);
RDX: description = "INVERSE X GRID LENGTH";
RDX: units = "";
float RDY(Time);
RDY: description = "INVERSE Y GRID LENGTH";
RDY: units = "";
float RESM(Time);
RESM:description = "TIME WEIGHT CONSTANT FOR SMALL STEPS" ;
RESM:units = "" ;
float ZETATOP(Time) ;
ZETATOP:description = "ZETA AT MODEL TOP" ;
ZETATOP:units = "" ;
float CF1(Time) ;
CF1:description = "2nd order extrapolation constant" ;
CF1:units = "" ;
float CF2(Time) ;
CF2:description = "2nd order extrapolation constant" ;
CF2:units = "" ;
float CF3(Time) ;
CF3:description = "2nd order extrapolation constant" ;
CF3:units = "" ;
int ITIMESTEP(Time) ;
ITIMESTEP:description = "" ;
ITIMESTEP:units = "" ;
float XTIME(Time) ;
XTIME:description = "minutes since simulation start" ;
XTIME:units = "" ;
float QVAPOR(Time, bottom_top, south_north, west_east) ;
QVAPOR:description = "Water vapor mixing ratio" ;
QVAPOR:units = "kg kg^{-1}" ;
float QCLOUD(Time, bottom_top, south_north, west_east) ;
QCLOUD:description = "Cloud water mixing ratio" ;
QCLOUD:units = "kg kg^{-1}" ;
float QRAIN(Time, bottom_top, south_north, west_east) ;
QRAIN:description = "Rain water mixing ratio" ;
QRAIN:units = "kg kg^{-1}" ;
float LANDMASK(Time, south_north, west_east) ;
LANDMASK:description = "LAND MASK (1 FOR LAND, 0 FOR WATER)" ;
LANDMASK:units = "" ;
float TSLB(Time, soil_layers_stag, south_north, west_east) ;
TSLB:description = "SOIL TEMPERATURE" ;
TSLB:units = "K" ;
float SMOIS(Time, soil_layers_stag, south_north, west_east) ;
SMOIS:description = "SOIL MOISTURE" ;
SMOIS:units = "m^3 m^{-3}" ;
float SH2O(Time, soil_layers_stag, south_north, west_east) ;
SH2O:description = "SOIL LIQUID WATER" ;
SH2O:units = "m^3 m^{-3}" ;
float SEAICE(Time, south_north, west_east) ;
SEAICE:description = "SEA ICE FLAG" ;
SEAICE:units = "" ;
float XICEM(Time, south_north, west_east) ;
XICEM:description = "SEA ICE FLAG (PREVIOUS STEP)" ;
XICEM:units = "" ;
float SFROFF(Time, south_north, west_east) ;
SFROFF:description = "SURFACE RUNOFF" ;
SFROFF:units = "mm" ;
float UDROFF(Time, south_north, west_east) ;
UDROFF:description = "UNDERGROUND RUNOFF" ;
UDROFF:units = "mm" ;
int IVGTYP(Time, south_north, west_east) ;
IVGTYP:description = "DOMINANT VEGETATION CATEGORY" ;
IVGTYP:units = "" ;
int ISLTYP(Time, south_north, west_east) ;
ISLTYP:description = "DOMINANT SOIL CATEGORY" ;
ISLTYP:units = "" ;
float VEGFRA(Time, south_north, west_east) ;
VEGFRA:description = "VEGETATION FRACTION" ;
VEGFRA:units = "" ;
float GRDFLX(Time, south_north, west_east) ;
GRDFLX:description = "GROUND HEAT FLUX" ;
GRDFLX:units = "W m^{-2}" ;
float SNOW(Time, south_north, west_east) ;
   SNOW:description = "SNOW WATER EQUIVALENT" ;
   SNOW:units = "kg m^-2" ;
float SNOWH(Time, south_north, west_east) ;
   SNOWH:description = "PHYSICAL SNOW DEPTH" ;
   SNOWH:units = "m" ;
float RHOSN(Time, south_north, west_east) ;
   RHOSN:description = " SNOW DENSITY" ;
   RHOSN:units = "kg m^-3" ;
float CANWAT(Time, south_north, west_east) ;
   CANWAT:description = "CANOPY WATER" ;
   CANWAT:units = "kg m^-2" ;
float SST(Time, south_north, west_east) ;
   SST:description = "SEA SURFACE TEMPERATURE" ;
   SST:units = "K" ;
float SSTSK(Time, south_north, west_east) ;
   SSTSK:description = "SKIN SEA SURFACE TEMPERATURE" ;
   SSTSK:units = "K" ;
float MAPFAC_M(Time, south_north, west_east) ;
   MAPFAC_M:description = "Map scale factor on mass grid" ;
   MAPFAC_M:units = "" ;
float MAPFAC_U(Time, south_north, west_east_stag) ;
   MAPFAC_U:description = "Map scale factor on u-grid" ;
   MAPFAC_U:units = "" ;
float MAPFAC_V(Time, south_north_stag, west_east) ;
   MAPFAC_V:description = "Map scale factor on v-grid" ;
   MAPFAC_V:units = "" ;
float MAPFAC_MX(Time, south_north, west_east) ;
   MAPFAC_MX:description = "Map scale factor on mass grid, x direction" ;
   MAPFAC_MX:units = "" ;
float MAPFAC_MY(Time, south_north, west_east) ;
   MAPFAC_MY:description = "Map scale factor on mass grid, y direction" ;
   MAPFAC_MY:units = "" ;
float MAPFAC_UX(Time, south_north, west_east_stag) ;
   MAPFAC_UX:description = "Map scale factor on u-grid, x direction" ;
   MAPFAC_UX:units = "" ;
float MAPFAC_UY(Time, south_north, west_east_stag) ;
   MAPFAC_UY:description = "Map scale factor on u-grid, y direction" ;
   MAPFAC_UY:units = "" ;
float MAPFAC_VX(Time, south_north_stag, west_east) ;
   MAPFAC_VX:description = "Map scale factor on v-grid, x direction" ;
   MAPFAC_VX:units = "" ;
float MAPFAC_VY(Time, south_north_stag, west_east) ;
   MAPFAC_VY:description = "Map scale factor on v-grid, y direction" ;
   MAPFAC_VY:units = "" ;
float F(Time, south_north, west_east) ;
   F:description = "Coriolis sine latitude term" ;
   F:units = "s^-1" ;
float E(Time, south_north, west_east) ;
   E:description = "Coriolis cosine latitude term" ;
   E:units = "s^-1" ;
float SINALPHA(Time, south_north, west_east) ;
   SINALPHA:description = "Local sine of map rotation" ;
   SINALPHA:units = "" ;
float COSALPHA(Time, south_north, west_east) ;
   COSALPHA:description = "Local cosine of map rotation" ;
   COSALPHA:units = "" ;
float HGT(Time, south_north, west_east) ;
   HGT:description = "Terrain Height" ;
   HGT:units = "m" ;
float HGT_SHAD(Time, south_north, west_east) ;
   HGT_SHAD:description = "Height of orographic shadow" ;
HGT_SHAD:units = "m";
float TSK(Time, south_north, west_east);
  TSK:description = "SURFACE SKIN TEMPERATURE";
  TSK:units = "°K";
float P_TOP(Time);
  P_TOP:description = "PRESSURE TOP OF THE MODEL";
  P_TOP:units = "Pa";
float T00(Time);
  T00:description = "BASE STATE TEMPERATURE";
  T00:units = "°K";
float P00(Time);
  P00:description = "BASE STATE PRESSURE";
  P00:units = "Pa";
float TLP(Time);
  TLP:description = "BASE STATE LAPSE RATE";
  TLP:units = "";
float TISO(Time);
  TISO:description = "TEMP AT WHICH THE BASE T TURNS CONST";
  TISO:units = "°K";
float MAX_MSTFX(Time);
  MAX_MSTFX:description = "Max map factor in domain";
  MAX_MSTFX:units = "";
float MAX_MSTFY(Time);
  MAX_MSTFY:description = "Max map factor in domain";
  MAX_MSTFY:units = "";
float RAINC(Time, south_north, west_east);
  RAINC:description = "ACCUMULATED TOTAL CUMULUS PRECIPITATION";
  RAINC:units = "mm";
float RAINSH(Time, south_north, west_east);
  RAINSH:description = "ACCUMULATED SHALLOW CUMULUS PRECIPITATION";
  RAINSH:units = "mm";
float RAINNC(Time, south_north, west_east);
  RAINNC:description = "ACCUMULATED TOTAL GRID SCALE PRECIPITATION";
  RAINNC:units = "mm";
float PRATEC(Time, south_north, west_east);
  PRATEC:description = "PRECIP RATE FROM CUMULUS SCHEME";
  PRATEC:units = "mm s\(^{-1}\)";
float RAINCV(Time, south_north, west_east);
  RAINCV:description = "TIME-STEP CUMULUS PRECIPITATION";
  RAINCV:units = "mm";
float SNOWNC(Time, south_north, west_east);
  SNOWNC:description = "ACCUMULATED TOTAL GRID SCALE SNOW AND ICE";
  SNOWNC:units = "mm";
float GRAUPELNC(Time, south_north, west_east);
  GRAUPELNC:description = "ACCUMULATED TOTAL GRID SCALE GRAUPEL";
  GRAUPELNC:units = "mm";
float SWDOWN(Time, south_north, west_east);
  SWDOWN:description = "DOWNWARD SHORT WAVE FLUX AT GROUND SURFACE";
  SWDOWN:units = "W m\(^{-2}\)";
float GLW(Time, south_north, west_east);
  GLW:description = "DOWNWARD LONG WAVE FLUX AT GROUND SURFACE";
  GLW:units = "W m\(^{-2}\)";
float SWINORM(Time, south_north, west_east);
  SWINORM:description = "NORMAL SHORT WAVE FLUX AT GROUND SURFACE";
  SWINORM:units = "W m\(^{-2}\)";
float OLR(Time, south_north, west_east);
  OLR:description = "TOA OUTGOING LONG WAVE";
  OLR:units = "W m\(^{-2}\)";
float XLAT(Time, south_north, west_east);
  XLAT:description = "LATITUDE, SOUTH IS NEGATIVE";
  XLAT:units = "degree_north";
float XLONG(Time, south_north, west_east);
  XLONG:description = "LONGITUDE, WEST IS NEGATIVE";
  XLONG:units = "degree_east";
float XLAT_U(Time, south_north, west_east_stag);
Model

```
XLAT_U: description = "LATITUDE, SOUTH IS NEGATIVE";
XLAT_U: units = "degree_north";
float XLAT_U(Time, south_north, west_east_stag);
XLONG_U: description = "LONGITUDE, WEST IS NEGATIVE";
XLONG_U: units = "degree_east";
float XLONG_U(Time, south_north_stag, west_east);
XLAT_V: description = "LATITUDE, SOUTH IS NEGATIVE";
XLAT_V: units = "degree_north";
float XLAT_V(Time, south_north_stag, west_east);
XLONG_V: description = "LONGITUDE, WEST IS NEGATIVE";
XLONG_V: units = "degree_east";
float ALBEDO(Time, south_north, west_east);
ALBEDO: description = "ALBEDO";
ALBEDO: units = "";
float ALBBCK(Time, south_north, west_east);
ALBBCK: description = "BACKGROUND ALBEDO";
ALBBCK: units = "";
float EMISS(Time, south_north, west_east);
EMISS: description = "SURFACE EMISSIVITY";
EMISS: units = "";
float NOAHRES(Time, south_north, west_east);
NOAHRES: description = "RESIDUAL OF THE NOAH SURFACE ENERGY BUDGET";
NOAHRES: units = "W m(-2)";
float TMN(Time, south_north, west_east);
TMN: description = "SOIL TEMPERATURE AT LOWER BOUNDARY";
TMN: units = "K";
float XLAND(Time, south_north, west_east);
XLAND: description = "LAND MASK (1 FOR LAND, 2 FOR WATER)"
XLAND: units = "";
float ZNT(Time, south_north, west_east);
ZNT: description = "TIME-VARYING ROUGHNESS LENGTH"
ZNT: units = "m";
float UST(Time, south_north, west_east);
UST: description = "U* IN SIMILARITY THEORY"
UST: units = "m s(-1)";
float PBLH(Time, south_north, west_east);
PBLH: description = "PBL HEIGHT"
PBLH: units = "m";
float HFX(Time, south_north, west_east);
HFX: description = "UPWARD HEAT FLUX AT THE SURFACE"
HFX: units = "W m(-2)";
float QFX(Time, south_north, west_east);
QFX: description = "UPWARD MOISTURE FLUX AT THE SURFACE"
QFX: units = "kg m(-2) s(-1)";
float LH(Time, south_north, west_east);
LH: description = "LATENT HEAT FLUX AT THE SURFACE"
LH: units = "W m(-2)";
float SNOWC(Time, south_north, west_east);
SNOWC: description = "FLAG INDICATING SNOW COVERAGE (1 FOR SNOW COVER)"
SNOWC: units = ""
int SAVE_TOPO_FROM_REAL(Time);
SAVE_TOPO_FROM_REAL: description = "1=original topo from real/0=topo modified by WRF"
SAVE_TOPO_FROM_REAL: units = "flag"
```

List of Global Attributes

// global attributes:

```
:TITLE = " OUTPUT FROM WRF V3.6 MODEL";
:START_DATE = "2000-01-24_12:00:00";
:SIMULATION_START_DATE = "2000-01-24_12:00:00"
```
WEST-EAST_GRID_DIMENSION = 74;
SOUTH-NORTH_GRID_DIMENSION = 61;
BOTTOM-TOP_GRID_DIMENSION = 28;
DX = 30000.f;
DY = 30000.f;
STOCH_FORCE_OPT = 0;
GRIDTYPE = "C";
DIFF_OPT = 1;
KM_OPT = 4;
DAMP_OPT = 0;
DAMPCOEF = 0.2f;
KHDIF = 0.f;
KVDIF = 0.f;
MP_PHYSICS = 3;
RA_LW_PHYSICS = 1;
RA_SW_PHYSICS = 1;
SF_SFCLAY_PHYSICS = 1;
SF_SURFACE_PHYSICS = 2;
BL_PBL_PHYSICS = 1;
CU_PHYSICS = 1;
SF_LAKE_PHYSICS = 0;
SURFACE_INPUT_SOURCE = 1;
SST_UPDATE = 0;
GRID_FDDA = 0;
GFDDA_INTERVAL_M = 0;
GFDDA_END_H = 0;
GRID_SFDDA = 0;
SGFDDA_INTERVAL_M = 0;
SGFDDA_END_H = 0;
HYPSOMETRIC_OPT = 2;
SF_URBAN_PHYSICS = 0;
SHCU_PHYSICS = 0;
MFSHCONV = 0;
FEEDBACK = 1;
SMOOTH_OPTION = 0;
SWRAD_SCAT = 1.f;
W_DAMPING = 0;
DT = 180.f;
RADT = 30.f;
BLDT = 0.f;
CUDT = 5.f;
AER_OPT = 0;
SWINT_OPT = 0;
AER_TYPE = 1;
AER_AOD550_OPT = 1;
AER_ANGEXP_OPT = 1;
AER_SSA_OPT = 1;
AER_ASY_OPT = 1;
AER_AOD550_VAL = 0.12f;
AER_ANGEXP_VAL = 1.3f;
AER_SSA_VAL = 0.f;
AER_ASY_VAL = 0.f;
MOIST_ADV_OPT = 1;
SCALAR_ADV_OPT = 1;
TKE_ADV_OPT = 1;
DIFF_6TH_OPT = 0;
DIFF_6TH_FACTOR = 0.12f;
OBS_NUDGE_OPT = 0;
BUCKET_MM = -1.f;
BUCKET_J = -1.f;
PREC_ACC_DT = 0.f;
SF_OCEAN_PHYSICS = 0;
ISFTCFLX = 0;
ISHALLOW = 0;
ISFFLX = 1;
The WRF model outputs the state variables defined in the Registry file, and these state variables are used in the model's prognostic equations. Some of these variables are perturbation fields; therefore the following definitions for reconstructing meteorological variables are necessary:

- **total geopotential** \( \text{PH + PHB} \)
- **total geopotential height in m** \( \frac{\text{PH + PHB}}{9.81} \)
- **total potential temperature in K** \( \text{T + 300} \)
- **total pressure in mb** \( \frac{(\text{P + PB})}{0.01} \)

**Special WRF Output Variables**

The WRF model outputs the state variables defined in the Registry file, and these state variables are used in the model's prognostic equations. Some of these variables are perturbation fields; therefore the following definitions for reconstructing meteorological variables are necessary:
wind components, grid relative \( U, V \)
surface pressure in Pa \( psfc \)
surface winds, grid relative \( U10, V10 \) (valid at mass points)
surface temperature and mixing ratio \( T2, Q2 \)

The definitions for map projection options:

\[
\text{map_proj} =
\begin{align*}
1: & \text{ Lambert Conformal} \\
2: & \text{ Polar Stereographic} \\
3: & \text{ Mercator} \\
6: & \text{ latitude and longitude (including global)}
\end{align*}
\]
Chapter 6: WRF Data Assimilation

Table of Contents

- Introduction
- Installing WRFDA for 3D-Var Run
- Installing WRFPLUS and WRFDA for 4D-Var Run
- Running Observation Preprocessor (OBSPROC)
- Running WRFDA
- Radiance Data Assimilation in WRFDA
- Radar Data Assimilation in WRFDA
- Precipitation Data Assimilation in WRFDA 4D-Var
- Updating WRF Boundary Conditions
- Background Error and running GEN_BE
- Additional WRFDA Exercises
- WRFDA Diagnostics
- Generating ensembles with RANDOMCV
- Hybrid Data Assimilation
- ETKF Data Assimilation
- Description of Namelist Variables

Introduction

Data assimilation is the technique by which observations are combined with an NWP product (the first guess or background forecast) and their respective error statistics to provide an improved estimate (the analysis) of the atmospheric (or oceanic, Jovian, etc.) state. Variational (Var) data assimilation achieves this through the iterative minimization of a prescribed cost (or penalty) function. Differences between the analysis and observations/first guess are penalized (damped) according to their perceived error. The difference between three-dimensional (3D-Var) and four-dimensional (4D-Var) data assimilation is the use of a numerical forecast model in the latter.

The MMM Laboratory of NCAR supports a unified (global/regional, multi-model, 3/4D-Var) model-space data assimilation system (WRFDA) for use by the NCAR staff and collaborators, and is also freely available to the general community, together with further documentation, test results, plans etc., from the WRFDA web-page (http://www2.mmm.ucar.edu/wrf/users/wrfda/index.html).

Various components of the WRFDA system are shown in blue in the sketch below, together with their relationship with the rest of the WRF system.
In this chapter, you will learn how to install and run the various components of the WRFDA system. For training purposes, you are supplied with a test case, including the following input data:

- observation files,
- a netCDF background file (WPS/real.exe output, the first guess of the analysis)
- background error statistics (estimate of errors in the background file).

This tutorial dataset can be downloaded from the WRFDA Users Page (http://www2.mmm.ucar.edu/wrf/users/wrfda/download/testdata.html), and will be described later in more detail. In your own work, however, you will have to create all these input files yourself. See the section "Running Observation Preprocessor" for creating your observation files. See the section "Background Error and running GEN BE" for generating your background error statistics file, if you want to use cv_options=5, 6, or 7.
Before using your own data, we suggest that you start by running through the WRFDA-related programs using the supplied test case. This serves two purposes: First, you can learn how to run the programs with data we have tested ourselves, and second you can test whether your computer is capable of running the entire data assimilation system. After you have done the tutorial, you can try running other, more computationally intensive case studies, and experimenting with some of the many namelist variables.

**WARNING:** It is impossible to test every permutation of computer, compiler, number of processors, case, namelist option, etc. for every WRFDA release. The namelist options that are supported are indicated in the “WRFDA/var/README.namelist”, and these are the default options.

Hopefully, our test cases will prepare you for the variety of ways in which you may wish to run your own WRFDA experiments. Please inform us about your experiences.

As a professional courtesy, we request that you include the following references in any publication that uses any component of the community WRFDA system:


Running WRFDA requires a Fortran 90 compiler. The WRFDA system can be compiled on the following platforms: Linux (ifort, gfortran, pgf90), Macintosh (gfortran, ifort), IBM (xlIf), and SGI Altix (ifort). Please let us know if this does not meet your requirements, and we will attempt to add other machines to our list of supported architectures, as resources allow. Although we are interested in hearing about your experiences in modifying compiler options, we do not recommend making changes to the configure file used to compile WRFDA.
Installing WRFDA for 3D-Var Run

a. Obtaining WRFDA Source Code

Users can download the WRFDA source code from [http://www2.mmm.ucar.edu/wrf/users/wrfda/download/get_source.html](http://www2.mmm.ucar.edu/wrf/users/wrfda/download/get_source.html).

*Note: Although the WRFDA package also contains the WRF source code, they cannot be built together. WRF should be downloaded and compiled separately.*

After the tar file is unzipped (`gunzip WRFDAV3.8.1.tar.gz`) and untarred (`tar -xf WRFDAV3.8.1.tar`), the directory `WRFDA` should be created. This directory contains the WRFDA source, external libraries, and fixed files. The following is a list of the system components and content for each subdirectory:

<table>
<thead>
<tr>
<th>Directory Name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>var/da</td>
<td>WRFDA source code</td>
</tr>
<tr>
<td>var/run</td>
<td>Fixed input files required by WRFDA, such as background error covariance, radiance-related files, CRTM coefficients and VARBC.in</td>
</tr>
<tr>
<td>var/external</td>
<td>Libraries needed by WRFDA, includes CRTM, BUFR, LAPACK, BLAS</td>
</tr>
<tr>
<td>var/obsproc</td>
<td>OBSPROC source code, namelist, and observation error files</td>
</tr>
<tr>
<td>var/gen_be</td>
<td>Source code of gen_be, the utility to create background error statistics files</td>
</tr>
<tr>
<td>var/build</td>
<td>Builds all .exe files.</td>
</tr>
</tbody>
</table>

b. Compile WRFDA and Libraries

Some external libraries (e.g., LAPACK, BLAS, and NCEP BUFR) are included in the WRFDA tar file. To compile the WRFDA code, the only mandatory library is the netCDF library. You should set an environment variable `NETCDF` to point to the directory where your netCDF library is installed

```
> setenv NETCDF your_netcdf_path
```

The source code for BUFLIB 10.2.3 (with minor modifications) is included in the WRFDA tar file, and is compiled automatically. This library will be used for assimilating files in PREPBUFR and NCEP BUFR format.
Starting with WRFDA version 3.8, AMSR2 data can be assimilated in HDF5 format, which requires the use of HDF5 libraries. If you wish to make use of this capability, you should ensure that HDF5 libraries are installed on your system (or download and install them yourself; the source code is available from https://www.hdfgroup.org/HDF5/). To use HDF5 in WRFDA, you should set the environment variable “HDF5” to the parent path of your HDF5 build:

```
> setenv HDF5 your_hdf5_path
```

The HDF5 path should contain the directories “include” and “lib”.

For some platforms, you may have to also add the HDF5 “lib” directory to your environment variable LD_LIBRARY_PATH:

```
> setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:your_hdf5_path/lib
```

If satellite radiance data are to be used, a Radiative Transfer Model (RTM) is required. The current RTM versions that WRFDA supports are CRTM V2.1.3 and RTTOV V11.1–11.3.

The CRTM V2.1.3 source code is included in the WRFDA tar file, and is compiled automatically. No action is needed from the user.

If the user wishes to use RTTOV, download and install the RTTOV v11 library before compiling WRFDA. This library can be downloaded from http://nwpsaf.eu/deliverables/rtm/index.html. The RTTOV libraries must be compiled with the “emis_atlas” option in order to work with WRFDA; see the RTTOV “readme.txt” for instructions on how to do this. After compiling RTTOV (see the RTTOV documentation for detailed instructions), set the “RTTOV” environment variable to the path where the lib directory resides. For example, if the library files can be found in /usr/local/rttov11/gfortran/lib/librttov11.*.a, you should set RTTOV as:

```
> setenv RTTOV /usr/local/rttov11/gfortran
```

*Note: Make sure the required libraries were all compiled using the same compiler that will be used to build WRFDA,* since the libraries produced by one compiler may not be compatible with code compiled with another.

Assuming all required libraries are available and the WRFDA source code is ready, you can start to build WRFDA using the following steps:

Enter the WRFDA directory and run the configure script:

```
> cd WRFDA
> ./configure wrfda
```
A list of configuration options should appear. Each option combines an operating system, a compiler type, and a parallelism option. Since the configuration script doesn’t check which compilers are actually installed on your system, be sure to select only among the options that you have available to you. The available parallelism options are single-processor (serial), shared-memory parallel (smpar), distributed-memory parallel (dmpar), and distributed-memory with shared-memory parallel (sm+dm). **However, shared-memory (smpar and sm+dm) options are not supported as of WRFDA Version 3.8.1, so we do not recommend selecting any of these options.**

For example, on a Linux machine such as NCAR’s Yellowstone, the above steps will look similar to the following:

```bash
checking for perl5... no
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /glade/apps/opt/netcdf/4.3.0/gnu/4.8.2/
Will use HDF5 in dir: /glade/u/apps/opt/hdf5/1.8.12/gnu/4.8.2/
P HDF5 not set in environment. Will configure WRF for use without.
Will use 'time' to report timing information
$JASPERLIB or $JASPERINC not found in environment, configuring to build without grib2
I/O...

Please select from among the following Linux x86_64 options:

1. (serial) 2. (smpar) 3. (dmpar) 4. (dm+sm)  **PGI (pgf90/gcc)**  
5. (serial) 6. (smpar) 7. (dmpar) 8. (dm+sm)  **PGI (pgf90/pgcc): SGI MPT**  
17. (dm+sm)  **INTEL (ifort/icc): SGI MPT**  
22. (serial) 23. (smpar) 24. (dmpar) 25. (dm+sm)  **INTEL (ifort/icc): IBM POE**  
26. (serial) 27. (smpar) 28. (dmpar) 29. (dm+sm)  **PATHSCALE (pathf90/pathcc)**  
30. (serial) 31. (dmpar) 32. (dm+sm)  **GNU (gfortran/gcc)**  
36. (serial) 37. (smpar) 38. (dmpar) 39. (dm+sm)  **IBM (xlf90_r/cc_r)**  
40. (serial) 41. (smpar) 42. (dmpar) 43. (dm+sm)  **PGI (ftn/gcc): Cray XC CLE**  
44. (serial) 45. (smpar) 46. (dmpar) 47. (dm+sm)  **Cray CCE (ftn/cc): CrayXE and XC**  
48. (serial) 49. (smpar) 50. (dmpar) 51. (dm+sm)  **INTEL (ftn/icc): Cray XC**  
52. (serial) 53. (smpar) 54. (dmpar) 55. (dm+sm)  **PGI (pgf90/pgcc): -f90=pgf90**  
56. (serial) 57. (smpar) 58. (dmpar) 59. (dm+sm)  **PGI (pgf90/gcc): -f90=pgf90**  
60. (serial) 61. (smpar) 62. (dmpar) 63. (dm+sm)  **PGI (pgf90/pgcc): -f90=pgf90**  
64. (serial) 65. (smpar) 66. (dmpar) 67. (dm+sm)  **INTEL (ifort/icc): HSW/BDW**  
68. (serial) 69. (smpar) 70. (dmpar) 71. (dm+sm)  **INTEL (ifort/icc): KNL MIC**

Enter selection [1-71] : 34

Configuration successful!

... ...
After running the configuration script and choosing a compilation option, a configure.wrf file will be created. Because of the variety of ways that a computer can be configured, if the WRFDA build ultimately fails, there is a chance that minor modifications to the configure.wrf file may be needed.

To compile WRFDA, type

> ./compile all_wrfvar >& compile.out

Successful compilation will produce 44 executables: 43 of which are in the var/build directory and linked in the var/da directory, with the 44th, obsproc.exe, found in the var/obsproc/src directory. You can list these executables by issuing the command:

>ls -l var/build/*exe var/obsproc/src/obsproc.exe

The main executable for running WRFDA is da_wrfvar.exe. Make sure it has been created after the compilation: it is fairly common that all the executables will be successfully compiled except this main executable. If this occurs, please check the compilation log file carefully for any errors.
The basic gen_be utility for the regional model consists of gen_be_stage0_wrf.exe, gen_be_stage1.exe, gen_be_stage2.exe, gen_be_stage2a.exe, gen_be_stage3.exe, gen_be_stage4_regional.exe, and gen_be_diags.exe.

da_update_bc.exe is used for updating the WRF lower and lateral boundary conditions before and after a new WRFDA analysis is generated. This is detailed in the section on Updating WRF Boundary Conditions.

da_advance_time.exe is a very handy and useful tool for date/time manipulation. Type $WRFDA_DIR/var/build/da_advance_time.exe to see its usage instructions.

obsproc.exe is the executable for preparing conventional observations for assimilation by WRFDA. Its use is detailed in the section on Running Observation Preprocessor.

If you plan on using CRTM for radiance assimilation, check $WRFDA_DIR/var/external/crtm_2.1.3/libsrc to ensure that libCRTM.a was generated.

c. Clean Compilation

To remove all object files and executables, type:

    ./clean

To remove all build files, including configure.wrf, type:

    ./clean -a

The clean –a command is recommended if your compilation fails, or if the configuration file has been changed and you wish to restore the default settings.

Installing WRFPLUS and WRFDA for 4D-Var Run

If you intend to run WRFDA 4DVAR, it is necessary to have WRFPLUS installed. WRFPLUS contains the adjoint and tangent linear models based on a simplified WRF model, which includes a few simplified physics packages, such as surface drag, large scale condensation and precipitation, and cumulus parameterization.

    Note: if you intend to run both 3DVAR and 4DVAR experiments, it is not necessary to compile the code twice. The da_wrfvar.exe executable compiled for 4DVAR can be used for both 3DVAR and 4DVAR assimilation.

To install WRFPLUS:
• Get the WRFPLUS zipped tar file from
  
  http://www2.mmm.ucar.edu/wrf/users/wrfda/download/wrfplus.html

• Unzip and untar the WRFPLUS file, then run the configure script

  > gunzip WRFPLUSV3.8.1.tar.gz
  > tar -xf WRFPLUSV3.8.1.tar
  > cd WRFPLUSV3
  > ./configure wrfplus

  As with 3D-Var, “serial” means single-processor, and “dmpar” means Distributed Memory Parallel (MPI). Be sure to select the same option for WRFPLUS as you will use for WRFDA.

• Compile WRFPLUS

  > ./compile wrf >& compile.out
  > ls -ls main/*.exe

  If compilation was successful, you should see the WRFPLUS executable (named wrf.exe):

  53292 -rwxr-xr-x 1 user man 54513254 Apr  6 22:43 main/wrf.exe

  Finally, set the environment variable WRFPLUS_DIR to the appropriate directory:

  > setenv WRFPLUS_DIR ${your_source_code_dir}/WRFPLUSV3

To install WRFDA for the 4D-Var run:

• If you intend to use RTTOV to assimilate radiance data, you will need to set the appropriate environment variable at compile time. See the previous 3D-Var section for instructions.

  >./configure 4dvar

  >./compile all_wrfvar >& compile.out

  >ls -ls var/build/*.exe var/obsproc/*.exe

  You should see the same 44 executables as are listed in the above 3D-Var section, including da_wrfvar.exe

**Running Observation Preprocessor (OBSPROC)**

The OBSPROC program reads observations in LITTLE_R format (a text-based format, in use since the MM5 era). We have provided observations for the tutorial case, but for your
own applications, you will have to prepare your own observation files. Please see
http://www2.mmm.ucar.edu/wrf/users/wrfda/download/free_data.html for the sources of some
freely-available observations. Because the raw observation data files have many possible
formats, such as ASCII, BUFR, PREPBUFR, MADIS (*note: a converter for MADIS data
to LITTLE_R is available on the WRFDA website:
http://www2.mmm.ucar.edu/wrf/users/wrfda/download/madis.html), and HDF, the free data site also
contains instructions for converting the observations to LITTLE_R format. To make the
WRFDA system as general as possible, the LITTLE_R format was adopted as an inter-
mediate observation data format for the WRFDA system, however, *the conversion of the
user-specific source data to LITTLE_R format is the user’s task.* A more complete
description of the LITTLE_R format, as well as conventional observation data sources for
WRFDA, can be found by reading the “Observation Pre-processing” tutorial found at
http://www2.mmm.ucar.edu/wrf/users/wrfda/Tutorials/2015_Aug/tutorial_presentations_summer_2015.html, or by
referencing *Chapter 7 of this User’s Guide.*

The purpose of OBSPROC is to:

- Remove observations outside the specified temporal and spatial domains
- Re-order and merge duplicate (in time and location) data reports
- Retrieve pressure or height based on observed information using the hydrostatic
  assumption
- Check multi-level observations for vertical consistency and superadiabatic condi-
  tions
- Assign observation errors based on a pre-specified error file
- Write out the observation file to be used by WRFDA in ASCII or BUFR format

The OBSPROC program (obsproc.exe) should be found under the directory
$WRFDA_DIR/var/obsproc/src if “compile all_wrfvar” completed successfully.

If you haven’t already, you should download the tutorial case, which contains example
files for all the exercises in this User’s Guide. The case can be found at the WRFDA
website (http://www2.mmm.ucar.edu/wrf/users/wrfda/download/testdata.html).

**a. Prepare observational data for 3D-Var**

As an example, to prepare the observation file at the analysis time, all the observations in
the range ±1h will be processed, which means that (in the example case) the observations
between 23h and 1h are treated as the observations at 0h. This is illustrated in the follow-
ing figure:
OBSPROC requires at least 3 files to run successfully:

- A namelist file (`namelist.obsproc`)
- An observation error file (`obserr.txt`)
- One or more observation files
- Optionally, a table for specifying the elevation information for marine observations over the US Great Lakes (`msfc.tbl`)

The files `obserr.txt` and `msfc.tbl` are included in the source code under `var/obsproc`. To create the required namelist file, we have provided an example file (`namelist_obsproc.3dvar.wrfvar-tut`) in the `var/obsproc` directory. Thus, proceed as follows.

```shell
> cd $WRFDA_DIR/var/obsproc
> cp namelist.obsproc.3dvar.wrfvar-tut namelist.obsproc
```

Next, edit the namelist file, `namelist.obsproc`, to accommodate your experiments. You will likely only need to change variables listed under records 1, 2, 6, 7, and 8. See `$WRFDA_DIR/var/obsproc/README.namelist`, or the section Description of Namelist Variables for details. You should pay special attention to the record 7 and record 8 variables: these will determine the domain for which observations will be written to the output observation file. Alternatively, if you do not wish to filter the observations spatially, you can set `domain_check_h = .false. under &record4`.

If you are running the tutorial case, you should copy or link the sample observation file (`ob/2008020512/ob.2008020512`) to the `obsproc` directory. Alternatively, you can edit the namelist variable `obs_gts_filename` to point to the observation file’s full path.

To run OBSPROC, type

```shell
> ./obsproc.exe >& obsproc.out
```

Once `obsproc.exe` has completed successfully, you will see an observation data file, with the name formatted `obs_gtsYYYY-MM-DD_HH:NN:SS.3DVAR`, in the `obsproc` directory.
Before running WRFDA, you may find it useful to learn more about various types of data that will be processed (e.g., their geographical distribution). This file is in ASCII format and so you can easily view it. For a graphical view of the file's content, there are NCL scripts available which can display the distribution and type of observations. In the WRFDA Tools package (can be downloaded at [http://www2.mmm.ucar.edu/wrf/users/wrfda/download/tools.html](http://www2.mmm.ucar.edu/wrf/users/wrfda/download/tools.html)), the relevant script is located at $TOOLS_DIR/var/graphics/ncl/plot_ob_ascii_loc.ncl. You will need to have NCL installed in your system to use this script; for more information on NCL, the NCAR Command Language, see [http://www.ncl.ucar.edu/](http://www.ncl.ucar.edu/).

### b. Prepare observational data for 4D-Var

To prepare the observation file, for example, at the analysis time 0h for 4D-Var, all observations from 0h to 6h will be processed and grouped in 7 sub-windows (slot1 through slot7) as illustrated in the following figure:
NOTE: The “Analysis time” in the above figure is not the actual analysis time (0h). It indicates the time_analysis setting in the namelist file, which in this example is three hours later than the actual analysis time. The actual analysis time is still 0h.

An example namelist (namelist_obsproc.4dvar.wrfvar-tut) has already been provided in the var/obsproc directory. Thus, proceed as follows:

```bash
> cd $WRFDA_DIR/var/obsproc
> cp namelist.obsproc.4dvar.wrfvar-tut namelist.obsproc
```

In the namelist file, you need to change the following variables to accommodate your experiments. In this tutorial case, the actual analysis time is 2008-02-05_12:00:00, but in the namelist, time_analysis should be set to 3 hours later. The different values of time_analysis, num_slots_past, and time_slots_ahead contribute to the actual times analyzed. For example, if you set time_analysis = 2008-02-05_16:00:00, and set the num_slots_past = 4 and time_slots_ahead=2, the final results will be the same as before.

Edit all the domain settings according to your own experiment; a full list of namelist options and descriptions can be found in the section Description of Namelist Variables. You should pay special attention to the record 7 and record 8 variables: these will determine the domain for which observations will be written to the output observation file. Alternatively, if you do not wish to filter the observations spatially, you can set domain_check_h = .false. under &record4.

If you are running the tutorial case, you should copy or link the sample observation file (ob/2008020512/obs.2008020512) to the obsproc directory. Alternatively, you can edit the namelist variable obs_gts_filename to point to the observation file’s full path.

To run OBSPROC, type

```bash
> obsproc.exe >& obsproc.out
```

Once obsproc.exe has completed successfully, you will see 7 observation data files, which for the tutorial case are named...
obs_gts_2008-02-05_12:00:00.4DVAR
obs_gts_2008-02-05_13:00:00.4DVAR
obs_gts_2008-02-05_14:00:00.4DVAR
obs_gts_2008-02-05_15:00:00.4DVAR
obs_gts_2008-02-05_16:00:00.4DVAR
obs_gts_2008-02-05_17:00:00.4DVAR
obs_gts_2008-02-05_18:00:00.4DVAR

They are the input observation files to WRF 4D-Var.

**Running WRFDA**

**a. Download Test Data**

The WRFDA system requires three input files to run:

a) **WRF first guess** file, output from either WPS/real.exe (cold-start) or a WRF forecast (warm-start)

b) **Observations** (in ASCII format, PREPBUFR or BUFR for radiance)

c) **A background error statistics file** (containing background error covariance)

The following table summarizes the above info:

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Format</th>
<th>Created By</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Guess</td>
<td>NETCDF</td>
<td>WRF Preprocessing System (WPS) and real.exe or WRF</td>
</tr>
<tr>
<td>Observations</td>
<td>ASCII (PREPBUFR also possible)</td>
<td><strong>Observation Preprocessor</strong> (OBSPROC)</td>
</tr>
<tr>
<td>Background Error Statistics</td>
<td>Binary</td>
<td><strong>WRFDA gen_be utility</strong> or generic CV3</td>
</tr>
</tbody>
</table>

In the test case, you will store data in a directory defined by the environment variable `$DAT_DIR`. This directory can be in any location, and it should have read access. Type

> setenv DAT_DIR your_choice_of_dat_dir

Here, `your_choice_of_dat_dir` is the directory where the WRFDA input data is stored.

If you have not already done so, download the sample data for the tutorial case, valid at 12 UTC 5th February 2008, from http://www2.mmm.ucar.edu/wrf/users/wrfda/download/testdata.html
Once you have downloaded the WRFDAV3.8-testdata.tar.gz file to $DAT_DIR, extract it by typing

```
> gunzip WRFDAV3.8-testdata.tar.gz
> tar -xvf WRFDAV3.8-testdata.tar
```

Now you should find the following four files under "$DAT_DIR"

- `ob/2008020512/ob.2008020512` # Observation data in “little_r” format
- `rc/2008020512/wrfinput_d01` # First guess file
- `rc/2008020512/wrfbdy_d01` # Lateral boundary file
- `be/be.dat` # Background error file

......

At this point you should have three of the input files (first guess, observations from OBSPROC, and background error statistics files in the directory $DAT_DIR) required to run WRFDA, and have successfully downloaded and compiled the WRFDA code. If this is correct, you are ready to run WRFDA.

**b. Run the Case—3D-Var**

The data for the tutorial case is valid at 12 UTC 5 February 2008. The first guess comes from the NCEP FNL (Final) Operational Global Analysis data, passed through the WRF-WPS and real.exe programs.

To run WRF 3D-Var, first create and enter into a working directory (for example, $WRFDA_DIR/workdir), and set the environment variable WORK_DIR to this directory (e.g., `setenv WORK_DIR $WRFDA_DIR/workdir`). Then follow the steps below:

```
> cd $WORK_DIR
> cp $DAT_DIR/namelist.input.3dvar namelist.input
> ln -sf $WRFDA_DIR/run/LANDUSE.TBL .
> ln -sf $DAT_DIR/rc/2008020512/wrfinput_d01 ./fg
> ln -sf $DAT_DIR/ob/2008020512/obs_gts_2008-02-05_12:00:00.3DVAR ./ob.ascii (note the different name!)
> ln -sf $DAT_DIR/be/be.dat .
> ln -sf $WRFDIR/da/da_wrfvar.exe .
```

Now edit the file `namelist.input`, which is a very basic namelist for the tutorial test case, and is shown below.

```
&wrfvar1
var4d=false,
print_detail_grad=false,
/
&wrfvar2
/
&wrfvar3
ob_format=2,
/
&wrfvar4
/
&wrfvar5
/
```

&wrfvar6
max_ext_its=1,
ntmax=50,
orthonorm_gradient=true,
/
&wrfvar7
cv_options=5,
/
&wrfvar8
/
&wrfvar9
/
&wrfvar10
test_transforms=false,
test_gradient=false,
/
&wrfvar11
/
&wrfvar12
/
&wrfvar13
/
&wrfvar14
/
&wrfvar15
/
&wrfvar16
/
&wrfvar17
/
&wrfvar18
analysis_date="2008-02-05_12:00:00.0000",
/
&wrfvar19
/
&wrfvar20
/
&wrfvar21
time_window_min="2008-02-05_11:00:00.0000",
/
&wrfvar22
time_window_max="2008-02-05_13:00:00.0000",
/
&time_control
start_year=2008,
start_month=02,
start_day=05,
start_hour=12,
end_year=2008,
end_month=02,
end_day=05,
end_hour=12,
/
&fdda
/
&domains
e_we=90,
e_sn=60,
e_vert=41,
dx=60000,
dy=60000,
/
&dfi_control
/
&tc
/
&physics
mp_physics=3,
ra_lw_physics=1,
ra_sw_physics=1,
radt=60,
sf_sfclay_physics=1,
sf_surface_physics=1,
No edits should be needed if you are running the tutorial case without radiance data. If you plan to use the PREPBUFR-format data, change the ob_format=1 in &wrfvar3 in namelist.input and link the data as ob.bufr.

```bash
> ln -fs $DAT_DIR/ob/2008020512/gdas1.t12z.prepbufr.nr ob.bufr
```

Once you have changed any other necessary namelist variables, run WRFDA 3D-Var:

```bash
> da_wrfvar.exe >& wrfda.log
```

The file `wrfda.log` (or `rsl.out.0000`, if run in distributed-memory mode) contains important WRFDA runtime log information. Always check the log after a WRFDA run:

```
*** VARIATIONAL ANALYSIS ***
DYNAMICS OPTION: Eulerian Mass Coordinate
 alloc_space_field: domain            1 , 419050728 bytes allocated
 Tile Strategy is not specified. Assuming 1D-Y
 WRF TILE 1 IS  1 IE  89 JS  1 JE  59
 WRF NUMBER OF TILES =  1
Set up observations (ob)

Using ASCII format observation input
Observation summary
  ob time 1
  sound 86 global, 86 local
  synop 750 global, 750 local
  pilot 85 global, 85 local
  satem 105 global, 105 local
  geoamv 2499 global, 2499 local
  airep 216 global, 216 local
  gpspw 187 global, 187 local
  gprsrf 3 global, 3 local
  metar 2408 global, 2408 local
  ships 135 global, 135 local
  qscat 2126 global, 2126 local
  profiler 61 global, 61 local
  buoy 247 global, 247 local
  sonde_sfc 86 global, 86 local
Set up background errors for regional application for cv_options = 5
```
Using the averaged regression coefficients for unbalanced part

WRFDA dry control variables are: psi, chi_u, t_u and ps_u
Humidity control variable is rh

Vertical truncation for psi = 16 (99.00%)
Vertical truncation for chi_u = 21 (99.00%)
Vertical truncation for t_u = 28 (99.00%)
Vertical truncation for rh = 21 (99.00%)

Scaling: var, len, ds: 0.100000E+01 0.100000E+01 0.600000E+05
Scaling: var, len, ds: 0.100000E+01 0.100000E+01 0.600000E+05
Scaling: var, len, ds: 0.100000E+01 0.100000E+01 0.600000E+05
Scaling: var, len, ds: 0.100000E+01 0.100000E+01 0.600000E+05
Scaling: var, len, ds: 0.100000E+01 0.100000E+01 0.600000E+05

Calculate innovation vector (iv)

Minimize cost function using CG method

Starting outer iteration: 1
Starting cost function: 2.89907454207616D+04, Gradient= 4.257569431368675D+02
For this outer iteration gradient target is: 4.257569431368675D+00

<table>
<thead>
<tr>
<th>Iter</th>
<th>Cost Function</th>
<th>Gradient</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.771806953353228D+04</td>
<td>3.89443400010825D+02</td>
<td>1.404189554585295D-02</td>
</tr>
<tr>
<td>2</td>
<td>2.607328841808729D+04</td>
<td>3.776010549295726D+02</td>
<td>2.23152424457818D-02</td>
</tr>
<tr>
<td>3</td>
<td>2.472259943260248D+04</td>
<td>3.22269578709167D+02</td>
<td>1.89458616664622D-02</td>
</tr>
<tr>
<td>4</td>
<td>2.38608665241553D+04</td>
<td>2.43464087175013D+02</td>
<td>1.65945534972616D-02</td>
</tr>
<tr>
<td>5</td>
<td>2.32765820675138D+04</td>
<td>2.26325649097211D+02</td>
<td>1.98463869146577D-02</td>
</tr>
<tr>
<td>6</td>
<td>2.28838707811670D+04</td>
<td>1.97280920254657D+02</td>
<td>1.51809315679554D-02</td>
</tr>
<tr>
<td>7</td>
<td>2.25385262144951D+04</td>
<td>1.76642834719637D+02</td>
<td>1.37469498211314D-02</td>
</tr>
<tr>
<td>8</td>
<td>2.21941774422390D+04</td>
<td>1.61892287824531D+02</td>
<td>2.20763722476843D-02</td>
</tr>
<tr>
<td>9</td>
<td>2.19236822168224D+04</td>
<td>1.22982646495943D+02</td>
<td>2.06413110543492D-02</td>
</tr>
<tr>
<td>10</td>
<td>2.17723415187660D+04</td>
<td>1.09845293230578D+02</td>
<td>2.00123460476772D-02</td>
</tr>
<tr>
<td>11</td>
<td>2.1655145604121D+04</td>
<td>9.03882133939826D+01</td>
<td>1.94243445990581D-02</td>
</tr>
<tr>
<td>12</td>
<td>2.15490544907016D+04</td>
<td>6.17171229003712D+01</td>
<td>2.59729966471773D-02</td>
</tr>
<tr>
<td>13</td>
<td>2.14826583275935D+04</td>
<td>5.90027652146414D+01</td>
<td>3.29654210894269D-02</td>
</tr>
<tr>
<td>14</td>
<td>2.14438889933652D+04</td>
<td>5.5858808504704D+02</td>
<td>2.48212312370112D-02</td>
</tr>
<tr>
<td>15</td>
<td>2.14109465673664D+02</td>
<td>4.32857439190531D+01</td>
<td>2.10644338424475D-02</td>
</tr>
<tr>
<td>16</td>
<td>2.13884197931919D+04</td>
<td>4.62385983298469D+04</td>
<td>2.04458005237759D-02</td>
</tr>
<tr>
<td>17</td>
<td>2.13737093934308D+04</td>
<td>2.94484535910351D+02</td>
<td>1.59487080521242D-02</td>
</tr>
<tr>
<td>18</td>
<td>2.13580107761494D+04</td>
<td>2.70311945287092D+01</td>
<td>3.08105202543609D-02</td>
</tr>
<tr>
<td>19</td>
<td>2.13499092836225D+04</td>
<td>2.67607032351367D+02</td>
<td>2.64621212370112D-02</td>
</tr>
<tr>
<td>20</td>
<td>2.13424430434521D+02</td>
<td>1.89572473612341D+01</td>
<td>2.73421291452306D-07</td>
</tr>
<tr>
<td>21</td>
<td>2.13374366897855D+04</td>
<td>1.37180941886291D+01</td>
<td>2.78111365357901D-04</td>
</tr>
<tr>
<td>22</td>
<td>2.13343901337165D+04</td>
<td>1.23285998916033D+02</td>
<td>3.23781866762183D-04</td>
</tr>
<tr>
<td>23</td>
<td>2.13317508367775D+04</td>
<td>9.69687152746297D+00</td>
<td>3.47288751209171D-06</td>
</tr>
<tr>
<td>24</td>
<td>2.13303361867996D+04</td>
<td>7.71918878570402D+03</td>
<td>3.00895121505260D-04</td>
</tr>
<tr>
<td>25</td>
<td>2.13293737269109D+04</td>
<td>6.17175020843937D+04</td>
<td>3.23048769634235D-07</td>
</tr>
<tr>
<td>26</td>
<td>2.13280381311464D+04</td>
<td>5.05545249057161D+00</td>
<td>2.99243741464962D-02</td>
</tr>
<tr>
<td>27</td>
<td>2.13281963437784D+04</td>
<td>4.26744234103547D+00</td>
<td>4.75372757176031D-02</td>
</tr>
<tr>
<td>28</td>
<td>2.132795798122098D+04</td>
<td>3.55954764871664D+03</td>
<td>2.61777365364680D-02</td>
</tr>
</tbody>
</table>

Diagnostics
Final cost function J = 21327.96
Total number of obs. = 39820
Final value of J = 21327.95798
Final value of Jo = 18771.08660
Final value of Jb = 2556.87138
Final value of Je = 0.00000
Final value of Jc = 0.00000

Inner iteration stopped after 28 iterations
Final: 28 iter, J = 2.132795798122041D+04, g = 3.559547648716685D+00

Diagnostics
Final value of Jp = 0.00000
Final value of Jl = 0.00000
Final J / total num_obs = 0.53561
Jb factor used(1) = 1.00000
Jb factor used(2) = 1.00000
Jb factor used(3) = 1.00000
Jb factor used(4) = 1.00000
Jb factor used(5) = 1.00000
Jb factor used = 1.00000
Je factor used = 1.00000
VarBC factor used = 1.00000

*** WRF - Var completed successfully ***

The file namelist.output.da (which contains the complete namelist settings) will be generated after a successful run of da_wrfvar.exe. The settings appearing in namelist.output.da, but not specified in your namelist.input, are the default values from $WRFDA_DIR/Registry/registry.var.

After successful completion, wrfvar_output (the WRFDA analysis file, i.e. the new initial condition for WRF) should appear in the working directory along with a number of diagnostic files. Text files containing various diagnostics will be explained in the WRFDA Diagnostics section.

To understand the role of various important WRFDA options, try re-running WRFDA by changing different namelist options. For example, try making the WRFDA convergence criterion more stringent. This is achieved by reducing the value of “EPS” to e.g. 0.0001 by adding "EPS=0.0001" in the namelist.input record &wrfvar6. See the section Additional WRFDA exercises for more namelist options.

c. Run the Case—4D-Var

To run WRF 4D-Var, first create and enter a working directory, such as $WRFDA_DIR/workdir. Set the WORK_DIR environment variable (e.g. setenv WORK_DIR $WRFDA_DIR/workdir)

For the tutorial case, the analysis date is 2008020512 and the test data directories are:

> setenv DAT_DIR {directory where data is stored}
> ls -lr $DAT_DIR
ob/2008020512
ob/2008020513
ob/2008020514
ob/2008020515
ob/2008020516
ob/2008020517
ob/2008020518
rc/2008020512
be

Note: WRFDA 4D-Var is able to assimilate conventional observational data, satellite radiance BUFR data, and precipitation data. The input data format can be PREPBUFR format data or ASCII observation data, processed by OBSPROC.

Now follow the steps below:
1) Link the executable file

   > cd $WORK_DIR
   > ln -fs $WRFDA_DIR/var/da/da_wrfvar.exe

2) Link the observational data, first guess, BE and LANDUSE.TBL, etc.

   > ln -fs $DAT_DIR/ob/2008020512/ob01.ascii
   > ln -fs $DAT_DIR/ob/2008020513/ob02.ascii
   > ln -fs $DAT_DIR/ob/2008020514/ob03.ascii
   > ln -fs $DAT_DIR/ob/2008020515/ob04.ascii
   > ln -fs $DAT_DIR/ob/2008020516/ob05.ascii
   > ln -fs $DAT_DIR/ob/2008020517/ob06.ascii
   > ln -fs $DAT_DIR/ob/2008020518/ob07.ascii
   > ln -fs $DAT_DIR/rc/2008020512/wrfinput_d01
   > ln -fs $DAT_DIR/rc/2008020512/wrfbdy_d01
   > ln -fs wrfinput_d01 fg
   > ln -fs $DAT_DIR/be/be.dat
   > ln -fs $WRFDA_DIR/run/LANDUSE.TBL
   > ln -fs $WRFDA_DIR/run/GENPARM.TBL
   > ln -fs $WRFDA_DIR/run/SOILPARM.TBL
   > ln -fs $WRFDA_DIR/run/VEGPARM.TBL
   > ln -fs $WRFDA_DIR/run/RRTM_DATA DBL RRTM_DATA

3) Copy the sample namelist

   > cp $DAT_DIR/namelist.input.4dvar namelist.input

4) Edit necessary namelist variables, link optional files

WRFDA 4D-Var has the capability to consider lateral boundary conditions as control variables as well during minimization. The namelist variable var4d_lbc=true turns on this capability. To enable this option, WRF 4D-Var needs not only the first guess at the beginning of the time window, but also the first guess at the end of the time window.

   > ln -fs $DAT_DIR/rc/2008020518/wrfinput_d01 fg02

Please note: WRFDA beginners should not use this option until you have a good understanding of the 4D-Var lateral boundary conditions control. To disable this feature, make sure var4d_lbc in namelist.input is set to false.

If you use PREPBUFR format data, set ob_format=1 in &wrfvar3 in namelist.input. Because 12UTC PREPBUFR data only includes the data from 9UTC to 15UTC, for 4D-Var you should include 18UTC PREPBUFR data as well:

   > ln -fs $DAT_DIR/ob/2008020512/gdas1.t12z.prepbufr.nr ob01.bufr
   > ln -fs $DAT_DIR/ob/2008020518/gdas1.t18z.prepbufr.nr ob02.bufr

Edit namelist.input to match your experiment settings. The most important namelist variables related to 4D-Var are listed below. Please refer to README.namelist under the $WRFDA_DIR/var directory. A common mistake users make is in the time information settings. The rules are: analysis_date, time_window_min and start_xxx in &time_control should always be equal to each other; time_window_max and end_xxx
should always be equal to each other; and run_hours is the difference between start_xxx and end_xxx, which is the length of the 4D-Var time window.

```
&wrfvar1
var4d=true,
var4d_lbc=false,
var4d_bin=3600,
......

&time_control
run_hours=6,
start_year=2008,
start_month=02,
start_day=05,
start_hour=12,
end_year=2008,
end_month=02,
end_day=05,
end_hour=18,
interval_seconds=21600,
debug_level=0,
......
```

5) Run WRF 4D-Var

```
> cd $WORK_DIR
> ./da_wrfvar.exe >& wrfda.log
```

4DVAR is much more computationally expensive than 3DVAR, so running may take a while; you can set \texttt{ntmax} to a lower value so that WRFDA uses fewer minimization steps. You can also MPI with multiple processors to speed up the execution:

```
> mpirun -np 4 ./da_wrfvar.exe >& wrfda.log &
```

The “mpirun” command may be different depending on your machine. The output logs will be found in files named \texttt{rsl.out.####} and \texttt{rsl.error.####} for MPI runs.

Please note: If you utilize the lateral boundary conditions option (\texttt{var4d_lbc=true}), in addition to the analysis at the beginning of the time window (\texttt{wrfvar_output}), the analysis at the end of the time window will also be generated as \texttt{ana02}, which will be used in subsequent updating of boundary conditions before the forecast.
Radiance Data Assimilation in WRFDA

This section gives a brief description for various aspects related to radiance assimilation in WRFDA. Each aspect is described mainly from the viewpoint of usage, rather than more technical and scientific details, which will appear in a separate technical report and scientific paper. Namelist parameters controlling different aspects of radiance assimilation will be detailed in the following sections. It should be noted that this section does not cover general aspects of the assimilation process with WRFDA; these can be found in other sections of chapter 6 of this user’s guide, or other WRFDA documentation.

a. Running WRFDA with radiances

In addition to the basic input files ($\text{LANDUSE.\_TBL, fg, ob.\_ascii, be.\_dat}$) mentioned in the “Running WRFDA” section, the following additional files are required for radiances: radiance data (typically in NCEP BUFR format), radiance_info files, VARBC.in (if you plan on using variational bias correction VARBC, as described in the section on bias correction), and RTM (CRTM or RTTOV) coefficient files.

Edit namelist.input (Pay special attention to &wrfvar4, &wrfvar14, &wrfvar21, and &wrfvar22 for radiance-related options. A very basic namelist.input for running the radiance test case is provided in WRFDA/var/test/radiance/namelist.input)

> ln -sf $\text{DAT\_DIR/gdas1.t00z.1bamua.tm00.bufr\_d} ./amsua.bufr
> ln -sf $\text{DAT\_DIR/gdas1.t00z.1bamub.tm00.bufr\_d} ./amsub.bufr
> ln -sf $\text{WRFDA\_DIR/var/run/radiance\_info} ./radiance\_info # (radiance\_info is a directory)
> ln -sf $\text{WRFDA\_DIR/var/run/VARBC\_in} ./VARBC\_in
(CRTM only) > ln -sf $\text{WRFDA\_DIR/var/run/crtm\_coeffs} ./crtm\_coeffs
# (crtm\_coeffs is a directory)
(RTTOV only) > ln -sf your\_RTTOV\_path/rtcoef\_rttov1/rttov7pred54L ./rttov\_coeffs # (rttov\_coeffs is a directory)
(HDF5 only) > ln -sf $\text{WRFDA\_DIR/var/run/leapsec\_dat} .

See the following sections for more details on each aspect of radiance assimilation.

Note: You can also specify the path of the “crtm\_coeffs” directory via the namelist; see the following section for more details

b. Radiance Data Ingest

Currently, the ingest interface for NCEP BUFR radiance data is implemented in WRFDA. The radiance data are available through NCEP’s public ftp server (ftp://ftp.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gdas.$\{yyyymmddhh\} in near real-time (with a 6-hour delay) and can meet requirements for both research purposes and some real-time applications.

As of Version 3.8.1, WRFDA can read data from NOAA ATOVS instruments (HIRS, AMSU-A, AMSU-B and MHS), EOS Aqua instruments (AIRS, AMSU-A), DMSP instruments (SSMIS), METOP instruments (HIRS, AMSU-A, MHS, IASI), Meteosat in-
WRFDA

Instruments (SEVIRI), and JAXA GCOM-W1 instruments (AMSR2). Note that NCEP radiance BUFR files are separated by instrument names (i.e., one file for each type of instrument), and each file contains global radiance (generally converted to brightness temperature) within a 6-hour assimilation window, from multi-platforms. For running WRFDA, users need to rename NCEP corresponding BUFR files (table 1) to hirs3.bufr (including HIRS data from NOAA-15/16/17), hirs4.bufr (including HIRS data from NOAA-18/19, METOP-2), amsua.bufr (including AMSU-A data from NOAA-15/16/18/19, METOP-1 and -2), amsub.bufr (including AMSU-B data from NOAA-15/16/17), mhs.bufr (including MHS data from NOAA-18/19 and METOP-1 and -2), airs.bufr (including AIRS and AMSU-A data from EOS-AQUA) ssmis.bufr (SSMIS data from DMSP-16, AFWA provided) iasi.bufr (IASI data from METOP-1 and -2) and seviri.bufr (SEVIRI data from Meteosat 8-10) for WRFDA filename convention. Note that the airs.bufr file contains not only AIRS data but also AMSU-A, which is collocated with AIRS pixels (1 AMSU-A pixel collocated with 9 AIRS pixels). Users must place these files in the working directory where the WRFDA executable is run. It should also be mentioned that WRFDA reads these BUFR radiance files directly without the use of any separate pre-processing program. All processing of radiance data, such as quality control, thinning, bias correction, etc., is carried out within WRFDA. This is different from conventional observation assimilation, which requires a pre-processing package (OBSPROC) to generate WRFDA readable ASCII files. For reading the radiance BUFR files, WRFDA must be compiled with the NCEP BUFR library (see http://www.ncno.ncep.noaa.gov/sib/decoders/BUFRLIB/).

Table 1: NCEP and WRFDA radiance BUFR file naming convention

<table>
<thead>
<tr>
<th>NCEP BUFR file names</th>
<th>WRFDA naming convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdas1.t00z.airsev.tm00.bufr_d</td>
<td>airs.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.1bamua.tm00.bufr_d</td>
<td>amsua.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.1bamub.tm00.bufr_d</td>
<td>amsub.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.atms.tm00.bufr_d</td>
<td>atms.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.1bhrs3.tm00.bufr_d</td>
<td>hirs3.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.1bhrs4.tm00.bufr_d</td>
<td>hirs4.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.mtiasi.tm00.bufr_d</td>
<td>iasi.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.1bmhs.tm00.bufr_d</td>
<td>mhs.bufr</td>
</tr>
<tr>
<td>gdas1.t00z.sevcsr.tm00.bufr_d</td>
<td>seviri.bufr</td>
</tr>
</tbody>
</table>

Namelist parameters are used to control the reading of corresponding BUFR files into WRFDA. For instance, USE_AMSUAOBS, USE_AMSUBOBS, USE_HIRS3OBS, USE_HIRS4OBS, USE_MHSOBS, USE_AIRSOBS, USE_EOS_AMSUAOBS, USE_SSMISOBS, USE_ATMSOBS, USE_IASIOBS, and USE_SEVIRIOBS control whether or not the respective file is read. These are logical parameters that are assigned to .FALSE. by default; therefore they must be set to .TRUE. to read the respective observation file. Also note that these parameters only control whether the data is read, not whether the data included in the files is to be assimilated. This is controlled by other namelist parameters explained in the next section.

Sources for downloading these and other data can be found on the WRFDA website:
Other data formats

Most of the above paragraphs describe NCEP BUFR data, but some of the satellite data supported by WRFDA are in alternate formats. Level-1R AMSR2 data from the JAXA GCOM-W1 satellite are available in HDF5 format, which requires compiling WRFDA with HDF5 libraries, as described in the “Compile WRFDA and Libraries” section.

HDF5 file naming conventions are different than those for BUFR files. For AMSR2 data, WRFDA will look for two data files: L1SGRTBR.h5 (brightness temperature) and L2SGCLWLD.h5 (cloud liquid water). Only the brightness temperature file is mandatory (you will also need to copy or link the “leapsec.dat” file from WRFDA/var/run). If you have multiple data files for your assimilation window, you should name them L1SGRTBR-01.h5, L1SGRTBR-02.h5, etc. and L2SGCLWLD-01.h5, L2SGCLWLD-02.h5, etc.

c. Radiative Transfer Model

The core component for direct radiance assimilation is to incorporate a radiative transfer model (RTM) into the WRFDA system as one part of observation operators. Two widely used RTMs in the NWP community, RTTOV (developed by ECMWF and UKMET in Europe), and CRTM (developed by the Joint Center for Satellite Data Assimilation (JCSDA) in US), are already implemented in the WRFDA system with a flexible and consistent user interface. WRFDA is designed to be able to compile with or without RTTOV by the definition of the “RTTOV” environment variable at compile time (see the “Compile WRFDA and Libraries” section). At runtime the user must select which RTM they intend to use via the namelist parameter RTM_OPTION (1 for RTTOV, the default, and 2 for CRTM).

Both RTMs can calculate radiances for almost all available instruments aboard the various satellite platforms in orbit. An important feature of the WRFDA design is that all data structures related to radiance assimilation are dynamically allocated during running time, according to a simple namelist setup. The instruments to be assimilated are controlled at run-time by four integer namelist parameters: RTMINIT_NSENSOR (the total number of sensors to be assimilated), RTMINIT_PLATFORM (the platforms IDs array to be assimilated with dimension RTMINIT_NSENSOR, e.g., 1 for NOAA, 9 for EOS, 10 for METOP and 2 for DMSP), RTMINIT_SATID (satellite IDs array) and RTMINIT_SENSOR (sensor IDs array, e.g., 0 for HIRS, 3 for AMSU-A, 4 for AMSU-B, etc.). The full list of instrument triplets can be found in the table below:

<table>
<thead>
<tr>
<th>Instrument</th>
<th>Satellite</th>
<th>Format</th>
<th>(PLATFORM, SATID, SENSOR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIRS</td>
<td>EOS-Aqua</td>
<td>BUFR</td>
<td>(9,2,11)</td>
</tr>
<tr>
<td>AMSR2</td>
<td>GCOM-W1</td>
<td>HDF5</td>
<td>(29,1,63)</td>
</tr>
<tr>
<td>AMSU-A</td>
<td>EOS-Aqua</td>
<td>BUFR</td>
<td>(9,2,3)</td>
</tr>
<tr>
<td>AMSU-A</td>
<td>METOP-A</td>
<td>BUFR</td>
<td>(10,2,3)</td>
</tr>
</tbody>
</table>
Here's an example of this section of the namelist for a user assimilating IASI observations from METOP-A, utilizing RTTOV as their RTM:

```wrfvar14
  rtminit_nsensor = 1
  rtminit_platform = 10,
  rtminit_satid = 2,
  rtminit_sensor = 16,
  rtm_option = 1,
/
```

Here’s another example of this section of the namelist, this time for a user assimilating AMSU-A from NOAA 18–19 and EOS-Aqua, MHS from NOAA 18–19, and AIRS from EOS-Aqua, utilizing CRTM as their RTM:

```wrfvar14
  rtminit_nsensor = 6
  rtminit_platform = 1, 1, 9, 1, 1, 9
  rtminit_satid = 18, 19, 2, 18, 19, 2
  rtminit_sensor = 3, 3, 15, 15, 11
  rtm_option = 2,
/
```

The instrument triplets (platform, satellite, and sensor ID) in the namelist can be ranked in any order. More detail about the convention of instrument triples can be found in tables 2 and 3 in the RTTOV v1.1 User’s Guide ([http://nwpsaf.eu/deliverables/rtm/docs_rttov11/users_guide_11_v1.4.pdf](http://nwpsaf.eu/deliverables/rtm/docs_rttov11/users_guide_11_v1.4.pdf))

CRTM uses a different instrument-naming method, however, a conversion routine inside WRFDA is implemented such that the user interface remains the same for RTTOV and CRTM, using the same instrument triplet for both.

When running WRFDA with radiance assimilation switched on, a set of RTM coefficient files need to be loaded. For the RTTOV option, RTTOV coefficient files are to be copied or linked to a sub-directory `rttov_coeffs/` under the working directory. For the CRTM option, CRTM coefficient files are to be copied or linked to a sub-directory.
crtm_coeffs/ under the working directory, or the location of this directory can be specified in the namelist:

```plaintext
&wrfvar14
crtm_coef_path = WRFDA/var/run/crtm_coeffs (Can be a relative or absolute path)
```

Only coefficients for instruments listed in the namelist are needed. Potentially WRFDA can assimilate all sensors as long as the corresponding coefficient files are provided. In addition, necessary developments on the corresponding data interface, quality control, and bias correction are important to make radiance data assimilate properly; however, a modular design of radiance relevant routines already facilitates the addition of more instruments in WRFDA.

The RTTOV package is not distributed with WRFDA, due to licensing restrictions. Users need to follow the instructions at [http://nwpsaf.eu/site/software/rttov/](http://nwpsaf.eu/site/software/rttov/) to download the RTTOV source code and supplement coefficient files and the emissivity atlas dataset. Only RTTOV v11 (11.1—11.3) can be used in WRFDA version 3.8.1, so if you have an older version of RTTOV you must upgrade.

As mentioned in a previous paragraph, the CRTM package is distributed with WRFDA, and is located in `$WRFDA_DIR/var/external/crtm_2.1.3`. The CRTM code in WRFDA is the same as the source code that users can download from [ftp://ftp.emc.ncep.noaa.gov/jclda/CRTM](ftp://ftp.emc.ncep.noaa.gov/jclda/CRTM), with only minor modifications (mainly for ease of compilation).

To use one or both of the above radiative transfer models, you will have to set the appropriate environment variable(s) at compile time. See the section “Compile WRFDA and Libraries” for details.

### d. Channel Selection

Channel selection in WRFDA is controlled by radiance ‘info’ files, located in the subdirectory `radiance_info`, under the working directory. These files are separated by satellites and sensors; e.g., `noaa-15-amsua.info`, `noaa-16-amsub.info`, `dmsp-16-ssmis.info` and so on. An example of 5 channels from `noaa-15-amsub.info` is shown below. The fourth column is used by WRFDA to control when to use a corresponding channel. Channels with the value “−1” in the fourth column indicate that the channel is “not assimilated,” while the value “1” means “assimilated.” The sixth column is used by WRFDA to set the observation error for each channel. Other columns are not used by WRFDA. It should be mentioned that these error values might not necessarily be optimal for your applications. It is the user’s responsibility to obtain the optimal error statistics for his/her own applications.

<table>
<thead>
<tr>
<th>Sensor channel</th>
<th>IR/MW</th>
<th>use</th>
<th>idum</th>
<th>varch</th>
<th>polarization(0:vertical;1:horizontal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>415</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0.5500000000E+01 0.0000000000E+00</td>
</tr>
<tr>
<td>415</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0.3750000000E+01 0.0000000000E+00</td>
</tr>
</tbody>
</table>
Satellite radiance is generally considered to be biased with respect to a reference (e.g., background or analysis field in NWP assimilation) due to systematic error of the observation itself, the reference field, and RTM. Bias correction is a necessary step prior to assimilating radiance data. There are two ways of performing bias correction in WRFDA. One is based on the Harris and Kelly (2001) method, and is carried out using a set of coefficient files pre-calculated with an off-line statistics package, which was applied to a training dataset for a month-long period. The other is Variational Bias Correction (VarBC). Only VarBC is introduced here, and recommended for users because of its relative simplicity in usage.

**Variational Bias Correction**

To use VarBC, set the namelist option `USE_VARBC` to TRUE and have the `VARBC.in` file in the working directory. `VARBC.in` is a VarBC setup file in ASCII format. A template is provided with the WRFDA package (`$WRFDA_DIR/var/run/VARBC.in`).

All VarBC input is passed through a single ASCII file called `VARBC.in`. Once WRFDA has run with the VarBC option switched on, it will produce a `VARBC.out` file in a similar ASCII format. This output file will then be used as the input file for the next assimilation cycle.

**VarBC Coldstart**

Coldstarting means starting the VarBC from scratch; i.e. when you do not know the values of the bias parameters.

The Coldstart is a routine in WRFDA. The bias predictor statistics (mean and standard deviation) are computed automatically and will be used to normalize the bias parameters. All coldstart bias parameters are set to zero, except the first bias parameter (= simple offset), which is set to the mode (=peak) of the distribution of the (uncorrected) innovations for the given channel.

A threshold of a number of observations can be set through the namelist option `VARBC_NOBSMIN` (default = 10), under which it is considered that not enough observations are present to keep the Coldstart values (i.e. bias predictor statistics and bias parameter values) for the next cycle. In this case, the next cycle will do another Coldstart.

**Background constraint for bias parameters**

The background constraint controls the inertia you want to impose on the predictors (i.e. the smoothing in the predictor time series). It corresponds to an extra term in the WRFDA cost function.
It is defined in the namelist via the option `VARBC_NBGERR`; the default value is 5000. This number is related to a number of observations; the bigger the number, the more inertia constraint. If these numbers are set to zero, the predictors can evolve without any constraint.

Scaling factor

The VarBC uses a specific preconditioning, which can be scaled through the namelist option `VARBC_FACTOR` (default = 1.0).

Offline bias correction

The analysis of the VarBC parameters can be performed "offline"; i.e. independently from the main WRFDA analysis. No extra code is needed. Just set the following `MAX_VERT_VAR*` namelist variables to be 0, which will disable the standard control variable and only keep the VarBC control variable.

```
MAX_VERT_VAR1=0.0
MAX_VERT_VAR2=0.0
MAX_VERT_VAR3=0.0
MAX_VERT_VAR4=0.0
MAX_VERT_VAR5=0.0
```

Freeze VarBC

In certain circumstances, you might want to keep the VarBC bias parameters constant in time (="frozen"). In this case, the bias correction is read and applied to the innovations, but it is not updated during the minimization. This can easily be achieved by setting the namelist options:

```
USE_VARBC=false
FREEZE_VARBC=true
```

Passive observations

Some observations are useful for preprocessing (e.g. Quality Control, Cloud detection) but you might not want to assimilate them. If you still need to estimate their bias correction, these observations need to go through the VarBC code in the minimization. For this purpose, the VarBC uses a separate threshold on the QC values, called "qc_varbc_bad". This threshold is currently set to the same value as "qc_bad", but can easily be changed to any ad hoc value.

f. Other namelist variables to control radiance assimilation

**RAD_MONITORING** (30)

Integer array of dimension RTMINIT_NSENSOR, 0 for assimilating mode, 1 for monitoring mode (only calculates innovation).
**THINNING**

Logical, TRUE will perform thinning on radiance data.

**THINNING_MESH (30)**

Real array with dimension RTMINIT_NSENSOR, values indicate thinning mesh (in km) for different sensors.

**QC_RAD**

Logical, controls if quality control is performed, always set to TRUE.

**WRITE_IV_RAD_ASCII**

Logical, controls whether to output observation-minus-background (O-B) files, which are in ASCII format, and separated by sensors and processors.

**WRITE_OA_RAD_ASCII**

Logical, controls whether to output observation-minus-analysis (O-A) files (including also O-B information), which are in ASCII format, and separated by sensors and processors.

**USE_ERROR_FACTOR_RAD**

Logical, controls use of a radiance error tuning factor file (radiance_error.factor) which is created with empirical values, or generated using a variational tuning method (Desroziers and Ivanov, 2001).

**ONLY_SEA_RAD**

Logical, controls whether only assimilating radiance over water.

**TIME_WINDOW_MIN**

String, e.g., "2007-08-15_03:00:00.0000", start time of assimilation time window

**TIME_WINDOW_MAX**

String, e.g., "2007-08-15_09:00:00.0000", end time of assimilation time window

**USE_ANTCORR (30)**

Logical array with dimension RTMINIT_NSENSOR, controls if performing Antenna Correction in CRTM.

**USE_CLDDDET_MMR**

Logical, controls whether using the MMR scheme to conduct cloud detection for infrared radiance.

**USE_CLDDDET_ECMWF**

Logical, controls whether using the ECMWF scheme to conduct cloud detection for infrared radiance.

**AIRS_WARMEST_FOV**
Logical, controls whether using the observation brightness temperature for AIRS Window channel #914 as criterion for GSI thinning.

**USE_CRTM_KMATRIX**
Logical, controls whether using the CRTM K matrix rather than calling CRTM TL and AD routines for gradient calculation.

**USE_RTTOV_KMATRIX**
Logical, controls whether using the RTTOV K matrix rather than calling RTTOV TL and AD routines for gradient calculation.

**RTTOV_EMIS_ATLAS_IR**
Integer, controls the use of the IR emissivity atlas. Emissivity atlas data (should be downloaded separately from the RTTOV website) need to be copied or linked under a sub-directory of the working directory (emis_data) if RTTOV_EMIS_ATLAS_IR is set to 1.

**RTTOV_EMIS_ATLAS_MW**
Integer, controls the use of the MW emissivity atlas. Emissivity atlas data (should be downloaded separately from the RTTOV website) need to be copied or linked under a sub-directory of the working directory (emis_data) if RTTOV_EMIS_ATLAS_MW is set to 1 or 2.

---

g. Diagnostics and Monitoring

**Monitoring capability within WRFDA**

Run WRFDA with the rad_monitoring namelist parameter in record wrfvar14 in namelist.input.

0 means assimilating mode. Innovations (O minus B) are calculated and data are used in minimization.
1 means monitoring mode: innovations are calculated for diagnostics and monitoring. Data are not used in minimization.

The value of rad_monitoring should correspond to the value of rtminit_nsensor. If rad_monitoring is not set, then the default value of 0 will be used for all sensors.

**Outputting radiance diagnostics from WRFDA**

Run WRFDA with the following namelist options in record wrfvar14 in namelist.input.
**write_iv_rad_ascii**

Logical. TRUE to write out (observation-background, etc.) diagnostics information in plain-text files with the prefix ‘inv,’ followed by the instrument name and the processor id. For example, 01_inv_noaa-17-amsub.0000 (01 is outerloop index, 0000 is processor index)

**write_oa_rad_ascii**

Logical. TRUE to write out (observation-background, observation-analysis, etc.) diagnostics information in plain-text files with the prefix ‘oma,’ followed by the instrument name and the processor id. For example, 01_oma_noaa-18-mhs.0001

Each processor writes out the information for one instrument in one file in the WRFDA working directory.

**Radiance diagnostics data processing**

One of the 44 executables compiled as part of the WRFDA system is the file da_rad_diags.exe. This program can be used to collect the 01_inv* or 01_oma* files and write them out in netCDF format (one instrument in one file with prefix diags followed by the instrument name, analysis date, and the suffix .nc) for easier data viewing, handling and plotting with netCDF utilities and NCL scripts. See WRFDA/var/da/da_monitor/README for information on how to use this program.

**Radiance diagnostics plotting**

Two NCL scripts (available as part of the WRFDA Tools package, which can be downloaded at http://www2.mmm.ucar.edu/wrf/users/wrfda/download/tools.html) are used for plotting: $TOOLS_DIR/var/graphics/ncl/plot_rad_diags.ncl and $TOOLS_DIR/var/graphics/ncl/advance_cymdh.ncl. The NCL scripts can be run from a shell script, or run alone with an interactive ncl command (the NCL script and set the plot options must be edited, and the path of advance_cymdh.ncl, a date-advancing script loaded in the main NCL plotting script, may need to be modified).

Steps (3) and (4) can be done by running a single ksh script (also in the WRFDA Tools package: $TOOLS_DIR/var/scripts/da_rad_diags.ksh) with proper settings. In addition to the settings of directories and what instruments to plot, there are some useful plotting options, explained below.

```
setenv OUT_TYPE=ncgm
ncgm or pdf
pdf will be much slower than ncgm and generate huge output if plots are not split. But pdf has higher resolution than ncgm.

setenv PLOT_STATS_ONLY=false
true or false
true: only statistics of OMB/OMA vs channels and OMB/OMA vs dates will be plotted.
```
false: data coverage, scatter plots (before and after bias correction), histograms (before and after bias correction), and statistics will be plotted.

setenv PLOT_OPT=sea_only
setenv PLOT_QCED=false
setenv PLOT_HISTO=false
setenv PLOT_SCATT=true
setenv PLOT_EMISS=false
setenv PLOT_SPLIT=false
setenv PLOT_CLOUDY=false
setenv PLOT_CLOUDY_OPT=si
setenv CLWP_VALUE=0.2
setenv SI_VALUE=3.0

Evolution of VarBC parameters

NCL scripts (also in the WRFDA Tools package: $TOOLS_DIR/var/graphics/ncl/plot_rad_varbc_param.ncl and $TOOLS_DIR/var/graphics/ncl/advance_cymdh.ncl) are used for plotting the evolution of VarBC parameters.

Radar Data Assimilation in WRFDA

WRFDA has the ability to assimilate Doppler radar data, either for 3DVAR or 4DVAR assimilation. Both Doppler velocity and reflectivity can be assimilated, and there are several different reflectivity operator options available.

a. Preparing radar observations

Radar observations are read by WRFDA in a text-based format. This format is described in the radar tutorial presentation available on the WRFDA website (http://www2.mmm.ucar.edu/wrf/users/wrfda/Tutorials/2015_Aug/docs/WRFDA_Radar.pdf). Because
radar data comes in a variety of different formats, it is the user’s responsibility to convert their data into this format. For 3DVAR, these observations should be placed in a file named ob.radar. For 4DVAR, they should be placed in files named ob01.radar, ob02.radar, etc., with one observation file per time slot, as described in the earlier 4DVAR section.

b. Running WRFDA for radar assimilation

Once your observations are prepared, you can run WRFDA the same as you would normally (see the previous sections on how to run either 3DVAR or 4DVAR). For guidance, there is an example 3DVAR case available for download at http://www2.mmm.ucar.edu/wrf/users/wrfda/download/V38/wrfda_radar_testdata.tar.gz.

Edit namelist.input and pay special attention to the radar options under &wrfvar4:

```fortran
&wrfvar4
use_radarobs = .true. ; radar obs will be read by WRFDA
use_radar_rv = .true. ; Assimilate radar velocity observations
use_radar_rf = .true. ; Assimilate radar reflectivity using original reflectivity operator (total mixing ratio)
use_radar_rqv = .false. ; Assimilate radar reflectivity using estimated humidity from radar reflectivity
use_radar_rhv = .false. ; Assimilate radar reflectivity using rainwater and ice mixing ratios
use_3dvar_phy = .true. ; Partition hydrometeors via the moist explicit scheme (warm rain process)
/
&wrfvar7
cloud_cv_options = 3 ; Use hydrometeor control variables to get moisture increments
for use_radar_rhv = true
```

Starting with Version 3.7, WRFDA has an improved radar data assimilation capability: there are now two different options for assimilating radar reflectivity data. The first (use_radar_rf) directly assimilates the observed reflectivity using a reflectivity operator to convert the model rainwater mixing ratio into reflectivity and the total mixing ratio as the control variable, as described in Xiao and Sun, 2007 (http://journals.ametsoc.org/doi/full/10.1175/MWR3471.1); this is the only option available in previous versions of WRFDA. For this option, the hydrometeor partition using a warm rain scheme described in the above reference can be turned on (use_3dvar_phy = .true.). The second (use_radar_rhv) is a scheme described in Wang et al, 2013 (http://journals.ametsoc.org/doi/full/10.1175/JAMC-D-12-0120.1), which assimilates rainwater mixing ratio that is estimated from radar reflectivity, described as an “indirect method” in the paper. This second option also includes an option (use_radar_rqv) that allows the assimilation of in-cloud humidity estimated from reflectivity using a method described in Wang et al, 2013. It also includes the assimilation of snow and graupel converted from reflectivity using formulas as described in Gao and Stensrud, 2012 (http://journals.ametsoc.org/doi/abs/10.1175/JAS-D-11-0162.1).
c. CLOUD.CV option for 4DVAR

To take full advantage of the use_radar_rhv option for 4DVAR radar assimilation, you must compile WRFDA in a specific way. Prior to running the configure script, set the environment variable "CLOUD.CV=1". This will enable the extra moisture control variables for this option. Default values are used for the background error statistics for the extra moisture control variables, so no extra BE input is necessary. We do not recommend compiling with this option for other methods of data assimilation at this time, and it will not work with background error option cv_options=3.

Precipitation Data Assimilation in WRFDA 4D-Var

The assimilation of precipitation observations in WRFDA 4D-Var is described in this section. Currently, WRFPLUS has already included the adjoint and tangent linear codes of large-scale condensation and cumulus scheme, therefore precipitation data can be assimilated directly in 4D-Var. Users who are interested in the scientific detail of 4D-Var assimilation of precipitation should refer to related scientific papers, as this section is only a basic guide to running WRFDA Precipitation Assimilation. This section instructs users on data processing, namelist variable settings, and how to run WRFDA 4D-Var with precipitation observations.

a. Preparing precipitation observations for 4D-Var

WRFDA 4D-Var can assimilate NCEP Stage IV radar and gauge precipitation data. NCEP Stage IV archived data are available on the NCAR CODIAC web page at: http://data.eol.ucar.edu/codiac/dss/id=21.093 (for more information, please see the NCEP Stage IV Q&A Web page at http://www.emc.ncep.noaa.gov/mmb/ylin/pcpanl/QandA/). The original precipitation data are at 4-km resolution on a polar-stereographic grid. Hourly, 6-hourly and 24-hourly analyses are available. The following image shows the accumulated 6-h precipitation for the tutorial case.
It should be mentioned that the NCEP Stage IV archived data is in GRIB1 format and it cannot be ingested into the WRFDA directly. A tool “precip_converter” is provided to reformat GRIB1 observations into the WRFDA-readable ASCII format. It can be downloaded from the WRFDA users page at http://www2.mmm.ucar.edu/wrf/users/wrfda/download/precip_converter.tar.gz. The NCEP GRIB libraries, w3 and g2 are required to compile the precip_converter utility. These libraries are available for download from NCEP at http://www.nco.ncep.noaa.gov/pmb/codes/GRIB2/. The output file to the precip_converter utility is named in the format ob.rain.yyyymmddhh.xxh; The 'yyyymmddhh' in the file name is the ending hour of the accumulation period, and 'xx' (=01,06 or 24) is the accumulating time period.

For users wishing to use their own observations instead of NCEP Stage IV, it is the user’s responsibility to write a Fortran main program and call subroutine writerainobs (in write_rainobs.f90) to generate their own precipitation data. For more information please refer to the README file in the precip_converter directory.

b. Running WRFDA 4D-Var with precipitation observations

WRFDA 4D-Var is able to assimilate hourly, 3-hourly and 6-hourly precipitation data. According to experiments and related scientific papers, 6-hour precipitation accumulations are the ideal observations to be assimilated, as this leads to better results than directly assimilating hourly data.

The tutorial example is for assimilating 6-hour accumulated precipitation. In your working directory, link all the necessary files as follows,

```bash
> ln -fs $WRFDA_DIR/var/da/da_wrfvar.exe .
> ln -fs $DAT_DIR/rc/2008020512/wrfinput_d01 .
> ln -fs $DAT_DIR/rc/2008020512/wrfbdy_d01 .
> ln -fs $DAT_DIR/rc/2008020512/wrfinput_d01 fg02 (only necessary for var4d_lbc=true)
```
WRFDA

> ln -fs wrfinput_d01 fg
> ln -fs $DAT_DIR/be/be.dat.
> ln -fs $WRFDA_DIR/run/LANDUSE.TBL.
> ln -fs $WRFDA_DIR/run/RRTM_DATA DBL./RRTM_DATA
> ln -fs $DAT_DIR/ob/2008020518/ob.rain.2008020518.06h ob07.rain

Note: The reason why the observation ob.rain.2008020518.06h is linked as ob07.rain will be explained in section d.

Edit namelist.input (you can start with the same namelist as for the 4dvar tutorial case) and pay special attention to &wrfvar1 and &wrfvar4 for precipitation-related options.

```
&wrfvar1
  var4d=true,
  var4d_lbc=true,
  var4d_bin=3600,
  var4d_bin_rain=21600,
  /   
/.....
&wrfvar4
  use_rainobs=true,
  thin_rainobs=true,
  thin_mesh_conv=28*20.,
  /   
```

Then, run 4D-Var in serial or parallel mode,

```bash
>./da_wrfvar.exe >& wrfda.log
```

c. Namelist variables to control precipitation assimilation

**var4d_bin_rain**
- Precipitation observation sub-window length for 4D-Var. It does not need to be consistent with var4d_bin.

**thin_rainobs**
- Logical, TRUE will perform thinning on precipitation data.

**thin_mesh_conv**
- Specify thinning mesh size (in km)

d. Properly linking observation files

In section b, ob.rain.2008020518.06h is linked as ob07.rain. The number 07 is assigned according to the following rule:

\[
x = i \times \frac{\text{var4d_bin_rain}}{\text{var4d_bin}} + 1,
\]

Here, i is the sequence number of the observation. For \( x < 10 \), the observation file should be renamed as ob0x.rain;
for $x \geq 10$, it should be renamed as $obx.rain$

In the example above, 6-hour accumulated precipitation data is assimilated in 6-hour time window. In the namelist, values should be set at $var4d\_bin=3600$ and $var4d\_bin\_rain=21600$, and there is one observation file (i.e., $i=1$) in the time window. Thus the value of $x$ is 7. The file $ob.rain.2008020518.06h$ should be renamed as $ob07.rain$.

Let us take another example for how to rename observation files for 3-hourly precipitation data in 6-hour time window. The sample namelist is as follows,

```
&wrfvar1
  var4d=true,
  var4d\_lbc=true,
  var4d\_bin=3600,
  var4d\_bin\_rain=10800,
  ....
/
```

There are two observation files, $ob.rain.2008020515.03h$ and $ob.rain.2008020518.03h$. For the first file ($i=1$) $ob.rain.2008020515.03h$, it should be renamed as $ob04.rain$, and the second file ($i=2$) renamed as $ob07.rain$.

### Updating WRF Boundary Conditions

#### a. Lateral boundary conditions

When using WRFDA output to run a WRF forecast, it is essential that you update the WRF lateral boundary conditions (contained in the file $wrfbdy\_01$, created by $real\_exe$) to match your new analysis. Domain-1 ($wrfbdy\_d01$) must be updated to be consistent with the new WRFDA initial condition (analysis). This is absolutely essential. For nested domains, domain-2, domain-3, etc., the lateral boundary conditions are provided by their parent domains, so no lateral boundary update is needed for these domains. The update procedure is performed by the WRFDA utility called $da\_update\_bc\_exe$, and after compilation can be found in $\$WRFDA\_DIR/var/build$.

$da\_update\_bc\_exe$ requires three input files: the WRFDA analysis ($wrfvar\_output$), the $wrfbdy$ file from $real\_exe$, and a namelist file: param.e.in. To run $da\_update\_bc\_exe$ to update lateral boundary conditions, follow the steps below:

```
> cd $WRFDA\_DIR/var/test/update\_bc
> cp –p $DAT\_DIR/rc/2008020512/wrfbdy\_d01 .
(IMPORTANT: make a copy of $wrfbdy\_d01$, as the $wrf\_bdy\_file$ will be overwritten by $da\_update\_bc\_exe$)
```
< vi param
&control_param
da_file = './tutorial/wrfvar_output'
wrf_bdy_file = './wrfbdy_d01'
domain_id = 1
debug = .true.
update_lateral_bdy = .true.
update_low_bdy = .true.
update_lsm = .false.
iswater = 16
var4d_lbc = .false.
/

> ln -sf $WRFDA_DIR/var/da/da_update_bc.exe .
> ./da_update_bc.exe

At this stage, you should have the files wrfvar_output and wrfbdy_d01 in your WRFDA working directory. They are the WRFDA updated initial and boundary condition files for any subsequent WRF model runs. To use, link a copy of wrfvar_output and wrfbdy_d01 to wrfinput_d01 and wrfbdy_d01, respectively, in your WRF working directory.

You should also see two additional output files: fort.11 and fort.12. These contain information about the changes made to wrfbdy_d01.

b. Cycling with WRF and WRFDA (warm-start)

In cycling mode (warm-start), the lower boundary in the first guess file also needs to be updated based on the information from the wrfinput file, generated by WPS/real.exe at analysis time. If in cycling mode (especially if you are doing radiance data assimilation and there are SEA ICE or SNOW in your domain), it is recommended that before you run WRFDA, you run da_update_bc.exe with the following namelist options:

da_file = './fg'
wrf_input = './wrfinput_d01'
update_lateral_bdy = .false.
update_low_bdy = .true.
iswater = 16

Note: “iswater” (water point index) is 16 for USGS LANDUSE and 17 for MODIS LANDUSE.

This creates a lower-boundary updated first guess (da_file will be overwritten by da_update_bc with updated lower boundary conditions from wrf_input). Then, after WRFDA has finished, run da_update_bc.exe again with the following namelist options:

da_file = './wrfvar_output'
wrf_bdy_file = './wrfbdy_d01'
update_lateral_bdy = .true.
update_low_bdy = .false.

This updates the lateral boundary conditions (wrf_bdy_file will be overwritten by da_update_bc with lateral boundary conditions from da_file).
As mentioned previously, lateral boundary conditions for child domains (wrfinput_02, wrfinput_03, etc.) come from the respective parent domains, so update_bc is not necessary after running WRFDA. However, in a cycling procedure, the lower boundaries in each of the nested domains’ WRFDA analysis files still need to be updated. In these cases, you must set the namelist variable, domain_id > 1 (default is 1 for domain-1) and provide the appropriate wrfinput file (wrf_input = './wrfinput_d02' for domain 2, for instance).

c. WRFDA 4DVAR with lateral boundary conditions as control variables

If you activate the var4d_lbc option in a WRF 4D-Var run, in addition to the above-mentioned files you will also need the ana02 file from the WRFDA working directory. In param.in, set var4d_lbc to TRUE and use “da_file_02” to point to the location of the ana02 file.

da_file_02 = './ana02'
var4d_lbc = .true.

Background Error and running GEN_BE

Quick-start guide: running WRFDA with different BE options

To run WRFDA with the generic CV3 option, simply link the provided be.dat file in the var/run directory:

   > cp –p $WRFDA_DIR/var/run/be.dat.cv3 $WORK_DIR/be.dat

To run WRFDA with any other option, you will have to run GEN_BE first. GEN_BE takes a series of forecasts initialized at different times, and compares the forecasts that are valid at the same time (e.g., compare a 24-hour forecast initialized at 00Z with a 12-hour forecast initialized at 12Z) to get an estimate of the background error statistics.

You will use the wrapper script gen_be_wrapper.ksh to run GEN_BE. For instructions on how to set up your experiment to run GEN_BE, reference the test case as described in the section “Domain-specific background error options: Running GEN_BE” below.

Background error options in WRFDA

Users have four choices to define the background error covariance (BE). We call them CV3, CV5, CV6, and CV7. Each of these has different properties, which are outlined in the table below:

<table>
<thead>
<tr>
<th>CV option</th>
<th>Data source</th>
<th>Control variables</th>
<th>cv_options =</th>
</tr>
</thead>
</table>

With CV3, the control variables are in physical space while with CV5, CV6, and CV7, the control variables are in eigenvector space. The major difference between these two kinds of BE is the vertical covariance; CV3 uses the vertical recursive filter to model the vertical covariance but the others use an empirical orthogonal function (EOF) to represent the vertical covariance. The recursive filters to model the horizontal covariance are also different with these BEs. We have not conducted the systematic comparison of the analyses based on these BEs. However, CV3 (a BE file provided with our WRFDA system) is a global BE and can be used for any regional domain, while CV5, CV6, and CV7 BE’s are domain-dependent, and so should be generated based on forecast or ensemble data from the same domain.

As summarized in the above table, CV5, CV6, and CV7 differ in the control variables they use. CV5 utilizes streamfunction ($\psi$), unbalanced velocity potential ($\chi_u$), unbalanced temperature ($T_u$), pseudo relative humidity ($RH_s$), and unbalanced surface pressure ($P_{s,u}$). The pseudo relative humidity is defined as $Q/Q_{b,s}$, where $Q_{b,s}$ is the saturated specific humidity from the background field. For CV6 the moisture control variable is the unbalanced portion of the pseudo relative humidity ($RH_{s,u}$). Additionally, CV6 introduces six additional correlation coefficients in the definition of the balanced part of analysis control variables. See the section GEN_BE for CV6 for more details on this option. Finally, CV7 uses a different set of control variables: u, v, temperature, pseudo relative humidity ($RH_s$), and surface pressure ($P_s$).

**Generic BE option: CV3**

CV3 is the NCEP background error covariance. It is estimated in grid space by what has become known as the NMC method (Parrish and Derber 1992). The statistics are estimated with the differences of 24 and 48-hour GFS forecasts with T170 resolution, valid at the same time for 357 cases, distributed over a period of one year. Both the amplitudes and the scales of the background error have to be tuned to represent the forecast error in the estimated fields. The statistics that project multivariate relations among variables are also derived from the NMC method.

The variance of each variable, and the variance of its second derivative, are used to estimate its horizontal scales. For example, the horizontal scales of the stream function can be estimated from the variance of the vorticity and stream function.

The vertical scales are estimated with the vertical correlation of each variable. A table is built to cover the range of vertical scales for the variables. The table is then used to find the scales in vertical grid units. The filter profile and the vertical correlation are fitted lo-
cally. The scale of the best fit from the table is assigned as the scale of the variable at that vertical level for each latitude. Note that the vertical scales are locally defined so that the negative correlation further away, in the vertical direction, is not included.

Theoretically, CV3 BE is a generic background error statistics file which can be used for any case. It is quite straightforward to use CV3 in your own case. To use the CV3 BE file in your case, set cv_options=3 in &wrfvar7 in namelist.input in your working directory, and use the be.dat is located in WRFDA/var/run/be.dat.cv3.

Modifying CV3 length scales and variance
Because CV3 is a generic background error option, it is necessary to tune the default background error scale length and variance values for each experiment. These can be controlled at run time with a series of namelist variables described below.

The scaling factors for CV3 are stored as an array of values for each individual control variable:

- as1: stream function
- as2: unbalanced velocity potential
- as3: unbalanced temperature
- as4: pseudo relative humidity
- as5: unbalanced surface pressure

These variables are all 3-element vectors. The first element is the variance scaling factor. The second is the horizontal length scale factor. The third is the vertical length scale factor. So setting the default values in your namelist would look like this:

```fortran
&wrfvar7
cv_options = 3,
as1        = 0.25, 1.00, 1.50,
as2        = 0.25, 1.00, 1.50,
as3        = 0.25, 1.00, 1.50,
as4        = 0.25, 1.00, 1.50,
as5        = 0.25, 1.00, 1.50,
/
```

The first column is the variance, the second is the horizontal length scale factor, and the third is the vertical length scale factor.

For multiple outer loops, the next 3 elements of each vector must be filled in. So for 2 outer loops (max_ext_its=2), to use the default values, the namelist should look like this:

```fortran
&wrfvar7
cv_options = 3,
as1        = 0.25, 1.00, 1.50, 0.25, 1.00, 1.50,
as2        = 0.25, 1.00, 1.50, 0.25, 1.00, 1.50,
as3        = 0.25, 1.00, 1.50, 0.25, 1.00, 1.50,
as4        = 0.25, 1.00, 1.50, 0.25, 1.00, 1.50,
as5        = 0.25, 1.00, 1.50, 0.25, 1.00, 1.50,
/
```

Again, the first column is the variance, the second is the horizontal length scale factor, and the third is the vertical length scale factor for the first outer loop. The fourth column is the variance, the fifth is the horizontal length scale factor, and the sixth is the vertical length scale factor for the second outer loop.

Continue in this manner for more outer loops. The values listed above are the default values, but can be adjusted for each individual experiment.

**Domain-specific background error options: Running GEN_BE**

To use CV5, CV6 or CV7 background error covariance, it is necessary to generate your domain-specific background error statistics with the `gen_be` utility. The default CV3 background error statistics file, supplied with the WRFDA source code, can NOT be used with these control variable options.

The main source code for the various `gen_be` stages can be found in `WRFDA/var/gen_be`. The executables of `gen_be` should have been created when you compiled the WRFDA code (as described earlier). The scripts to run these codes are in `WRFDA/var/scripts/gen_be`. The user can run `gen_be` using the wrapper script `WRFDA/var/scripts/gen_be/gen_be_wrapper.ksh`.

The input data for `gen_be` are WRF forecasts, which are used to generate model perturbations, used as a proxy for estimates of forecast error. For the NMC-method, the model perturbations are differences between forecasts (e.g. T+24 minus T+12 is typical for regional applications, T+48 minus T+24 for global) valid at the same time. Climatological estimates of background error may then be obtained by averaging these forecast differences over a period of time (e.g. one month). Given input from an ensemble prediction system (EPS), the inputs are the ensemble forecasts, and the model perturbations created are the transformed ensemble perturbations. The `gen_be` code has been designed to work with either forecast difference or ensemble-based perturbations. The former is illustrated in this tutorial example.

It is important to include forecast differences valid at different parts of the day (for example, forecasts valid at 00Z and 12Z through the forecast period) to remove contributions from the diurnal cycle (i.e. do not run `gen_be` using model perturbations valid for a single time each day).

The inputs to `gen_be` are netCDF WRF forecast output ("wrfout") files at specified forecast ranges. To avoid unnecessary large single data files, it is assumed that all forecast ranges are output to separate files. For example, if we wish to calculate BE statistics using the NMC-method with (T+24)-(T+12) forecast differences (default for regional) then by setting the WRF namelist.input options `history_interval=720`, and `frames_per_outfile=1` we get the necessary output datasets. Then the forecast output files should be arranged as follows: directory name is the forecast initial time, time info in the file name is the forecast valid time. 2008020512/wrfout_d01_2008-02-
06_00:00:00 means a 12-hour forecast valid at 2008020600, initialized at 2008020512.

Example dataset for a test case (90 x 60 x 41 gridpoints) can be downloaded from http://www2.mmm.ucar.edu/wrf/users/wrfda/download/testdata.html. Untar the gen_be_forecasts_20080205.tar.gz file. You will have:

```
$FC_DIR
-rw-r--r--  1 users  11556492 2008020512/wrfout_d01_2008-02-06-00:00:00
-rw-r--r--  1 users  11556492 2008020512/wrfout_d01_2008-02-06-12:00:00
-rw-r--r--  1 users  11556492 2008020600/wrfout_d01_2008-02-06-12:00:00
-rw-r--r--  1 users  11556492 2008020600/wrfout_d01_2008-02-07-00:00:00
-rw-r--r--  1 users  11556492 2008020612/wrfout_d01_2008-02-07-12:00:00
```

In the above example, only 1 day (12Z 05 Feb to 12Z 06 Feb. 2008) of forecasts, every 12 hours is supplied to gen_be_wrapper to estimate forecast error covariance. It is only for demonstration. The minimum number of forecasts required depends on the application, number of grid points, etc. Month-long (or longer) datasets are typical for the NMC-method. Generally, at least a 1-month dataset should be used.

Under WRFDA/var/scripts/gen_be, gen_be_wrapper.ksh is used to generate the BE data. The following variables need to be set to fit your case:

- `export WRFVAR_DIR=/glade/p/work/wrfhelp/PRE_COMPILED_CODE/WRFDA`
- `export NL_CV_OPTIONS=5`  # 5 for CV5, 7 for CV7
- `export START_DATE=2008020612`  # the first perturbation valid date
- `export END_DATE=2008020700`  # the last perturbation valid date
- `export NUM_LEVELS=40`  # e_vert - 1
- `export BIN_TYPE=5`  # How data is binned for calculating statistics
- `export FC_DIR=/glade/p/work/wrfhelp/WRFDA_DATA/fc`  # where wrf forecasts are
- `export RUN_DIR=`pwd`/gen_be`  # Where GEN_BE will run and output files

**Note:** The START_DATE and END_DATE are perturbation valid dates. As shown in the forecast list above, when you have 24-hour and 12-hour forecasts initialized at 2008020512, through 2008020612, the first and final forecast difference valid dates are 2008020612 and 2008020700, respectively.

**Note:** The forecast dataset should be located in $FC_DIR. Then type:

```
$ gen_be_wrapper.ksh
```

Once the gen_be_wrapper.ksh run is completed, the be.dat can be found under the $RUN_DIR directory.

To get a clear idea about what is included in be.dat, the script gen_be_plot_wrapper.ksh may be used. This plots various data in be.dat; for example:
GEN_BE for CV6

CV6 is a multivariate background error statistics option in WRFDA. It may be activated by setting the namelist variable `cv_options=6`. This option introduces six additional correlation coefficients in the definition of the balanced part of analysis control variables. Thus with this implementation, moisture analysis is multivariate, in the sense that temperature and wind may lead to moisture increments, and vice-versa. Further details may be seen at: [http://www2.mmm.ucar.edu/wrf/users/wrfda/Docs/WRFDA_updated_for_cv6.pdf](http://www2.mmm.ucar.edu/wrf/users/wrfda/Docs/WRFDA_updated_for_cv6.pdf)

How to generate CV6 background error statistics for WRFDA

CV6 background error statistics for WRFDA are generated by executing a top-level script, `gen_be/wrapper_gen_mbe.ksh`, residing under `SCRIPTS_DIR` via a suitable wrapper script. The rest of the procedure remains the same as with normal running of the `gen_be` utility. A successful run will create a `be.dat` file in the `RUN_DIR` directory.
How to run WRFDA with CV6 background error statistics

After successfully generating the CV6 background error statistics file `be.dat`, the procedure for running WRFDA is straightforward: Include `cv_options=6` in the `namelist.input` file under the `&wrfvar7` list of namelist options.

How to tune CV6 background error statistics

Below is a list of nine tuning parameters available in WRFDA; these can be specified under `&wrfvar7` in the namelist. Default values for these variables are set as “1.0”. Setting corresponding values > 1.0 (< 1.0) will increase (decrease) the corresponding contributions:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>psi_ch_factor</td>
<td>Parameter to control contribution of stream function in defining balanced part of velocity potential</td>
</tr>
<tr>
<td>psi_t_factor</td>
<td>Parameter to control contribution of stream function in defining balanced part of temperature</td>
</tr>
<tr>
<td>psi_ps_factor</td>
<td>Parameter to control contribution of stream function in defining balanced part of surface pressure</td>
</tr>
<tr>
<td>psi_rh_factor</td>
<td>Parameter to control contribution of stream function in defining balanced part of moisture</td>
</tr>
<tr>
<td>chi_u_t_factor</td>
<td>Parameter to control contribution of unbalanced part of velocity potential in defining balanced part of temperature</td>
</tr>
<tr>
<td>chi_u_ps_factor</td>
<td>Parameter to control contribution of unbalanced part of velocity potential in defining balanced part of surface pressure</td>
</tr>
<tr>
<td>chi_u_rh_factor</td>
<td>Parameter to control contribution of unbalanced part of velocity potential in defining balanced part of moisture</td>
</tr>
<tr>
<td>t_u_rh_factor</td>
<td>Parameter to control contribution of unbalanced part of temperature in defining balanced part of moisture</td>
</tr>
<tr>
<td>ps_u_rh_factor</td>
<td>Parameter to control contribution of unbalanced part of surface pressure in defining balanced part of moisture</td>
</tr>
</tbody>
</table>

Additional WRFDA Exercises:

a. Single Observation response in WRFDA:

With the single observation test, you may get some ideas of how the background and observation error statistics work in the model variable space. A single observation test is done in WRFDA by setting `num_pseudo=1`, along with other pre-specified values in record `&wrfvar15` and `&wrfvar19` of `namelist.input`. 
With the settings shown below, WRFDA generates a single observation with a pre-specified *innovation* (Observation – First Guess) value at the desired location; e.g. at (in terms of grid coordinate) 23x23, level 14 for “U” observation with error characteristics 1 m/s, and innovation size = 1.0 m/s.

```wrf
&wrfvar15
num_pseudo = 1,
pseudo_x = 23.0,
pseudo_y = 23.0,
pseudo_z = 14.0,
pseudo_err = 1.0,
pseudo_val = 1.0,
/
&wrfvar19
pseudo_var = "u", (Note: pseudo_var can be u, v, t, p, q. If pseudo_var is q, then the reasonable values of pseudo_err and pseudo_val are 0.001)
/
```

Note: You may wish to repeat this exercise for other observations, like temperature “t”, pressure “p”, specific humidity “q”, and so on.

**b. Response of BE length scaling parameter:**

Run the single observation test with the following additional parameters in record `&wrfvar7` of `namelist.input`.

```wrf
&wrfvar7
len_scaling1 = 0.5, # reduce psi length scale by 50%
len_scaling2 = 0.5, # reduce chi_u length scale by 50%
len_scaling3 = 0.5, # reduce T length scale by 50%
len_scaling4 = 0.5, # reduce q length scale by 50%
len_scaling5 = 0.5, # reduce Ps length scale by 50%
/
```

Note: You may wish to try the response of an individual variable by setting one parameter at a time. Note the spread of analysis increment.

**c. Response of changing BE variance:**

Run the single observation test with the following additional parameters in record `&wrfvar7` of `namelist.input`.

```wrf
&wrfvar7
var_scaling1 = 0.25, # reduce psi variance by 75%
var_scaling2 = 0.25, # reduce chi_u variance by 75%
var_scaling3 = 0.25, # reduce T variance by 75%
var_scaling4 = 0.25, # reduce q variance by 75%
var_scaling5 = 0.25, # reduce Ps variance by 75%
/
```
Note: You may wish to try the response of individual variable by setting one parameter at a time. Note the magnitude of analysis increments.

d. Response of convergence criteria:

Run the tutorial case with

```
&wrfvar6
  eps = 0.0001,
/
```

You may wish to compare various diagnostics with an earlier run.

e. Response of outer loop on minimization:

Run the tutorial case with

```
&wrfvar6
  max_ext_its = 2,
/
```

With this setting, the “outer loop” for the minimization procedure will be activated. You may wish to compare various diagnostics with an earlier run.

Note that when running multiple outer loops with the CV3 background error option, you must specify the scaling factors which are called as1, as2, as3, as4, and as5. More details can be found in the section “Modifying CV3 length scales and variance”.

f. Response of suppressing particular types of data in WRFDA:

The types of observations that WRFDA gets to use actually depend on what is included in the observation file and the WRFDA namelist settings. For example, if you have SYNOP data in the observation file, you can suppress its usage in WRFDA by setting use_synopobs=false in record &wrfvar4 of namelist.input. It is OK if there are no SYNOP data in the observation file and use_synopobs=true.

Turning on and off certain types of observations is widely used for assessing the impact of observations on data assimilations.

Note: It is important to go through the default “use_*” settings in record &wrfvar4 in WRFDA/Registry/registry.var to know what observations are activated in default.

g. Utilizing wind speed/direction assimilation:

If observations containing wind speed/direction information are provided to WRFDA, you can assimilate these observations directly, rather than converting the wind to its u- and v-components prior to assimilation.
Wind speed/direction assimilation is controlled by the following namelist options:

```
&wrfvar2
wind_sd       true: all wind values which are reported as speed/direction will be assimilated as such
               false: (default behavior) all wind obs are converted to u/v prior to assimilation
wind_stats_sd Assimilate wind in u/v format, but output speed/direction statistics
```

The following settings only matter if `check_max_iv=true` (if innovation is greater than observation error times the error factor listed below, the observation will be rejected):

```
&wrfvar2
qc_rej_both  true: if either u or v (spd or dir) do not pass quality control, both obs are rejected
               false: (default behavior) qc on wind obs is handled individually
&wrfvar5
max_omb_spd   Max absolute value of innovation for wind speed obs in m/s; greater than this and the innovation will be set to zero (default: 100.0)
max_omb_dir   Max absolute value of innovation for wind direction obs in degrees; greater than this and the innovation will be set to zero (default: 1000.0)
max_error_spd Speed error factor (default: 5.0)
max_error_dir Direction error factor (default: 5.0)
```

The assimilation of wind speed/direction can also be controlled by observation type, using the following variables (note: setting `wind_sd = .true.` as above will override these individual settings):

```
&wrfvar2
wind_sd_airep Aircraft reports
wind_sd_buoy Buoy reports
wind_sd_geoamv Geostationary satellite atmospheric motion vectors
wind_sd_metar METAR reports
wind_sd_mtgirs Meteosat Third Generation
wind_sd_pilot Pilot reports
wind_sd_polaramv Polar satellite atmospheric motion vectors
wind_sd_profiler Wind profiler reports
wind_sd_qscat QuikScat reports
wind_sd_ships Ship reports
wind_sd_sound Sounding reports
wind_sd_synop Synoptic reports
wind_sd_tamdar TAMDAR reports
```

```true: wind values which are reported as speed/direction will be assimilated as such
false: (default behavior) all wind obs are converted to u/v prior to assimilation```

Further details about this method can be found in the following publications:


h. Utilizing the Weak Penalty Constraint option:

For Version 3.8, a new “weak penalty constraint” (WPEC) option has been added to WRFDA which aims to enforce quasi-gradient balance on a WRFDA analysis. It was designed with the specific aim of improving assimilation of radar data within tropical cyclones, but may be useful for other weather phenomena of similar scales. It can be used with 3DVAR or hybrid 3DVAR (4DVAR is not compatible with this new capability).

Further details about this method can be found in the following publication:


This new option is controlled by the following set of namelist options:

```
&wrfvar12
use_wpec    true: enables the constraint term
             false: (default behavior) disables the constraint term
wpec_factor The constraint’s weighting factor (1/Γ) as described in the paper
balance_type 1 = geostrophic term
              2 = cyclostrophic term
              3 = geostrophic + cyclostrophic terms (default; recommended)
```

WRFDA Diagnostics

WRFDA produces a number of diagnostic files that contain useful information on how the data assimilation has performed. This section will introduce you to some of these files, and what to look for.

After running WRFDA, it is important to check a number of output files to see if the assimilation appears sensible. The WRFDA package, which includes several useful scripts, may be downloaded from [http://www2.mmm.ucar.edu/wrf/users/wrfda/download/tools.html](http://www2.mmm.ucar.edu/wrf/users/wrfda/download/tools.html)

The content of some useful diagnostic files are as follows:

`cost_fn` and `grad_fn`: These files hold (in ASCII format) WRFDA cost and gradient function values, respectively, for the first and last iterations. If you run with `PRINT_DETAIL_GRAD=true`, however, these values will be listed for each iteration; this
can be helpful for visualization purposes. The NCL script in the WRFDA TOOLS package located at var/graphics/ncl/plot_cost_grad_fn.ncl may be used to plot the content of cost_fn and grad_fn, if these files are generated with PRINT_DETAIL_GRAD=true.

**Note:** Make sure that you remove the first two lines (header) in cost_fn and grad_fn before you plot. You also need to specify the directory name for these two files.

gts_omb_oma_01: It contains (in ASCII format) information on all of the observations used by the WRFDA run. Each observation has its observed value, quality flag, observation error, observation minus background (OMB), and observation minus analysis (OMA). This information is very useful for both analysis and forecast verification purposes.

namelist.input: This is the WRFDA input namelist file, which contains all the user-defined non-default options. Any namelist-defined options that do not appear in this file should have their names checked against the values in $WRFDA_DIR/Registry/registry.var.

namelist.output: A list of all the namelist options used. If an option was not specified in namelist.input, the default listed in the registry value will be used.

rsl*: Files containing information for standard WRFDA output from individual processors when multiple processors are used. It contains a host of information on a number of observations, minimization, timings, etc. Additional diagnostics may be printed in these files by including various “print” WRFDA namelist options. To learn more about these additional “print” options, search for the “print_” string in $WRFDA_DIR/Registry/registry.var.

statistics: Text file containing OMB (OI) and OMA (OA) statistics (minimum, maximum, mean and standard deviation) for each observation type and variable. This information is very useful in diagnosing how WRFDA has used different components of the
observing system. Also contained are the analysis minus background (A-B) statistics, i.e. statistics of the analysis increments for each model variable at each model level. This information is very useful in checking the range of analysis increment values found in the analysis, and where they are in the WRF-model grid space.

The WRFDA analysis file is `wrfvar_output`. It is in WRF (netCDF) format. It will become the input file `wrfinp_d01` of any subsequent WRF run after lateral boundary and/or lower boundary conditions are updated by another WRFDA utility (See the section [Updating WRF boundary conditions](#)).

An NCL script, `$TOOLS_DIR/var/graphics/ncl/WRF-Var_plot.ncl`, is provided in the tools package for plotting. You need to specify the `analysis_file` name, its full path, etc. Please see the in-line comments in the script for details.

As an example, if you are aiming to display the U-component of the analysis at level 18, use the script `WRF-Var_plot.ncl`, and make sure the following pieces of codes are uncommented:

```ncl
var = "U"
fg = first_guess->U
an = analysis->U
plot_data = an
```

When you execute the following command from `$WRFDA_DIR/var/graphics/ncl`.

```
> ncl WRF-Var_plot.ncl
```

The plot should look like:

![U(18) m s⁻¹](image)

You may change the variable name, level, etc. in this script to display the variable of your choice at the desired eta level.

Take time to look through the text output files to ensure you understand how WRFDA works. For example:
How closely has WRFDA fit individual observation types? Look at the statistics file to compare the O-B and O-A statistics.

How big are the analysis increments? Again, look in the statistics file to see minimum/maximum values of A-B for each variable at various levels. It will give you a feel for the impact of the input observation data you assimilated via WRFDA by modifying the input analysis first guess.

How long did WRFDA take to converge? Does it really converge? You will get the answers of all these questions by looking into the rsl.*-files, as it indicates the number of iterations taken by WRFDA to converge. If this is the same as the maximum number of iterations specified in the namelist (NTMAX), or its default value (=200) set in $WRFDA_DIR/Registry/registry.var, then it means that the analysis solution did not converge. If this is the case, you may need to increase the value of “NTMAX” and rerun your case to ensure that the convergence is achieved. On the other hand, a normal WRFDA run should usually converge within 100 iterations. If it still doesn’t converge in 200 iterations, that means there may be a problem in the observations or first guess.

A good way to visualize the impact of assimilation of observations is to plot the analysis increments (i.e. analysis minus the first guess difference). Many different graphics packages (e.g. RIP4, NCL, GRADS etc) can do this.

You need to modify this script to fix the full path for first_guess and analysis files. You may also use it to modify the display level by setting kl and the name of the variable to display by setting var. Further details are given in this script.

If you are aiming to display the increment of potential temperature at level 18, after modifying $WRFDA_DIR/var/graphics/ncl/WRF-Var_plot.ncl, make sure the following pieces of code are uncommented:

```ncl
var = "T"
f = first_guess->T ;Theta- 300
an = analysis->T     ;Theta- 300
plot_data = an - fg
```

When you execute the following command from WRFDA_DIR/var/graphics/ncl.

```
> ncl WRF-Var_plot.ncl
```

The plot created will look as follows:
Generating ensembles with RANDOMCV

In addition to the variational methods previously mentioned, the WRFDA system supports both ensemble and hybrid ensemble/variational assimilation methods. To utilize these methods, having an ensemble of forecasts is necessary. WRFDA has a built-in method for generating ensemble initial conditions known as RANDOMCV. RANDOMCV works by adding random noise to the analysis in control variable space.

a. Running WRFDA for RANDOMCV

RANDOMCV is a capability of the main WRFDA executable, so you will run it by setting the following variables in namelist.input and then running da_wrfvar.exe as you would with any data assimilation run:

```plaintext
&wrfvar5
put_rand_seed true: enter your own seed numbers to generate random background perturbations. The advantage of this setting is that the same seed numbers will always produce the same perturbation false: (default behavior) perturbations will be generated “randomly” and give new perturbations each time

&wrfvar11
seed_array1 First integer for seeding the random function (default: 1)

&wrfvar12
seed_array2 Second integer for seeding the random function (default: 1) It is not necessary to change both seeds to get different perturbations

&wrfvar17
analysis_type Set this to ‘RANDOMCV’ to use the RANDOMCV capability
```

When setting your own random seeds, a common good practice is to set the first seed as the experiment date/time in integer form, and the second seed as the ensemble member...
number. *The seeds should not be set to zero.*

Because the perturbations are made in control variable space, the general pattern of perturbations will depend on your background error. You should be able to use any background error option with RANDOMCV (CV3, CV5, CV6, or CV7). Additionally, this means you can control their magnitude and lengthscales by modifying the background error variance and length scaling variables respectively:

For CV5, CV6, or CV7

```
&wrfvar
var_scaling1 = 0.25,  # reduce psi perturbation magnitude by 75%
var_scaling2 = 0.25,  # reduce chi_u perturbation magnitude by 75%
var_scaling3 = 0.0,   # reduce T perturbation by 100% (there will be no T perturbation!)
var_scaling4 = 2.0,   # increase q perturbation by 100%
var_scaling5 = 1.0,   # Keep Ps perturbation magnitude the same
var_scaling1 = 0.5,   # reduce psi perturbation length scale by 50%
var_scaling2 = 0.5,   # reduce chi_u perturbation length scale by 50%
var_scaling3 = 1.0,   # Keep T perturbation length scale the same
var_scaling4 = 2.0,   # increase q perturbation length scale by 100%
var_scaling5 = 1.5,   # increase Ps perturbation length scale by 50%
```

For CV3, see the section “Modifying CV3 length scales and variance”.

Hybrid Data Assimilation in WRFDA

The WRFDA system also includes a hybrid data assimilation technique, which is based on the existing 3D-Var. The difference between hybrid and 3D-Var schemes is that 3D-Var relies solely on a static covariance model to specify the background errors, while the hybrid system uses a combination of 3D-Var static error covariances and ensemble-estimated error covariances to incorporate a flow-dependent estimate of the background error statistics. The following section will give a brief introduction of some aspects of using the hybrid system. Please refer to these papers for a detailed description of the methodology used in the WRFDA hybrid system:


b. Source code
Four executables are used in the hybrid system. If you have successfully compiled the WRFDA system, the following executables will exist in the WRFDA/var/build directory:

- WRFDA/var/build/gen_be_ensmean.exe
- WRFDA/var/build/gen_be_ep2.exe
- WRFDA/var/build/da_wrfvar.exe
- WRFDA/var/build/gen_be_vertloc.exe

`gen_be_ensmean.exe` is used to calculate the ensemble mean, while `gen_be_ep2.exe` is used to calculate the ensemble perturbations. `gen_be_vertloc.exe` is used for vertical localization. As with 3D-Var/4D-Var, `da_wrfvar.exe` is the main WRFDA program. However, in this case, `da_wrfvar.exe` will run in hybrid mode.

c. Running the hybrid system

The procedure is the same as running 3D-Var/4D-Var, with the exception of some extra input files and namelist settings. The basic input files for WRFDA are LANDUSE.TBL, ob.ascii or ob.bufr (depending on which observation format you use), and be.dat (static background errors). Additional input files required by the hybrid are a single ensemble mean file (used as the fg for the hybrid application) and a set of ensemble perturbation files (used to represent flow-dependent background errors).

A set of initial ensemble members must be prepared before the hybrid application can be started. The ensemble can be obtained from other ensemble model outputs, or you can generate them yourself. This can be done, for example, by adding random noise to the initial conditions at a previous time and integrating each member to the desired time. A tutorial case with a test ensemble can be found at [http://www2.mmm.ucar.edu/wrf/users/wrfda/download/V38/wrfda_hybrid_etkf_testdata.tar.gz](http://www2.mmm.ucar.edu/wrf/users/wrfda/download/V38/wrfda_hybrid_etkf_testdata.tar.gz). In this example, the ensemble forecasts were initialized at 2015102612 and valid 2015102712. A hybrid analysis at 2015102712 will be performed using the ensemble valid 2015102712 as input. Once you have the initial ensemble, the ensemble mean and perturbations can be calculated following the steps below:

1) Set an environment variable for your working directory and your data directory

```
> setenv WORK_DIR your_hybrid_path
> setenv DAT_DIR your_data_path
> cd $WORK_DIR
```

2) Calculate the ensemble mean

a) From your working directory, copy or link the ensemble forecasts to your working directory. The ensemble members are identified by three-digit numbers following the valid time.

```
> ln -sf $DAT_DIR/Hybrid/fc/2015102612/wrfout_d01_2015-10-27_12:00:00.e* .
```
b) Provide two template files (ensemble mean and variance files) in your working directory. These files will be overwritten with the ensemble mean and variance as discussed below.

```
> cp $DAT_DIR/ Hybrid/ fc/2015102612/ wrfout_d01_2015-10-27_12:00:00.e001 ./wrfout_d01_2015-10-27_12:00:00.mean
> cp $DAT_DIR/ Hybrid/ fc/2015102612/ wrfout_d01_2015-10-27_12:00:00.e001 ./wrfout_d01_2015-10-27_12:00:00.vari
```

c) Copy `gen_be_ensmean_nl.nl` (cp $DAT_DIR/ Hybrid/ gen_be_ensmean_nl.nl). You will need to set the information in this script as follows:

```
&gen_be_ensmean_nl
directory = '.'
filename = 'wrfout_d01_2015-10-27_12:00:00'
num_members = 10
nv = 7
cv = 'U', 'V', 'W', 'PH', 'T', 'MU', 'QVAPOR'
```

where directory is the folder containing the ensemble members and template files, filename is the name of the files before their suffixes (e.g., .mean, .vari, etc), num_members is the number of ensemble members you are using, nv is the number of variables, and cv is the name of variables used in the hybrid system. Be sure nv and cv are consistent!

d) Link `gen_be_ensmean.exe` to your working directory and run it.

```
> ln -sf $WRFDA_DIR/ var/ build/ gen_be_ensmean.exe .
> ./gen_be_ensmean.exe
```

Check the output files. `wrfout_d01_2015-10-27_12:00:00.mean` is the ensemble mean; `wrfout_d01_2015-10-27_12:00:00.vari` is the ensemble variance.

3) Calculate ensemble perturbations

a) Create a sub-directory in which you will be working to create ensemble perturbations.

```
> mkdir -p ./ep
> cd ./ep
```

b) Run `gen_be_ep2.exe`. The executable requires four command-line arguments (DATE, NUM_MEMBER, DIRECTORY, and FILENAME) as shown below for the tutorial example:

```
> ln -sf $WRFDA_DIR/ var/ build/ gen_be_ep2.exe .
> ./gen_be_ep2.exe 2015102712 10 ../wrfout_d01_2015-10-27_12:00:00
```

c) Check the output files. A list of binary files should now exist. Among them,
**WRFDA**

`tmp.e*` are temporary scratch files that can be removed.

4) Back in the working directory, create the input file for vertical localization. This program requires one command-line argument: the number of vertical levels of the model configuration (same value as `e_vert` in the namelist; for the tutorial example, this should be 42).

   ```
   > cd $WORK_DIR
   > ln -sf $WRFDA_DIR/var/build/gen_be_vertloc.exe .
   > ./gen_be_vertloc.exe 42
   ```

   The output is `./be.vertloc.dat` in your working directory.

5) Run WRFDA in hybrid mode

   a) In your hybrid working directory, link all the necessary files and directories as follows:

   ```
   > ln -fs ./wrfout_d01_2015-10-27 12:00:00.mean ./fg (first guess is the ensemble mean for this test case)
   > ln -fs $WRFDA_DIR/run/LANDUSE.TBL .
   > ln -fs $DAT_DIR/Hybrid/ob/2015102712/ob.ascii ./ob.ascii
   > ln -fs $DAT_DIR/Hybrid/be/be.dat ./be.dat
   > ln -fs $WRFDA_DIR/var/build/da_wrfvar.exe .
   > cp $DAT_DIR/Hybrid/namelist.input .
   ```

   b) Edit `namelist.input`, paying special attention to the following hybrid-related settings:

   ```
   &wrfvar7
   je_factor = 2.0
   /
   &wrfvar16
   ensdim_alpha = 10
   alphacv_method = 2
   alpha_corr_type=3
   alpha_corr_scale = 500.0
   alpha_std_dev=1.000
   alpha_vertloc = .true.
   /
   ```

   c) Finally, execute the WRFDA file, running in hybrid mode

   ```
   > ./da_wrfvar.exe >& wrfda.log
   ```

   Check the output files; the output file lists are the same as when you run WRF 3D-Var.

   d. Hybrid namelist options

   **je_factor**
ensemble covariance weighting factor. This factor controls the weighting component of ensemble and static covariances. The corresponding \( j_b \) factor = \( j_e \) factor/(\( j_e \) factor - 1).

**ensdim_alpha**
the number of ensemble members. Hybrid mode is activated when ensdim_alpha is larger than zero

**alphacv_method**
1=perturbations in control variable space ("psi","chi_u","t_u","rh","ps_u");
2=perturbations in model space ("u","v","t","q","ps"). Option 2 is extensively tested and recommended to use.

**alpha_corr_type**
correlation function. 1=Exponential; 2=SOAR; 3=Gaussian.

**alpha_corr_scale**
hybrid covariance localization scale in km unit. Default value is 200.

**alpha_std_dev**
alpha standard deviation. Default value is 1.0

**alpha_vertloc**
for vertical localization. .true.=use vertical localization; .false.=no vertical localization

---

**ETKF Data Assimilation**

The WRFDA system also includes a ETKF assimilation technique. The ETKF system updates the ensemble perturbations. Please refer to Bishop et al. (2001) and Wang et al. (2003) for a detailed description of the methodology. The following section will give a brief introduction of some aspects of using the ETKF system.

**a. Source Code**

Three executables are used in the ETKF system. If you have successfully compiled the WRFDA system, you will see the following:

- `WRFDA/var/build/gen_be_etkf.exe`
- `WRFDA/var/build/gen_be_addmean.exe`
- `WRFDA/var/build/da_wrfvar.exe`

The file `gen_be_etkf.exe` is used to update the ensemble perturbations, while `gen_be_addmean.exe` is used to combine the ensemble mean and the ensemble perturbations. As with 3D-Var/4D-Var, `da_wrfvar.exe` is the main WRFDA program. However, in this case, `da_wrfvar.exe` will create filtered observations and prepare formatted omb files for ETKF.

**b. Running the ETKF System**
The first procedure is to update the ensemble perturbations. A set of initial ensemble members must be prepared before the ETKF application can be started. The ensemble can be obtained from a previous ensemble forecast. A tutorial case with a test ensemble can be found at http://www2.mmm.ucar.edu/wrf/users/wrfda/download/V38/wrfda_hybrid_etkf_testdata.tar.gz. In this example, the ensemble forecasts were initialized at 2015102612 and valid 2015102712. ETKF will be performed using the ensemble valid 2015102712 as input. Once you have the initial ensemble, the ensemble perturbations can be updated by following the steps below:

1) Set environment variables for convenience

   > setenv WORK_DIR_ETKF your_etkf_path
   > setenv DAT_DIR your_data_path
   > setenv WRFDA_DIR your_WRFDA_path
   > cd $WORK_DIR_ETKF

2) Prepare the filtered observations

   a) In your ETKF working directory, make a subdirectory to prepare the filtered observations and link all the necessary files and directories as follows:

   > mkdir obs_filter
   > cd obs_filter
   > ln -fs $DAT_DIR/Hybrid/fc/2015102612/wrfout_d01_2015-10-27_12:00:00.mean ./fg (first guess is the ensemble mean)
   > ln -fs $WRFDA_DIR/run/LANDUSE.TBL .
   > ln -fs $DAT_DIR/Hybrid/ob/2015102712/ob.ascii ./ob.ascii
   > ln -fs $DAT_DIR/Hybrid/be/be.dat ./be.dat
   > cp $DAT_DIR/ETKF/namelist.input .

   b) Edit namelist.input, paying special attention to the following 'QC-OBS'-related settings:

   &wrfvar17
   analysis_type = 'QC-OBS',
   /

   c) Execute the WRFDA file, running in QC-OBS mode

   > ./da_wrfvar.exe >& wrfda.log

Check the output files; you should see a 'filtered_obs_01' file which contains the filtered observations.

3) Prepare omb files for ETKF

   a) In your ETKF working directory, make a subdirectory to prepare the omb files for
each ensemble member and link all the necessary files and directories as follows:

```bash
> cd $WORK_DIR_ETKF
> mkdir -p omb/working.e001
> cd omb/working.e001
> ln -fs $DAT_DIR/Hybrid/fc/2015102612/wrfout_d01_2015-10-27_12:00:00.e001 ./fg (first guess is the ensemble member)
> ln -fs $WRFDA_DIR/run/LANDUSE.TBL .
> ln -fs $WRFDA_DIR/obs_filter/filtered_obs_01 ./ob.ascii
> ln -fs $WRFDA_DIR/var/build/da_wrfvar.exe .
> cp $DAT_DIR/ETKF/namelist.input .
```

b) Edit namelist.input, paying special attention to the following 'VERIFY'-related settings:

```plaintext
&wrfvar17
  analysis_type = 'VERIFY',
/
```

c) Execute the WRFDA file, running in VERIFY mode

```bash
> ./da_wrfvar.exe >& wrfda.log
```

Check the output files. The output files are the same as when you run WRF 3D-Var (except wrfvar_output will NOT be created), and the 'ob.etkf.0*' files are omb files.

d) Combine the ob.etkf.0* files and add the observation number in the head of ob.etkf.e0*

```bash
> cat ob.etkf.0* > ob.all
> wc -l ob.all > ob.etkf.e001
> cat ob.all >> ob.etkf.e001
```

e) Likewise, prepare ob.etkf.e0* files for other ensemble members

4) Run ETKF

a) Copy or link the ensemble mean and forecasts and ob.etkf.e0* files to your working directory and make a parameter directory to save the parameter files.

```bash
> cd $WORK_DIR_ETKF
> setenv PAR_DIR $WORK_DIR_ETKF/param
> mkdir $PAR_DIR_ETKF
> ln -sf $DAT_DIR/Hybrid/fc/2015102612/wrfout_d01_2015-10-27_12:00:00.mean ./etkf_input
> ln -sf $DAT_DIR/Hybrid/fc/2015102612/wrfout_d01_2015-10-27_12:00:00.e001 ./etkf_input.e001
...> ln -sf $DAT_DIR/Hybrid/fc/2015102712/wrfout_d01_2015-10-27_12:00:00.e010 ./etkf_input.e010
```
b) Provide template files. These files will be overwritten with the ensemble perturbations.

   > cp $DAT_DIR/ETKF/gen_be_etkf_nl.nl

   You will need to set the information in this script as follows:

   &gen_be_etkf_nl
   num_members = 10,
   nv = 7,
   cv = 'U', 'V', 'W', 'PH', 'T', 'QVAPOR', 'MU',
   naccum1 = 20,
   naccum2 = 20,
   nstartaccum1 = 1,
   nstartaccum2 = 1,
   nout = 1,
   tainflatinput = 1,
   rhoinput = 1,
   infl_fac_file = '$PAR_DIR_ETKF/inflation_factor.dat',
   infl_let_file = '$PAR_DIR_ETKF/inflation_letkf.dat',
   eigen_val_file = '$PAR_DIR_ETKF/eigen_value.dat',
   inno2_val_file = '$PAR_DIR_ETKF/innovation_value.dat',
   proj2_val_file = '$PAR_DIR_ETKF/projection_value.dat',
   infl_fac_TRNK = .false.,
   infl_fac_WG03 = .false.,
   infl_fac_WG07 = .true.,
   infl_fac_BOWL = .false.,
   letkf_flg = .false.,
   rand_filt = .false.,
   rnd_seed = 201502712,
   rnd_nobs = 5000
   etkf_erro_max = 20.
   etkf_erro_min = .00001
   etkf_inno_max = 20.
   etkf_inno_min = .00001
   etkf_erro_flg = .true.
   etkf_inno_flg = .true.
   etkf_wrfda = .false.

   Important note: since environment variables are not parsed when reading namelists, you
   MUST manually change $PAR_DIR_ETKF to its actual value in the namelist

   Where the various namelist parameters are as follows:
- **num\_members** is the ensemble members size
- **nv** is the number of variables
- **cv** the name of variables
- **naccum\_1** and **naccum\_1** are number of previous cycles used to accumulate for inflation and rho factor
- **nstartaccum\_1** and **nstartaccum\_2** are not used for ordinary ETKF
- **nout** is the cycle index
- **tainflatinput** and **rhoinput** are prescribed factors for inflation and rho factor
- **infl\_fac\_file**, **eigen\_val\_file**, **inno2\_val\_file** and **proj2\_val\_file** are files to save template parameters
- **infl\_fac\_TRNK**, **infl\_fac\_WG03**, **infl\_fac\_WG07**, and **infl\_fac\_BOWL** are options for different adaptive inflation schemes
- **rand\_filt**, **rnd\_seed** and **rnd\_nobs** are options for using filtered observation and random observations
- **etkf\_erro\_max**, **etkf\_erro\_min**, **etkf\_inno\_max**, **etkf\_inno\_min**, **etkf\_erro\_flg**, **etkf\_inno\_flg**, and **etkf\_wrfda** are options to conduct further observation filtering.

**d)** Link gen\_be\_etkf.exe to your working directory and run it.

```bash
> ln -sf $WRFDA_DIR/var/build/gen_be_etkf.exe .
> ./gen_be_etkf.exe
```

Check the output files. etkf\_output.* files are updated ensemble perturbations.

**5)** Add updated ensemble perturbations to the ensemble mean to get new ensemble members

```bash
> cd $WORK\_DIR\_ETKF
```

**a)** Copy add\_mean\_nl.nl (cp SDAT\_DIR/ETKF/add\_mean\_nl.nl .)

You will need to set the information in this script as follows for each member:

```bash
&add\_mean\_nl
num\_members = 10
.cv = 'U', 'V', 'W', 'PH', 'T', 'QVAPOR', 'MU'
nv = 7
path = '$WORK\_DIR\_ETKF'
file\_mean = 'etkf\_input'
file\_pert = 'etkf\_output.e001' (for each member, etkf\_output.e0*...)
```

*Again, be sure to substitute the actual path in the place of $WORK\_DIR\_ETKF*

**b)** Run gen\_be\_addmean.exe.
Check the output files. etkf_output.e0* files are the new ensemble members.

### Description of Namelist Variables

#### a. WRFDA namelist variables

<table>
<thead>
<tr>
<th>Variable Names</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;wrfvar1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>write_increments</td>
<td>false</td>
<td>.true.: write out a binary analysis increment file</td>
</tr>
<tr>
<td>var4d</td>
<td>false</td>
<td>.true.: 4D-Var mode</td>
</tr>
<tr>
<td>var4d_lbc</td>
<td>true</td>
<td>.true.: on/off for lateral boundary control in 4D-Var</td>
</tr>
<tr>
<td>var4d_bin</td>
<td>3600</td>
<td>seconds, observation sub-window length for 4D-Var</td>
</tr>
<tr>
<td>var4d_bin_rain</td>
<td>3600</td>
<td>seconds, precipitation observation sub-window length for 4D-Var</td>
</tr>
<tr>
<td>multi_inc</td>
<td>0</td>
<td>&gt; 0: multi-incremental run</td>
</tr>
<tr>
<td>print_detail_radar</td>
<td>false</td>
<td>print_detail_xxx: output extra (sometimes can be too many) diagnostics for debugging; not recommended to turn these on for production runs</td>
</tr>
<tr>
<td>print_detail_xa</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>print_detail_xb</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>print_detail_obs</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>print_detail_grad</td>
<td>false</td>
<td>.true.: to print out a detailed gradient of each observation type at each iteration</td>
</tr>
<tr>
<td>check_max_iv_print</td>
<td>true</td>
<td>obsolete (used only by Radar)</td>
</tr>
<tr>
<td>update_sfc_diags</td>
<td>false</td>
<td>.true.: update T2/Q2/U10/V10/TH2 with WRFDA re-diagnosed values. Use only with sf_sfclay.physics=91 in WRF</td>
</tr>
<tr>
<td>&amp;wrfvar2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>analysis_accu</td>
<td>900</td>
<td>in seconds: if the time difference between the namelist date (analysis_date) and date info read-in from the first guess is larger than analysis_accu, WRFDA will abort.</td>
</tr>
<tr>
<td>calc_w_increment</td>
<td>false</td>
<td>.true.: the increment of the vertical velocity, W, will be diagnosed based on the increments of other fields. .false.: the increment of the vertical velocity W is zero if no W information is assimilated. If there is information on the W from observations assimilated, such as radar radial velocity, the W increments are always computed, whether calc_w_increment=true. or .false.</td>
</tr>
<tr>
<td>wind_sd</td>
<td>false</td>
<td>true: wind values which are reported as</td>
</tr>
</tbody>
</table>
**WRFDA**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>qc_rej_both</strong></td>
<td>false</td>
<td>false: all wind obs are converted to u/v prior to assimilation. True: if either u or v (spd or dir) do not pass quality control, both obs are rejected. False: (default behavior) qc on wind obs is handled individually.</td>
</tr>
<tr>
<td><strong>fg_format</strong></td>
<td>1</td>
<td>1: fg_format_wrf_arw_regional (default). 3: fg_format_wrf_arw_global. 4: fg_format_kma_global. Options 3 and 4 are untested; use with caution!</td>
</tr>
<tr>
<td><strong>ob_format</strong></td>
<td>2</td>
<td>1: read in NCEP PREPBUFR data from ob.bufr. 2: read in data from ob.ascii (default).</td>
</tr>
<tr>
<td><strong>ob_format_gpsro</strong></td>
<td>2</td>
<td>1: read in GPSRO data from gpsro.bufr. 2: read in GPSRO data from ob.ascii (default).</td>
</tr>
<tr>
<td><strong>num_fgat_time</strong></td>
<td>1</td>
<td>1: 3DVar. &gt; 1: number of time slots for FGAT and 4DVAR.</td>
</tr>
<tr>
<td><strong>thin_conv</strong></td>
<td>true</td>
<td>Turns on observation thinning for ob_format=1 (NCEP PREPBUFR) only. thin_conv can be set to .false., but this is not recommended.</td>
</tr>
<tr>
<td><strong>thin_conv_ascii</strong></td>
<td>false</td>
<td>Turns on observation thinning for ob_format=2 (ASCII from OBSPROC) only.</td>
</tr>
<tr>
<td><strong>thin_mesh_conv</strong></td>
<td>20</td>
<td>km, each observation type can set its thinning mesh (max_instrument and the index/order follows the definition in WRFDA/var/da/da_control/da_control.f90).</td>
</tr>
<tr>
<td><strong>use_synopobs</strong></td>
<td>true</td>
<td>use_xxxobs - .true.: assimilate xxx obs if available. .false.: do not assimilate xxx obs even available.</td>
</tr>
<tr>
<td><strong>use_shipsobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_metarobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_soundobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_pilotobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_airepobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_geoamvobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_polaramvobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_bogusobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_buoyobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_profilerobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_satemobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_gspwobs</strong></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td><strong>use_gpsztdobs</strong></td>
<td>false</td>
<td>Note: unlike most use_*obs variables, the default for use_gpsztdobs is false. This is because PW and</td>
</tr>
</tbody>
</table>
ZTD observations can not be assimilated simultaneously, so one of them must be false.

use_gpsrefobs true
use_qscatobs true
use_radarobs false .true.: Assimilate radar data
use_radar_rv false Assimilate radar velocity observations
use_radar_rf false Assimilate radar reflectivity using original reflectivity operator (total mixing ratio)
use_radar_rqv false Assimilate radar reflectivity using estimated humidity from radar reflectivity
use_radar_rhv false Assimilate radar reflectivity using rainwater and ice mixing ratios
use_3dvar_phy false Partition hydrometeors via the moist explicit scheme (warm rain process)
use_rainobs false .true.: Assimilate precipitation data
thin_rainobs true .true.: perform thinning on precipitation data
use_airsretobs true ; use_hirs2obs, use_hirs3obs, use_hirs4obs, use_mhsobs, use_msuobs,
; use_amsuaobs, use_amsubobs, use_airsobs, use_eos_amsuaobs, use_ssmisobs
; radiance-related variables that only control if corresponding BUFR files are read
; into WRFDA or not, but do not control if the data is assimilated or not. Additional
; variables have to be set in &wrfvar14 in order to assimilate radiance data.
use_hirs2obs false .true.: read in data from hirs2.bufr
use_hirs3obs false .true.: read in data from hirs3.bufr
use_hirs4obs false .true.: read in data from hirs4.bufr
use_mhsobs false .true.: read in data from mhs.bufr
use_msuobs false .true.: read in data from msu.bufr
use_amsuaobs false .true.: read in data from amsua.bufr
use_amsubobs false .true.: read in data from amsub.bufr
use_airsobs false .true.: read in data from airs.bufr
use_eos_amsuaobs false .true.: read in data EOS-AMSUA data from airs.bufr
use_ssmisobs false .true.: to read in data from ssmis.bufr
use_atmsobs false .true.: to read in data from atms.bufr
use_iasiobs false .true.: to read in data from iasi.bufr
use_seviriobs false .true.: to read in data from seviri.bufr
use_amsr2obs false .true.: to read in data from AMSR2 files (see section “Other data formats” for file names)
use_obs_errfac false .true.: apply obs error tuning factors if errfac.dat is available for conventional data only

&wrfvar5
check_max_iv true .true.: reject the observations whose innovations (O-B) are larger than a maximum value defined as a multiple of the observation error for each observation. i.e., inv > (obs_error*factor) -->
fails_error_max; the default maximum value is 5 times the observation error; the factor of 5 can be changed through max_error_* settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_error_t</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for t</td>
</tr>
<tr>
<td>max_error_uv</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for u</td>
</tr>
<tr>
<td>max_error_pw</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for pre-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>precipitable water</td>
</tr>
<tr>
<td>max_error_ref</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for gps</td>
</tr>
<tr>
<td>max_error_q</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for spe-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cific humidity</td>
</tr>
<tr>
<td>max_error_p</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for pres-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sure pressure</td>
</tr>
<tr>
<td>max_error_thickness</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for thic-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kness</td>
</tr>
<tr>
<td>max_error_rv</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for ra-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dar radial velocity</td>
</tr>
<tr>
<td>max_error_rf</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for ra-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dar reflectivity</td>
</tr>
<tr>
<td>max_error_rain</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for pre-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cipitation</td>
</tr>
<tr>
<td>max_error_spd</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for win-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d speed (wind_sd=.true. only)</td>
</tr>
<tr>
<td>max_error_dir</td>
<td>5.0</td>
<td>Maximum check_max_iv error check factor for win-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d direction (wind_sd=.true. only)</td>
</tr>
<tr>
<td>put_rand_seed</td>
<td>false</td>
<td>For RANDOMCV: setting to “true” allows you to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enter your own seed numbers (see &amp;wrfvar11)</td>
</tr>
</tbody>
</table>

**&wrfvar6 (for minimization options)**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_ext_its</td>
<td>1</td>
<td>Number of outer loops</td>
</tr>
<tr>
<td>nmax</td>
<td>200</td>
<td>Maximum number of iterations in an inner loop criterion</td>
</tr>
<tr>
<td>(max_ext_its)</td>
<td></td>
<td>(uses dimension: max_ext_its)</td>
</tr>
<tr>
<td>eps</td>
<td>0.01</td>
<td>Minimization convergence criterion (uses dimension: max_ext_its); minimiza-</td>
</tr>
<tr>
<td>(max_ext_its)</td>
<td></td>
<td>tion stops when the norm of the gradient of the cost function gradient is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reduced by a factor of eps. inner minimization stops either when the criter-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ion is met or when inner iterations reach nmax.</td>
</tr>
<tr>
<td>orthonorm_gradient</td>
<td>false</td>
<td>.true.: the gradient vectors are stored during the Conjugate Gradient for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>each iteration and used to re-orthogonalize the new gradient. This requires</td>
</tr>
<tr>
<td></td>
<td></td>
<td>extra storage of large vectors (each one being the size of the control vari-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>able) but results in a better convergence of the Conjugate Gradient after</td>
</tr>
<tr>
<td></td>
<td></td>
<td>around 20 iterations.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cv_options</td>
<td>5</td>
<td>3: NCEP Background Error model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: NCAR Background Error model (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6: Use of moisture-mutivariate background error statistics (CV6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7: New NCAR Background Error model (CV7)</td>
</tr>
<tr>
<td>as1(3)</td>
<td>0.25, 1.0, 1.5</td>
<td>tuning factors for variance, horizontal and vertical scales for control variable 1 = stream function. For cv_options=3 only.</td>
</tr>
<tr>
<td>as2(3)</td>
<td>0.25, 1.0, 1.5</td>
<td>tuning factors for variance, horizontal and vertical scales for control variable 2 - unbalanced potential velocity. For cv_options=3 only.</td>
</tr>
<tr>
<td>as3(3)</td>
<td>0.25, 1.0, 1.5</td>
<td>tuning factors for variance, horizontal and vertical scales for control variable 3 - unbalanced temperature. For cv_options=3 only.</td>
</tr>
<tr>
<td>as4(3)</td>
<td>0.25, 1.0, 1.5</td>
<td>tuning factors for variance, horizontal and vertical scales for control variable 4 - pseudo relative humidity. For cv_options=3 only.</td>
</tr>
<tr>
<td>as5(3)</td>
<td>0.25, 1.0, 1.5</td>
<td>tuning factors for variance, horizontal and vertical scales for control variable 5 - unbalanced surface pressure. For cv_options=3 only.</td>
</tr>
<tr>
<td>rf_passes</td>
<td>6</td>
<td>number of passes of recursive filter.</td>
</tr>
<tr>
<td>var_scaling1</td>
<td>1.0</td>
<td>tuning factor of background error covariance for control variable 1 - stream function. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>var_scaling2</td>
<td>1.0</td>
<td>tuning factor of background error covariance for control variable 2 - unbalanced velocity potential. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>var_scaling3</td>
<td>1.0</td>
<td>tuning factor of background error covariance for control variable 3 - unbalanced temperature. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>var_scaling4</td>
<td>1.0</td>
<td>tuning factor of background error covariance for control variable 4 - pseudo relative humidity. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>var_scaling5</td>
<td>1.0</td>
<td>tuning factor of background error covariance for control variable 5 - unbalanced surface pressure. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>len_scaling1</td>
<td>1.0</td>
<td>tuning factor of scale-length for stream function. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>len_scaling2</td>
<td>1.0</td>
<td>tuning factor of scale-length for unbalanced velocity potential. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>len_scaling3</td>
<td>1.0</td>
<td>tuning factor of scale-length for unbalanced temperature. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>len_scaling4</td>
<td>1.0</td>
<td>tuning factor of scale-length for pseudo relative humidity. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>len_scaling5</td>
<td>1.0</td>
<td>tuning factor of scale-length for unbalanced surface pressure. For cv_options=5, 6, and 7 only.</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>je_factor</td>
<td>1.0</td>
<td>Pressure. For cv_options=5, 6, and 7 only. ensemble covariance weighting factor</td>
</tr>
<tr>
<td>&amp;wrfvar8</td>
<td>not used</td>
<td></td>
</tr>
<tr>
<td>&amp;wrfvar9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdout</td>
<td>6</td>
<td>Unit number for standard output</td>
</tr>
<tr>
<td>stderr</td>
<td>0</td>
<td>Unit number for error output</td>
</tr>
<tr>
<td>trace_unit</td>
<td>7</td>
<td>Unit number for tracing output. Note that units 10 and 9 are reserved for reading namelist.input and writing namelist.output respectively.</td>
</tr>
<tr>
<td>trace_pe</td>
<td>0</td>
<td>Currently, statistics are always calculated for all processors, and output by processor 0.</td>
</tr>
<tr>
<td>trace_repeat_head</td>
<td>10</td>
<td>The number of times any trace statement will produce output for any particular routine. This stops overwhelming trace output when a routine is called multiple times. Once this limit is reached a 'going quiet' message is written to the trace file, and no more output is produced from the routine, though statistics are still gathered. see trace_repeat_head description</td>
</tr>
<tr>
<td>trace_repeat_body</td>
<td>10</td>
<td>see trace_repeat_head description</td>
</tr>
<tr>
<td>trace_max_depth</td>
<td>30</td>
<td>Define the deepest level to which tracing writes output</td>
</tr>
<tr>
<td>trace_use</td>
<td>false</td>
<td>.true.: activate tracing</td>
</tr>
<tr>
<td>trace_use_frequent</td>
<td>false</td>
<td>.true.: calculate allocated memory using a mallinfo call. On some platforms (Cray and Mac), mallinfo is not available and no memory monitoring can be done.</td>
</tr>
<tr>
<td>trace_use_dull</td>
<td>false</td>
<td>.true.: tracing is output for all pes. As stated in trace_pe, this does not change processor statistics.</td>
</tr>
<tr>
<td>trace_memory</td>
<td>true</td>
<td>.true.: tracing statistics are written to a xxxx.csv file in CSV format</td>
</tr>
<tr>
<td>trace_all_pes</td>
<td>false</td>
<td>.true.: tracing and error reporting routines will include HTML tags.</td>
</tr>
<tr>
<td>trace_csv</td>
<td>true</td>
<td>.true.: warning messages that would normally allow the program to continue are treated as fatal errors.</td>
</tr>
<tr>
<td>use_html</td>
<td>true</td>
<td>.true.: perform adjoint tests</td>
</tr>
<tr>
<td>warnings_areFatal</td>
<td>false</td>
<td>.true.: perform gradient test</td>
</tr>
<tr>
<td>&amp;wrfvar10 (for code developer)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>test_transforms</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>test_gradient</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>&amp;wrfvar11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cv_options_hum</td>
<td>1</td>
<td>Do not change</td>
</tr>
<tr>
<td>Option</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>check_rh</td>
<td>0</td>
<td>0 --&gt; No supersaturation check after minimization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 --&gt; supersaturation (rh &gt; 100%) and minimum rh (rh &lt; 10%) check, and make</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the local adjustment of q.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 --&gt; supersaturation (rh &gt; 95%) and minimum rh (rh &lt; 11%) check and make</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the multi-level q adjustment under the constraint of conserved column</td>
</tr>
<tr>
<td></td>
<td></td>
<td>integrated water vapor</td>
</tr>
<tr>
<td>sfc_assi_options</td>
<td>1</td>
<td>1 --&gt; surface observations will be assimilated based on the lowest model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>level first guess. Observations are not used when the elevation difference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>between the observing site and the lowest model level is larger than 100m.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 --&gt; surface observations will be assimilated based on surface similarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>theory in PBL. Innovations are computed based on 10-m wind, 2-m temperature</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and 2-m moisture.</td>
</tr>
<tr>
<td>psfc_from_slp</td>
<td>false</td>
<td>.true.: when sfc_assi_options=1, re-calculates Psfc from SLP when the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>observation elevation is below the lowest model level height. This was the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>behavior prior to V3.8, but is not recommended.</td>
</tr>
<tr>
<td>calculate_cg_cost_fn</td>
<td>false</td>
<td>conjugate gradient algorithm does not require the computation of cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td>function at every iteration during minimization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.true.: Compute and write out cost function and gradient for each iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>into files cost_fn and grad_fn for diagnostic purposes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false.: Only the initial and final cost functions are computed and output.</td>
</tr>
<tr>
<td>seed_array1</td>
<td>1</td>
<td>For RANDOMCV when put_rand_seed=true, first integer for seeding the random</td>
</tr>
<tr>
<td></td>
<td></td>
<td>function</td>
</tr>
<tr>
<td>seed_array2</td>
<td>1</td>
<td>For RANDOMCV when put_rand_seed=true, second integer for seeding the random</td>
</tr>
<tr>
<td></td>
<td></td>
<td>function</td>
</tr>
<tr>
<td>use_wpec</td>
<td>false</td>
<td>true: enables the WPEC dynamic constraint term</td>
</tr>
<tr>
<td>wpec_factor</td>
<td>0.001</td>
<td>WPEC dynamic constraint weighting factor</td>
</tr>
<tr>
<td>balance_type</td>
<td>3</td>
<td>1 = geostrophic term only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = cyclostrophic term only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = geostrophic + cyclostrophic terms</td>
</tr>
<tr>
<td>max_vert_var1</td>
<td>99.0</td>
<td>specify the maximum truncation value (percentage) to explain the variance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of stream function in eigenvector decomposition</td>
</tr>
<tr>
<td>max_vert_var2</td>
<td>99.0</td>
<td>specify the maximum truncation value (percentage) to explain the variance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of unbalanced potential velocity in eigenvector decomposition</td>
</tr>
</tbody>
</table>

WRF-ARW V3: User’s Guide 6-69
**max_vert_var3** 99.0 specify the maximum truncation value (percentage) to explain the variance of the unbalanced temperature in eigenvector decomposition.

**max_vert_var4** 99.0 specify the maximum truncation value (percentage) to explain the variance of pseudo relative humidity in eigenvector decomposition.

**max_vert_var5** 99.0 for unbalanced surface pressure, it should be a non-zero positive number.

set max_vert_var5=0.0 only for offline VarBC applications.

**psi_ch_factor** 1.0 Contribution of stream function in defining balanced part of velocity potential. For cv_options=6 only.

**psi_t_factor** 1.0 Contribution of stream function in defining balanced part of temperature. For cv_options=6 only.

**psi_ps_factor** 1.0 Contribution of stream function in defining balanced part of surface pressure. For cv_options=6 only.

**psi_rh_factor** 1.0 Contribution of stream function in defining balanced part of moisture. For cv_options=6 only.

**chi_u_t_factor** 1.0 Contribution of the unbalanced part of velocity potential in defining balanced part of temperature. For cv_options=6 only.

**chi_u_ps_factor** 1.0 Contribution of the unbalanced part of velocity potential in defining balanced part of surface pressure. For cv_options=6 only.

**chi_u_rh_factor** 1.0 Contribution of the unbalanced part of velocity potential in defining balanced part of moisture. For cv_options=6 only.

**t_u_rh_factor** 1.0 Contribution of the unbalanced part of temperature in defining balanced part of moisture. For cv_options=6 only.

**ps_u_rh_factor** 1.0 Contribution of the unbalanced part of surface pressure in defining balanced part of moisture. For cv_options=6 only.

---

**&wrfvar14**

the following 4 variables (rtminit_nsensor, rtminit_platform, rtminit_satid, rtminit_sensor) together control what sensors to be assimilated.

**rtminit_nsensor** 1 total number of sensors to be assimilated.

**rtminit_platform** -1 platforms IDs array (used dimension: (max_instruments) rtminit_nsensor); e.g., 1 for NOAA, 9 for EOS, 10 for METOP and 2 for DMSP.

**rtminit_satid** -1.0 satellite IDs array (used dimension: (max_instruments) rtminit_nsensor)

**rtminit_sensor** -1.0 sensor IDs array (used dimension: (max_instruments) rtminit_nsensor); e.g., 0 for HIRS, 3 for AMSU-
A, 4 for AMSU-B, 15 for MHS, 10 for SSMIS, 11 for AIRS

rad_monitoring 0 integer array (used dimension: rtminit_nsensor); (max_instruments) 0: assimilating mode; 
1: monitoring mode (only calculate innovations)

thinning_mesh 60.0 real array (used dimension: rtminit_nsensor); (max_instruments) specify thinning mesh size (in km) for different sensors.

thinning false .true.: perform thinning on radiance data

qc_rad true .true.: perform quality control. Do not change.

write_iv_rad_ascii false .true.: output radiance Observation minus Background files, which are in ASCII format and separated by sensor and processor.

write_oa_rad_ascii false .true.: output radiance Observation minus Analysis files (Observation minus Background information is also included), which are in ASCII format and separated by sensor and processor.

use_error_factor_rad false .true.: use a radiance error tuning factor file radiance_error.factor, which can be created with empirical values or generated using variational tuning method (Desroziers and Ivanov, 2001)

use_antcorr false .true.: perform Antenna Correction in CRTM (max_instruments)

rtm_option 1 which RTM (Radiative Transfer Model) to use (To use RTTOV, WRFDA must be compiled to include RTTOV libraries; see first section for details) 
1: RTTOV 
2: CRTM

only_sea_rad false .true.: assimilate radiance over water only

use_varbc false .true.: perform Variational Bias Correction. A parameter file in ASCII format called VARBC.in (a template is provided with the source code tar ball) is required.

freeze_varbc false .true: together with use_varbc=.false., keep the VarBC bias parameters constant in time. In this case, the bias correction is read and applied to the innovations, but it is not updated during the minimization.

varbc_factor 1.0 for scaling the VarBC preconditioning

varbc_nobsmin 10 defines the minimum number of observations required for the computation of the predictor statistics during the first assimilation cycle. If there are not enough data (according to "VARBC_NOBSMIN") on the first cycle, the
next cycle will perform a coldstart again.

`use_clddet_mmr` false
true: use the MMR scheme to conduct cloud detection for infrared radiance
false: uses the observation brightness temperature for AIRS Window channel #914 as criterion for GSI thinning (with a higher amplitude than the distance from the observation location to the nearest grid point).

`use_clddet_ecmwf` false
true: use the ECMWF operational scheme to conduct cloud detection for infrared radiance.
false: uses the observation brightness temperature for AIRS Window channel #914 as criterion for GSI thinning (with a higher amplitude than the distance from the observation location to the nearest grid point).

`airs_warmest_fov` false
true: uses the observation brightness temperature for AIRS Window channel #914 as criterion for GSI thinning (with a higher amplitude than the distance from the observation location to the nearest grid point).

`use_crtm_kmatrix` true
true: use CRTM K matrix rather than calling CRTM TL and AD routines for gradient calculation, which reduces runtime noticeably.
false: uses the observation brightness temperature for AIRS Window channel #914 as criterion for GSI thinning (with a higher amplitude than the distance from the observation location to the nearest grid point).

`use_rttov_kmatrix` false
true: use RTTOV K matrix rather than calling RTTOV TL and AD routines for gradient calculation, which reduces runtime noticeably.
false: uses the observation brightness temperature for AIRS Window channel #914 as criterion for GSI thinning (with a higher amplitude than the distance from the observation location to the nearest grid point).

`rttov_emis_atlas_ir` 0
0: do not use IR emissivity atlas
1: use IR emissivity atlas (recommended)

`rttov_emis_atlas_mw` 0
0: do not use MW emissivity atlas
1: use TELSEM MW emissivity atlas (recommended)
2: use CNRM MW emissivity atlas

`use_blacklist_rad` true
true: switch off the assimilation of known problematic channels (up to year 2012) that are hard-coded in var/da/da_radiance/da_blacklist_rad.inc.
false: users need to specify proper channel selections in the radiances_info files.

&wrfvar15 (needs to be set together with &wrfvar19)

`num_pseudo` 0
Set the number of pseudo observations, either 0 or 1 (single ob)

`pseudo_x` 1.0
Set the x-position (I) of the OBS in unit of grid-point.

`pseudo_y` 1.0
Set the y-position (J) of the OBS in unit of grid-point.

`pseudo_z` 1.0
Set the z-position (K) of OBS with the vertical level index, in bottom-up order.

`pseudo_val` 1.0
Set the innovation of the ob; wind in m/s, pressure in Pa, temperature in K, specific humidity in kg/kg

`pseudo_err` 1.0
set the error of the pseudo ob. Unit the same as pseudo_val.; if pseudo_var="q", pseudo_err=0.001 is more reasonable.

&wrfvar16 (for hybrid WRFDA/ensemble)

`alphacv_method` 2
1: ensemble perturbations in control variable space
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensdim_alpha</td>
<td>0</td>
</tr>
<tr>
<td>alpha_corr_type</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1: alpha_corr_type_exp</td>
</tr>
<tr>
<td></td>
<td>2: alpha_corr_type_soar</td>
</tr>
<tr>
<td></td>
<td>3: alpha_corr_type_gaussian (default)</td>
</tr>
<tr>
<td>alpha_corr_scale</td>
<td>200.0</td>
</tr>
<tr>
<td></td>
<td>km</td>
</tr>
<tr>
<td>&amp;wrfvar17 analysis_type</td>
<td>“3D-VAR”</td>
</tr>
<tr>
<td></td>
<td>&quot;3D-VAR&quot;: 3D-VAR mode (default);</td>
</tr>
<tr>
<td></td>
<td>&quot;QC-OBS&quot;: 3D-VAR mode plus extra filtered_obs output;</td>
</tr>
<tr>
<td></td>
<td>&quot;VERIFY&quot;: verification mode. WRFDA resets check_max_iv=.false. and ntmax=0;</td>
</tr>
<tr>
<td></td>
<td>&quot;RANDOMCV&quot;: for creating ensemble perturbations</td>
</tr>
<tr>
<td>adj_sens</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>.true.: write out gradient of Jo for adjoint sensitivity</td>
</tr>
<tr>
<td>&amp;wrfvar18 (needs to set &amp;wrfvar21 and &amp;wrfvar22 as well if ob_format=1 and/or radiances are used) analysis_date</td>
<td>“2002-08-03_00:00:00.00”</td>
</tr>
<tr>
<td></td>
<td>specify the analysis time. It should be consistent with the first guess time;</td>
</tr>
<tr>
<td></td>
<td>if time difference between analysis_date and date info read in from first guess</td>
</tr>
<tr>
<td></td>
<td>is larger than the &amp;wrfvar2 setting “analysis_accu”, WRFDA will abort.</td>
</tr>
<tr>
<td>&amp;wrfvar19 (needs to be set together with &amp;wrfvar15) pseudo_var</td>
<td>“t”</td>
</tr>
<tr>
<td></td>
<td>Set the name of the OBS variable:</td>
</tr>
<tr>
<td></td>
<td>’u’ = X-direction component of wind,</td>
</tr>
<tr>
<td></td>
<td>’v’ = Y-direction component of wind,</td>
</tr>
<tr>
<td></td>
<td>’t’ = Temperature,</td>
</tr>
<tr>
<td></td>
<td>’p’ = Pressure,</td>
</tr>
<tr>
<td></td>
<td>’q’ = Specific humidity</td>
</tr>
<tr>
<td></td>
<td>&quot;pw&quot;: total precipitable water</td>
</tr>
<tr>
<td></td>
<td>&quot;ref&quot;: refractivity</td>
</tr>
<tr>
<td></td>
<td>&quot;ztd&quot;: zenith total delay</td>
</tr>
<tr>
<td>&amp;wrfvar20 documentation_url</td>
<td>“<a href="http://www.mmm.ucar.edu/people/wrfhelp/wrfvar/code/trunk%E2%80%9D">http://www.mmm.ucar.edu/people/wrfhelp/wrfvar/code/trunk”</a></td>
</tr>
<tr>
<td>&amp;wrfvar21 time_window_min</td>
<td>&quot;2002-08-02_21:00:00.00Ob_format=1 and radiances to select observations inside the defined time_window. Note: Start from V3.1, this variable is also used for ob_format=2 to double-check if the obs are within the specified time window.</td>
</tr>
</tbody>
</table>
| &wrfvar22
**time_window_max**

"2002-08-03 03:00:00.00" end time of assimilation time window used for ob_format=1 and radiances to select observations inside the defined time_window. Note: this variable is also used for ob_format=2 to double-check if the obs are within the specified time_window.

**&perturbation (settings related to the 4D-Var)**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jcdfi_use</td>
<td>false</td>
<td>.true.: Include JcDF term in cost function.</td>
</tr>
<tr>
<td></td>
<td>.false.: Ignore JcDF term in cost function.</td>
<td></td>
</tr>
<tr>
<td>jcdfi_diag</td>
<td>1</td>
<td>0: Doesn't print out the value of Jc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Print out the value of Jc.</td>
</tr>
<tr>
<td>jcdfi_penalty</td>
<td>10</td>
<td>The weight to Jc term.</td>
</tr>
<tr>
<td>enable_identity</td>
<td>.false.</td>
<td>.true.: use identity adjoint and tangent linear model in 4D-Var.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.false.: use full adjoint and tangent linear model in 4D-Var.</td>
</tr>
<tr>
<td>trajectory_io</td>
<td>.true.</td>
<td>.true.: use memory I/O in 4D-Var for data exchange</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTE: trajectory_io=false is deprecated and should not be used.</td>
</tr>
<tr>
<td></td>
<td>.false.: use disk I/O in 4D-Var for data exchange</td>
<td></td>
</tr>
<tr>
<td>var4d_detail_out</td>
<td>false</td>
<td>.true.: output extra diagnostics for debugging 4D-Var</td>
</tr>
</tbody>
</table>

### b. OBSPROC namelist variables

**Variable Names**

<table>
<thead>
<tr>
<th>&amp;record1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obs_gts_filename</td>
<td>name and path of decoded observation file</td>
</tr>
<tr>
<td>fg_format</td>
<td>'MM5' for MM5 application, 'WRF' for WRF application</td>
</tr>
<tr>
<td>obser.err.txt</td>
<td>name and path of observational error file</td>
</tr>
<tr>
<td>first_guess_file</td>
<td>name and path of the first guess file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&amp;record2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_window_min</td>
<td>The earliest time edge as ccyy-mm-dd_hh:mm:ss</td>
</tr>
<tr>
<td>time_analysis</td>
<td>The analysis time as ccyy-mm-dd_hh:mm:ss</td>
</tr>
<tr>
<td>time_window_max</td>
<td>The latest time edge as ccyy-mm-dd_hh:mm:ss</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Only observations between [time_window_min, time_window_max] will kept.</td>
</tr>
</tbody>
</table>

**&record3       | Description                                                                 |
| max_number_of_obs | Maximum number of observations to be loaded, i.e. in domain and time window, this is independent of the number of obs actually read. |
| fa-              | .TRUE.: will stop when more than max_number_of_obs are loaded            |
| tal_if_exceed_max_obs | .FALSE.: will process the first max_number_of_obs loaded observations.  |

**&record4       | Description                                                                 |
| qc_test_vert_consiste | .TRUE. will perform a vertical consistency quality control check    |
**ncy**

qc_test_convective_adj .TRUE. will perform a convective adjustment quality control check on sounding

qc_test_above_lid .TRUE. will flag the observation above model lid

remove_above_lid .TRUE. will remove the observation above model lid

domain_check_h .TRUE. will discard the observations outside the domain

Thining_SATO .FALSE.: no thinning for SATOB data.

Thining_SSMI .TRUE.: thinning procedure applied to SSMI data.

Thining_QSCAT .FALSE.: no thinning for SMMI data.

&record5

print_gts_read .TRUE. will write diagnostic on the decoded obs reading in file obs_gts_read.diag

print_gpspw_read .TRUE. will write diagnostic on the gpsppw obs reading in file obs_gpspw_read.diag

print_recoverp .TRUE. will write diagnostic on the obs pressure recovery in file obs_recover_pressure.diag

print_duplicate_loc .TRUE. will write diagnostic on space duplicate removal in file obs_duplicate_loc.diag

print_duplicate_time .TRUE. will write diagnostic on time duplicate removal in file obs_duplicate_time.diag

print_recoverh .TRUE will write diagnostic on the obs height recovery in file obs_recover_height.diag

print_qc_vert .TRUE will write diagnostic on the vertical consistency check in file obs_qc1.diag

print_qc_conv .TRUE will write diagnostic on the convective adjustment check in file obs_qc1.diag

print_qc_lid .TRUE. will write diagnostic on the above model lid height check in file obs_qc2.diag

print_uncomplete .TRUE. will write diagnostic on the uncompleted obs removal in file obs_uncomplete.diag

user_defined_area .TRUE.: read in the record6: x_left, x_right, y_top, y_bottom,

FALSE.: not read in the record6.

&record6

x_left West border of sub-domain, not used

x_right East border of sub-domain, not used

y_bottom South border of sub-domain, not used

y_top North border of sub-domain, not used

ptop Reference pressure at model top

ps0 Reference sea level pressure

base_pres Same as ps0. User must set either ps0 or base_pres.

t0 Mean sea level temperature

base_temp Same as t0. User must set either t0 or base_temp.

tlp Temperature lapse rate
base_lapse
pis0
base_tropo_pres
tis0
base_start_temp

&record7
IPROJ
PHIC
XLONC
TRUELAT1
TRUELAT2
MOAD_CEN_LAT
STANDARD_LON

&record8
IDD
MAXNES
NESTIX
NESTJX
DIS
NUMC
NESTI
NESTJ

&record9
prepbufr_output_filename
prepbufr_table_filename
output_ob_format
use_for
num_slots_past
num_slots_ahead
write_synop
write_ship

WRFDA
write_metar  If keep metar obs in obs_gts (ASCII) files.
write_buoy   If keep buoy obs in obs_gts (ASCII) files.
write_pilot  If keep pilot obs in obs_gts (ASCII) files.
write_sound  If keep sound obs in obs_gts (ASCII) files.
write_amdar  If keep amdar obs in obs_gts (ASCII) files.
write_satem  If keep satem obs in obs_gts (ASCII) files.
write_satob  If keep satob obs in obs_gts (ASCII) files.
write_airep  If keep airep obs in obs_gts (ASCII) files.
write_gpspw  If keep gpspw obs in obs_gts (ASCII) files.
write_gpsztd If keep gpsztd obs in obs_gts (ASCII) files.
write_gpsref If keep gpsref obs in obs_gts (ASCII) files.
write_gpseph If keep gpseph obs in obs_gts (ASCII) files.
write_ssmt1  If keep ssmt1 obs in obs_gts (ASCII) files.
write_ssmt2  If keep ssmt2 obs in obs_gts (ASCII) files.
write_ssmi   If keep ssmi obs in obs_gts (ASCII) files.
write_tovs   If keep tovs obs in obs_gts (ASCII) files.
write_qscat  If keep qscat obs in obs_gts (ASCII) files.
write_prof1  If keep profile obs in obs_gts (ASCII) files.
write_bogus  If keep bogus obs in obs_gts (ASCII) files.
write_airs   If keep airs obs in obs_gts (ASCII) files.
Chapter 7: Objective Analysis (OBSGRID)

Table of Contents

- Introduction
- Program Flow
- Source of Observations
- Objective Analysis techniques in OBSGRID
- Quality Control for Observations
- Additional Observations
- Surface FDDA option
- Objective Analysis on Model Nests
- How to run OBSGRID
- Output Files
- Plot Utilities
- Observations Format
- OBSGRID Namelist

Introduction

The goal of objective analysis in meteorological modeling is to improve meteorological analyses (the first guess) on the mesoscale grid by incorporating information from observations. Traditionally, these observations have been "direct" observations of temperature, humidity, and wind from surface and radiosonde reports. As remote sensing techniques come of age, more and more "indirect" observations are available for researchers and operational modelers. Effective use of these indirect observations for objective analysis is not a trivial task. Methods commonly employed for indirect observations include three-dimensional or four-dimensional variational techniques ("3DVAR" and "4DVAR", respectively), which can be used for direct observations as well.

This chapter discusses the objective analysis program, OBSGRID. Discussion of variational techniques (WRFDA) can be found in Chapter 6 of this User’s Guide.

The analyses input to OBSGRID as the first guess are analyses output from the METGRID part of the WPS package (see Chapter 3 of this User’s Guide for details regarding the WPS package).
OBSGRID capabilities include:

- Choice of Cressman-style or Multiquadric objective analysis.
- Various tests to screen the data for suspect observations.
- Procedures to input bogus data.
- Expanded Grid: OBSGRID has the capability to cut the input model domain down on output. This feature allows you to incorporate data from outside your intended grid to improve analyses near the boundaries. To use this feature, a user must create a larger domain than the final intended domain when running WPS.

**Program Flow**

OBSGRID is run directly after metgrid.exe, and uses the `met_em*` output files from metgrid.exe as input. OBSGRID also requires additional observations (A) as input. The format of these observational files is described in the Observations Format section of this chapter.
Output from the objective analysis programs can be used to:

- Provide fields for Initial and Boundary conditions (1). Note that the files metoa_em* are formatted identically to the met_em* files from metgrid.exe. The only difference is that the fields in these files now incorporate observational information.
- Provide surface fields for surface-analysis-nudging FDDA (2). Note, when using the wrfsfdda file as input to WRF, it is also recommended to use the 3-D fdda file (wrfsfdda (5) – which is an optional output created when running real.exe) as input to WRF.
- Provide data for observational nudging (3). Note: since version 3.1.1 of OBSGRID this file can be read directly by the observational nudging code and no longer needs to pass through an additional perl script.
- Provide ASCII and netCDF output (4). These files provide information regarding the observations used and the quality control flags assigned. The information in these files can also be plotted with the provided plotting utilities.

### Source of Observations

OBSGRID reads observations provided by the user in formatted ASCII text files. This allows users to adapt their own data to use as input to the OBSGRID program. This format (wrf_obs / little_r format) is the same format used in the MM5 objective analysis program LITTLE_R (hence the name).

Programs are available to convert NMC ON29 and NCEP BUFR formatted files (see below) into the wrf_obs / little_r format. Users are responsible for converting other observations they may want to provide to OBSGRID into this format. A user-contributed (i.e., unsupported) program is available in the utils/ directory for converting observation files from the GTS to wrf_obs / little_r format.

**NCEP operational global surface and upper-air observation subsets, as archived by the Data Support Section (DSS) at NCAR.**

- Upper-air data in NMC ON29 format (from early 1970s to early 2000)
  [http://rda.ucar.edu/datasets/ds353.4/](http://rda.ucar.edu/datasets/ds353.4/)
- Surface data in NMC ON29 format (from early 1970s to early 2000)
  [http://rda.ucar.edu/datasets/ds464.0/](http://rda.ucar.edu/datasets/ds464.0/)
- Upper-air data in NCEP BUFR format (from 1999 to present)
  [http://rda.ucar.edu/datasets/ds351.0/](http://rda.ucar.edu/datasets/ds351.0/)
- Surface data in NCEP BUFR format (from 1999 to present)
  [http://rda.ucar.edu/datasets/ds461.0/](http://rda.ucar.edu/datasets/ds461.0/)

The newer data (ds351.0 and ds461.0) is also available in the little_r format. From outside NCAR, this data can be download from the web, while it is available on the NCAR /glade system for NCAR supercomputer users. This data is sorted into 6-hourly windows, which are typically too large for use in OBSGRID. To reorder this into 3-hourly windows:
• Get the little_r 6-hourly data
  o Non-NCAR super-computer users. Get the data directly from the above web sites. Combine (by using the Unix ‘cat’ command) all the surface and upper-air data into one large file called rda_obs.
  o NCAR super-computer users. Use the script util/get_rda_data.csh, to get the data and create the file rda_obs. You will need to edit this script to supply the date range that you are interested in.
• Compile the Fortran program util/get_rda_data.f. Place rda_obs file the in the top OBSGRID directory. Run the util/get_rda_data.exe executable. This executable will use the date range from namelist.oa, and create 3-hourly OBS:<date> files which are ready to use in OBSGRID.

NMC Office Note 29 can be found in many places on the World Wide Web, including: http://www.emc.ncep.noaa.gov/mmb/data_processing/on29.htm

Another method of obtaining little_r observations is to download observations from the Meteorological Assimilation Data Ingest System (MADIS; https://madis.noaa.gov/) and convert them to little_r format using the MADIS2LITTLER tool provided by NCAR (http://www2.mmm.ucar.edu/wrf/users/wrfda/download/madis.html). Note that to allow single-level above-surface observations to be properly dealt with by OBSGRID, MADIS2LITTLER must be modified to mark such observations as soundings (in module_output.F, subroutine write_littler_onelvl must be modified to set is_sound = .TRUE.).

Objective Analysis techniques in OBSGRID

Cressman Scheme

Three of the four objective analysis techniques used in OBSGRID are based on the Cressman scheme, in which several successive scans nudge a first-guess field toward the neighboring observed values.

The standard Cressman scheme assigns to each observation a circular radius of influence, R. The first-guess field at each grid point, P, is adjusted by taking into account all the observations that influence P.

The differences between the first-guess field and the observations are calculated, and a distance-weighted average of these difference values is added to the value of the first-guess at P. Once all grid points have been adjusted, the adjusted field is used as the first guess for another adjustment cycle. Subsequent passes each use a smaller radius of influence.
**Ellipse Scheme**

In analyses of wind and relative humidity (fields strongly deformed by the wind) at pressure levels, the circles from the standard Cressman scheme are elongated into ellipses, oriented along the flow. The stronger the wind, the greater the eccentricity of the ellipses. This scheme reduces to the circular Cressman scheme under low-wind conditions.

**Banana Scheme**

In analyses of wind and relative humidity at pressure levels, the circles from the standard Cressman scheme are elongated in the direction of the flow, and curved along the streamlines. The result is a banana shape. This scheme reduces to the Ellipse scheme under straight-flow conditions, and the standard Cressman scheme under low-wind conditions.
**Multiquadric scheme**

The Multiquadric scheme uses hyperboloid radial basis functions to perform the objective analysis. Details of the multiquadric technique may be found in Nuss and Titley, 1994: "Use of multiquadric interpolation for meteorological objective analysis." *Mon. Wea. Rev.*, 122, 1611-1631. Use this scheme with caution, as it can produce some odd results in areas where only a few observations are available.

**Quality Control for Observations**

A critical component of OBSGRID is the screening for bad observations. Many of these QC checks are optional in OBSGRID.

**Quality Control on Individual Reports**

- Gross Error Checks (same values, pressure decreases with height, etc.)
- Remove spikes from temperature and wind profiles.
- Adjust temperature profiles to remove superadiabatic layers.
- No comparisons to other reports or to the first-guess field.

**The ERRMAX test**

The ERRMAX quality-control check is optional, but highly recommended.

- Limited user control over data removal. The user may set thresholds, which vary the tolerance of the error check.
- Observations are compared to the first-guess field.
- If the difference value (obs - first-guess) exceeds a certain threshold, the observation is discarded.
- Threshold varies depending on the field, level, and time of day.
- Works well with a good first-guess field.
The Buddy test

The Buddy check is optional, but highly recommended.

- Limited user control over data removal. The user may set weighting factors, which vary the tolerance of the error check.
- Observations are compared to both the first guess and neighboring observations.
- If the difference value of an observation (obs - first-guess) varies significantly from the distance-weighted average of the difference values of neighboring observations, the observation is discarded.
- Works well in regions with good data density.

Additional Observations

Input of additional observations, or modification of existing (and erroneous) observations, can be a useful tool at the objective analysis stage.

In OBSGRID, additional observations are provided to the program the same way (in the same \textit{wrf\_obs / little\_r format}) as standard observations. Additional observations must be in the same file as the rest of the observations. Existing (erroneous) observations can be modified easily, as the observations input format is ASCII text. Identifying an observation report as "bogus" simply means that it is assumed to be good data, but no quality control is performed for that report.

Surface FDDA option

The surface FDDA option creates additional analysis files for the surface only, usually with a smaller time interval between analyses (\textit{i.e., more frequently}) than the full upper-air analyses. The purpose of these surface analysis files is for later use in WRF with the surface analysis nudging option.

The LAGTEM option controls how the first-guess field is created for surface analysis files. Typically, the surface and upper-air first-guess (\textit{analysis times}) is available at twelve-hour or six-hour intervals, while the surface analysis interval may be 3 hours (10800 seconds). So at analysis times, the available surface first-guess is used. If LAGTEM is set to \texttt{FALSE}, the surface first-guess at other times will be temporally interpolated from the first-guess at the analysis times. If LAGTEM is set to \texttt{TRUE}, the surface first guess at other times is the objective analysis from the previous time.
Objective Analysis on Model Nests

OBSGRID has the capability to perform the objective analysis on a nest. This is done manually with a separate OBSGRID process, performed on met_em_d0x files for the particular nest. Often, however, such a step is unnecessary; it complicates matters for the user and may introduce errors into the forecast. At other times, extra information available to the user, or extra detail that objective analysis may provide on a nest, makes objective analysis on a nest a good option.

The main reason to do objective analysis on a nest is if you have observations available with horizontal resolution somewhat greater than the resolution of your coarse domain. There may also be circumstances in which the representation of terrain on a nest allows for better use of surface observations \(i.e., \text{the model terrain better matches the real terrain elevation of the observation}\).

The main problem introduced by doing objective analysis on a nest is inconsistency in initial conditions between the coarse domain and the nest. Observations that fall just outside a nest will be used in the analysis of the coarse domain, but discarded in the analysis of the nest. With different observations used right at a nest boundary, one can get very different analyses.

How to run OBSGRID

Get the source code

The source code can be downloaded from:
http://www2.mmm.ucar.edu/wrf/users/download/get_source.html. Once the tar file is gunzipped (gunzip OBSGRID.TAR.gz), and untared (untar OBSGRID.TAR), it will create an OBSGRID/directory.

```
    cd OBSGRID
```

Generate the executable

The only library that is required to build the WRF model is netCDF. The user can find the source code, precompiled binaries, and documentation at the UNIDATA home page (http://www.unidata.ucar.edu/software/netcdf/).

To successfully compile the utilities `plot_level.exe` and `plot_sounding.exe`, NCAR Graphics needs to be installed on your system. These routines are not necessary to run OBSGRID, but are useful for displaying observations. Since version 3.7.0 NCL scripts are available and therefore these two utilities are no longer needed to plot the data.

To configure, type:

```
    ./configure
```
Choose one of the configure options, then compile.

    ./compile

If successful, this will create the executable obsgrid.exe. Executables plot_level.exe and plot_sounding.exe, will be created if NCAR Graphics is installed.

Prepare the observations files

Preparing observational files is a user responsibility. Some data are available from NCAR’s RDA web site. Data from the early 1970’s are in ON29 format, while data from 1999 to present are in NCEP BUFR format. Help using these datasets are available. For more information see the section Source of Observations on page 7-3 of this Users’ Guide.

A program is also available for reformatting observations from the GTS stream (unsupported). This can be found in OBSGRID/util, and is called gts_cleaner.f. The code expects to find one observational input file per analysis time. Each file should contain both surface and upper-air data (if available).

Edit the namelist for your specific case

The most critical information you'll be changing most often is the start date, end date, and file names.

Pay particularly careful attention to the file name settings. Mistakes in observation file names can go unnoticed because OBSGRID will happily process the wrong files, and if there are no data in the (wrongly-specified) file for a particular time, OBSGRID will happily provide you with an analysis of no observations.

Run the program

Run the program by invoking the command:

    ./obsgrid.exe >& obsgrid.out

Check the obsgrid.out file for information and runtime errors.
Check your output

Examine the `obsgrid.out` file for error messages or warning messages. The program should have created the files called `metoa_em*`. Additional output files containing information about observations found, used and discarded will probably be created, as well.

Important things to check include the number of observations found for your objective analysis, and the number of observations used at various levels. This can alert you to possible problems in specifying observation files or time intervals. This information is included in the printout file.

You may also want to experiment with a couple of simple plot utility programs, discussed below.

There are a number of additional output files, which you might find useful. These are discussed below.

Output Files

The OBSGRID program generates some ASCII/netCDF files to detail the actions taken on observations through a time cycle of the program. In support of users wishing to plot the observations used for each variable (at each level, at each time), a file is created with this information. Primarily, the ASCII/netCDF files are for consumption by the developers for diagnostic purposes. The main output of the OBSGRID program is the gridded, pressure-level data set to be passed to the `real.exe` program (files `metoa_em*`).

In each of the files listed below, the text ".dn.YYYY-MM-DD_HH:mm:ss.tttt" allows each time period that is processed by OBSGRID to output a separate file. The only unusual information in the date string is the final four letters "tttt" which is the decimal time to ten thousandths of a second. These files will be dependent on the domain being processed.

`metoa_em*`

These are the final analysis files at surface and pressure levels. Generating this file is the primary goal of running OBSGRID.

These files can now be used in place of the `met_em*` files from WPS to generate initial and boundary conditions for WRF. To use these files when running `real.exe` you can do one of two things:

1. Rename or link the `metoa_em*` files back to `met_em*`. This way `real.exe` will read the files automatically.
2. Use the `auxinput1_inname` namelist option in WRF's `namelist.input` file to overwrite the default filename `real.exe` uses. To do this, add the following to the
&time_control section of the WRF namelist.input file before running real.exe (use the exact syntax as below – do not substitute the <domain> and <date> for actual numbers):

auxinput1_inname = "metoa_em.d<domain>,<date>"

wrfsfdda_dn

Use of the surface FDDA option in OBSGRID creates a file called wrfsfdda_dn. This file contains the surface analyses at INTF4D intervals, analyses of T, TH, U, V, RH, QV, PSFC, PMSL, and a count of observations within 250 km of each grid point.

Due to the input requirements of the WRF model, data at the current time (_OLD) and data for the next time (_NEW) are supplied at each time interval. Due to this requirement, users must take care to specify the same interval in the WRF fdda section for surface nudging as the interval used in OBSGRID to create the wrfsfdda_dn file. This also means that the user may need to have data available for OBSGRID to create a surface analysis beyond the last analysis actually used by WRF surface analysis nudging. With a positive value for the length of rampdown, even though the _OLD field at the beginning of the rampdown will be nudged throughout the rampdown, WRF still requires a _NEW field at the beginning of the rampdown period.

OBS_DOMAINdxx

These files can be used in WRF for observational nudging. The format of this file is slightly different from the standard wrf_obs / little_r format. See the Observation Nudging User’s Guide or Chapter 5 of this User’s Guide for details on observational nudging.

The “d” in the file name represents the domain number. The “xx” is just a sequential number.

These files contain a list of all of the observations available for use by the OBSGRID program.

- The observations have been sorted and the duplicates have been removed.
- Observations outside of the analysis region have been removed.
- Observations with no information have been removed.
- All reports for each separate location (different levels, but at the same time) have been combined to form a single report.
- Data that has had the "discard" flag internally set (data which will not be sent to the quality control or objective analysis portions of the code) are not listed in this output.
- The data have gone through an expensive test to determine if the report is within the analysis region, and the data have been given various quality control flags. Unless a blatant error in the data is detected (such as a negative sea-level pressure), the observation data are not typically modified, but only assigned quality control flags.
• Data with qc flags higher than a specified value (user controlled, via the namelist), will be set to missing data.

The WRF observational nudging code requires that all observational data are available in a single file called OBS_DOMAINd01 (where d is the domain number), whereas OBSGRID creates one file per time. Therefore, to use these files in WRF, they should first be concatenated to a single file. A script (run_cat_obs_files.csh) is provided for this purpose. By running this script, the original OBS_DOMAINd01 files will be moved to OBS_DOMAINd01_sav, and a new OBS_DOMAINd01 file (containing all the observations for all times) will be created. This new file can be used directly in the WRF observational nudging code.

**qc_obs_raw.dn.YYYY-MM-DD_HH:mm:ss.tttt(.nc)**

This file contains a listing of all of the observations available for use by the OBSGRID program.

• The observations have been sorted and the duplicates have been removed.
• Observations outside of the analysis region have been removed.
• Observations with no information have been removed.
• All reports for each separate location (different levels, but at the same time) have been combined to form a single report.
• Data that has had the "discard" flag internally set (data which will not be sent to the quality control or objective analysis portions of the code) are not listed in this output.
• The data have gone through an expensive test to determine if the report is within the analysis region, and the data have been given various quality control flags. Unless a blatant error in the data is detected (such as a negative sea-level pressure), the observation data are not typically modified, but only assigned quality control flags.
• Two files are available, both containing identical information. One is the older ASCII format, while the other is in netCDF format.
• *The data in the ASCII file can be used as input to the plotting utility plot_sounding.exe*
• *The netCDF file can be used to plot both station data (util/station.ncl) and sounding data (util/sounding.ncl). This is available since version 3.7 and is the recommended option.*

**qc_obs_used.dn.YYYY-MM-DD_HH:mm:ss.tttt(.nc)**

These files are similar to the above “raw” files, and can be used in the same way. But in this case it contains the data used by the OBSGRID program, which are also the data saved to the OBS_DOMAINdxx files.

**qc_obs_used_earth_relative.dn.YYYY-MM-DD_HH:mm:ss.tttt(.nc)**

These files are identical to the above "qc_obs_used" files except that the winds are in an earth-relative framework rather than a model-relative framework. The non-netCDF version of these
files can be used as input for the Model Evaluation Tools (MET; http://www.dtcenter.org/met/users/).

plotobs_out.dn.YYYY-MM-DD_HH:mm:ss.tttt

This file lists data by variable and by level, where each observation that has gone into the objective analysis is grouped with all of the associated observations for plotting or some other diagnostic purpose. The first line of this file is the necessary FORTRAN format required to input the data. There are titles over the data columns to aid in the information identification. Below are a few lines from a typical file. This data can be used as input to the plotting utility plot_level.exe. But since version 3.7, it is recommended to use the station.ncl script that uses the data in the new netCDF data files.

```
Number of Observations 00001214
Variable Press  Obs    Station Obs        Obs-1st   X         Y         QC
Name     Level  Number ID      Value      Guess     Location  Location  Value
U        1001   1      CYYT    6.39806    4.67690   161.51    122.96    0
U        1001   2      CWRA    2.04794    0.89164   162.04    120.03    0
U        1001   3      CWVA    1.30433 -1.80660   159.54    125.52    0
U        1001   4      CWAR    1.20569    1.07567   159.53    121.07    0
U        1001   5      CYQX    0.470500 -2.10306   156.58    125.17    0
U        1001   6      CWDO    0.789376  2.14755   157.37    118.95    0
```

Plot Utilities

The OBSGRID package provides two utility programs for plotting observations. These programs are called plot_soundings.exe and plot_levels.exe. These optional programs use NCAR Graphics to build, which is often problematic. Two new NCL scripts are provided instead, sounding.ncl and station.ncl. Using these as opposed to the Fortran code are recommended.

```
sounding.ncl / plot_soundings.exe
```

The script util/sounding.ncl plots soundings. This script generates soundings from the netCDF files qc_obs_raw.dn.YYYY-MM-DD_HH:mm:ss.tttt.nc and qc_obs_used.dn.YYYY-MM-DD_HH:mm:ss.tttt.nc. Only data that are on the requested analysis levels are processed.

By default the script will plot the data from all the “qc_obs_used” files in the directory. This can be customized through the use of command line setting. For example:

```
ncl ./util/sounding.ncl 'qcOBS="raw"
```

will plot data from the “qc_obs_raw” files
ncl util/soundin.g.ncl YYYY=2010 MM=6
will plot data from the “qc.obs_used” files for June 2010

Available command line options are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qcOBS</td>
<td>Dataset to use. Options are “raw” or “used”. Default is “used”</td>
</tr>
<tr>
<td>YYYY</td>
<td>Integer year to plot. Default is all available years.</td>
</tr>
<tr>
<td>MM</td>
<td>Integer month to plot. Default is all available months.</td>
</tr>
<tr>
<td>DD</td>
<td>Integer day to plot. Default is all available days.</td>
</tr>
<tr>
<td>HH</td>
<td>Integer hour to plot. Default is all available hours.</td>
</tr>
</tbody>
</table>
| outTYPE   | Output type. Default is plotting to the screen, i.e., “x11”. Other options are “pdf” or “ps”. The script creates the following output files(s):

qc_obs <qcOBS>.sounding.<date>.<outTYPE> for instance:
qc_obs_used.sounding.2010-03-06_09.pdf |

The older program plot_soundings.exe also plots soundings. This program generates soundings from the qc_obs_raw.dn/YYYY-MM-DD HH:mm:ss.tttt and qc_obs_used.dn/YYYY-MM-DD HH:mm:ss.tttt data files. Only data that are on the requested analysis levels are processed. The program uses information from &record1, &record2 and &plot_sounding in the namelist.oa file to generate the required output. The program creates output file(s): sounding_<file_type>_<date>.cgm

plot_level.exe

The script util/station.ncl creates station plots for each analysis level. These plots contain both observations that have passed all QC tests and observations that have failed the QC tests. Observations that have failed the QC tests are plotted in various colors according to which test failed. This script generates soundings from the netCDF files qc_obs_raw.dn/YYYY-MM-DD HH:mm:ss.tttt.nc and qc_obs_used.dn/YYYY-MM-DD HH:mm:ss.tttt.nc.

By default the script will plot the data from all the “qc.obs_used” files in the directory. This can be customized through the use of command line setting. For example:

```
ncl /util/station.ncl 'qcOBS="raw"'
will plot data from the “qc.obs.raw” files
ncl util/station.ncl YYYY=2010 MM=6
will plot data from the “qc.obs_used” files for June 2010
```

Available command line options are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qcOBS</td>
<td>Dataset to use. Options are “raw” or “used”. Default is “used”</td>
</tr>
<tr>
<td>YYYY</td>
<td>Integer year to plot. Default is all available years.</td>
</tr>
<tr>
<td>MM</td>
<td>Integer month to plot. Default is all available months.</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>DD</td>
<td>Integer day to plot. Default is all available days.</td>
</tr>
<tr>
<td>HH</td>
<td>Integer hour to plot. Default is all available hours.</td>
</tr>
<tr>
<td>outTYPE</td>
<td>Output type. Default is plotting to the screen, i.e., “x11”. Other options are “pdf” or “ps”. The script creates the following output files(s): qc_obs_&lt;qcOBS&gt;.station.&lt;date&gt;.&lt;outTYPE&gt; for instance: qc_obs_used.station.2010-03-06_09.pdf</td>
</tr>
</tbody>
</table>

The older program plot_level.exe creates station plots for each analysis level. These plots contain both observations that have passed all QC tests and observations that have failed the QC tests. Observations that have failed the QC tests are plotted in various colors according to which test failed. The program uses information from &record1 and &record2 in the namelist.oa file to generate plots from the observations in the file plotobs_out.dn.YYYY-MM-DD_HH:mm:ss.tttt. The program creates the file(s): levels_<date>.cgm.

**Observations Format**

To make the best use of the OBSGRID program, it is important for users to understand the *wrf_obs/little_r* Observations Format.

Observations are conceptually organized in terms of reports. A report consists of a single observation or set of observations associated with a single latitude/longitude coordinate.

Examples

- a surface station report including observations of temperature, pressure, humidity, and winds.
- an upper-air station's sounding report with temperature, humidity, and wind observations at many height or pressure levels.
- an aircraft report of temperature at a specific lat/lon/height.
- a satellite-derived wind observation at a specific lat/lon/height.

Each report in the *wrf_obs/little_r* Observations Format consists of at least four records:

- A report header record
- one or more data records
- an end data record
- an end report record.

The report header record is a 600-character-long record (*much of which is unused and needs only dummy values*) that contains certain information about the station and the report as a whole.
(location, station id, station type, station elevation, etc.). The report header record is described fully in the following table. Shaded items in the table are unused:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Fortran I/O Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude</td>
<td>F20.5</td>
<td>station latitude (north positive)</td>
</tr>
<tr>
<td>longitude</td>
<td>F20.5</td>
<td>station longitude (east positive)</td>
</tr>
<tr>
<td>id</td>
<td>A40</td>
<td>ID of station</td>
</tr>
<tr>
<td>name</td>
<td>A40</td>
<td>Name of station</td>
</tr>
<tr>
<td>platform</td>
<td>A40</td>
<td>Description of the measurement device</td>
</tr>
<tr>
<td>source</td>
<td>A40</td>
<td>GTS, NCAR/ADP, BOGUS, etc.</td>
</tr>
<tr>
<td>elevation</td>
<td>F20.5</td>
<td>station elevation (m)</td>
</tr>
<tr>
<td>num_vld_fld</td>
<td>I10</td>
<td>Number of valid fields in the report</td>
</tr>
<tr>
<td>num_error</td>
<td>I10</td>
<td>Number of errors encountered during the decoding of this observation</td>
</tr>
<tr>
<td>num_warning</td>
<td>I10</td>
<td>Number of warnings encountered during decoding of this observation</td>
</tr>
<tr>
<td>seq_num</td>
<td>I10</td>
<td>Sequence number of this observation</td>
</tr>
<tr>
<td>num_dups</td>
<td>I10</td>
<td>Number of duplicates found for this observation</td>
</tr>
<tr>
<td>is_sound</td>
<td>L10</td>
<td>T/F Above-surface or surface (i.e., all non-surface observations should use T, even above-surface single-level obs)</td>
</tr>
<tr>
<td>bogus</td>
<td>L10</td>
<td>T/F bogus report or normal one</td>
</tr>
<tr>
<td>discard</td>
<td>L10</td>
<td>T/F Duplicate and discarded (or merged) report.</td>
</tr>
<tr>
<td>sut</td>
<td>I10</td>
<td>Seconds since 0000 UTC 1 January 1970</td>
</tr>
<tr>
<td>julian</td>
<td>I10</td>
<td>Day of the year</td>
</tr>
<tr>
<td>date_char</td>
<td>A20</td>
<td>YYYYMMDDHHmmss</td>
</tr>
<tr>
<td>slp, qc</td>
<td>F13.5, I7</td>
<td>Sea-level pressure (Pa) and a QC flag</td>
</tr>
<tr>
<td>ref_pres, qc</td>
<td>F13.5, I7</td>
<td>Reference pressure level (for thickness) (Pa) and a QC flag</td>
</tr>
<tr>
<td>ground_t, qc</td>
<td>F13.5, I7</td>
<td>Ground Temperature (T) and QC flag</td>
</tr>
<tr>
<td>sst, qc</td>
<td>F13.5, I7</td>
<td>Sea-Surface Temperature (K) and QC</td>
</tr>
</tbody>
</table>
Following the report header record are the *data records*. These data records contain the observations of pressure, height, temperature, dewpoint, wind speed, and wind direction. There are a number of other fields in the data record that are not used on input. Each data record contains data for a single level of the report. For report types that have multiple levels (*e.g.*, upper-air station sounding reports), each pressure or height level has its own data record. For report types with a single level (*such as surface station reports or a satellite wind observation*), the report will have a single data record. The data record contents and format are summarized in the following table:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Fortran I/O Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pressure, qc</td>
<td>F13.5, I7</td>
<td>Pressure (Pa) of observation, and QC</td>
</tr>
<tr>
<td>height, qc</td>
<td>F13.5, I7</td>
<td>Height (m MSL) of observation, and QC</td>
</tr>
<tr>
<td>temperature, qc</td>
<td>F13.5, I7</td>
<td>Temperature (K) and QC</td>
</tr>
<tr>
<td>dew_point, qc</td>
<td>F13.5, I7</td>
<td>Dewpoint (K) and QC</td>
</tr>
<tr>
<td>speed, qc</td>
<td>F13.5, I7</td>
<td>Wind speed (m/s) and QC</td>
</tr>
<tr>
<td>direction, qc</td>
<td>F13.5, I7</td>
<td>Wind direction (degrees) and QC</td>
</tr>
<tr>
<td>u, qc</td>
<td>F13.5, I7</td>
<td>u component of wind (m/s), and QC</td>
</tr>
<tr>
<td>v, qc</td>
<td>F13.5, I7</td>
<td>v component of wind (m/s), and QC</td>
</tr>
</tbody>
</table>
The *end data record* is simply a data record with pressure and height fields both set to -777777.

After all the data records and the end data record, an *end report record* must appear. The end report record is simply three integers, which really aren't all that important.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Fortran I/O Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>num_vld_fld</td>
<td>I7</td>
<td>Number of valid fields in the report</td>
</tr>
<tr>
<td>num_error</td>
<td>I7</td>
<td>Number of errors encountered during the decoding of the report</td>
</tr>
<tr>
<td>num_warning</td>
<td>I7</td>
<td>Number of warnings encountered during the decoding the report</td>
</tr>
</tbody>
</table>

**QCFlags**

In the observation files, most of the meteorological data fields also have space for an additional integer quality-control flag. The quality-control values are of the form $2^n$, where $n$ takes on positive integer values. This allows the various quality control flags to be additive, yet permits the decomposition of the total sum into constituent components. Following are the current quality control flags that are applied to observations:

\[
\begin{align*}
\text{pressure interpolated from first-guess height} & = 2 ** 1 = 2 \\
\text{pressure int. from std. atmos. and 1st-guess height} & = 2 ** 3 = 8 \\
\text{temperature and dew point both = 0} & = 2 ** 4 = 16 \\
\text{wind speed and direction both = 0} & = 2 ** 5 = 32 \\
\text{wind speed negative} & = 2 ** 6 = 64 \\
\text{wind direction < 0 or > 360} & = 2 ** 7 = 128 \\
\text{level vertically interpolated} & = 2 ** 8 = 256 \\
\text{value vertically extrapolated from single level} & = 2 ** 9 = 512 \\
\text{sign of temperature reversed} & = 2 ** 10 = 1024 \\
\text{superadiabatic level detected} & = 2 ** 11 = 2048 \\
\text{vertical spike in wind speed or direction} & = 2 ** 12 = 4096 \\
\text{convective adjustment applied to temperature field} & = 2 ** 13 = 8192 \\
\text{no neighboring observations for buddy check} & = 2 ** 14 = 16384 \\
\text{data outside normal analysis time and not QC-ed} & = 2 ** 15 = 32768 \\
\text{fails error maximum test} & = 2 ** 16 = 65536
\end{align*}
\]
fails buddy test = $2^{17} = 131072$
observation outside of domain detected by QC = $2^{18} = 262144$

**OBSGRID Namelist**

The OBSGRID namelist file is called "`namelist.oa`", and must be in the directory from which OBSGRID is run. The namelist consists of nine namelist records, named "record1" through "record9", each having a loosely related area of content. Each namelist record, which extends over several lines in the `namelist.oa` file, begins with "&record<#>" (where <#> is the namelist record number) and ends with a slash "/".

The namelist record &plot_sounding is only used by the corresponding utility.

**Namelist record1**

The data in namelist record1 define the analysis times to process:

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_year</td>
<td>2000</td>
<td>4-digit year of the starting time to process</td>
</tr>
<tr>
<td>start_month</td>
<td>01</td>
<td>2-digit month of the starting time to process</td>
</tr>
<tr>
<td>start_day</td>
<td>24</td>
<td>2-digit day of the starting time to process</td>
</tr>
<tr>
<td>start_hour</td>
<td>12</td>
<td>2-digit hour of the starting time to process</td>
</tr>
<tr>
<td>end_year</td>
<td>2000</td>
<td>4-digit year of the ending time to process</td>
</tr>
<tr>
<td>end_month</td>
<td>01</td>
<td>2-digit month of the ending time to process</td>
</tr>
<tr>
<td>end_day</td>
<td>25</td>
<td>2-digit day of the ending time to process</td>
</tr>
<tr>
<td>end_hour</td>
<td>12</td>
<td>2-digit hour of the ending time to process</td>
</tr>
<tr>
<td>interval</td>
<td>21600</td>
<td>Time interval (s) between consecutive times to process</td>
</tr>
</tbody>
</table>

**Namelist record2**

The data in record2 define the model grid and names of the input files:

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid_id</td>
<td>1</td>
<td>ID of domain to process</td>
</tr>
</tbody>
</table>
| obs_filename      | CHARACTER     | Root file name (may include directory information) of the observational files. All input files must have the
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>remove_data_above_qc_flag</td>
<td>200000</td>
<td>Data withqc flags higher than this will not be output to the OBS_DOMAINdxx files. Default is to output all data. Use 65536 to remove data that failed the buddy and error max tests. To also exclude data outside analysis times that could not be QC-ed use 32768 (recommended). This does not affect the data used in the OA process.</td>
</tr>
<tr>
<td>remove_unverified_data</td>
<td>.FALSE.</td>
<td>By setting this parameter to .TRUE. (recommended) any observations that could not be QC'd due to having a pressure insufficiently close to an analysis level will be removed from the OBS_DOMAINdxx files. Obs QC'd by adjusting them to a nearby analysis level or by comparing them to an analysis level within a user-specified tolerance will be included in the OBS_DOMAINdxx files. See use_p_tolerance_one_lev in &amp;record4.</td>
</tr>
<tr>
<td>trim_domain</td>
<td>.FALSE.</td>
<td>Set to .TRUE. if this domain must be cut down on output</td>
</tr>
<tr>
<td>trim_value</td>
<td>5</td>
<td>Value by which the domain will be cut down in each direction</td>
</tr>
</tbody>
</table>
The `met_em*` files which are being processed must be available in the OBSGRID/ directory.

The `obs_filename` and interval settings can get confusing, and deserve some additional explanation. Use of the `obs_filename` files is related to the times and time interval set in namelist `&record1`, and to the F4D options set in namelist `&record8`. The `obs_filename` files are used for the analyses of the full 3D dataset, both at upper levels and the surface. They are also used when F4D=.TRUE.; that is, if surface analyses are being created for surface FDDA nudging. The `obs_filename` files should contain all observations (upper-air and surface) to be used for a particular analysis at a particular time.

Ideally there should be an `obs_filename` for each time period for which an objective analysis is desired. Time periods are processed sequentially from the starting date to the ending date by the time interval, all specified in namelist `&record1`. All observational files must have a date associated with them. If a file is not found, the code will process as if this file contains zero observations, and then continue to the next time period.

If the F4D option is selected, the `obs_filename` files are similarly processed for surface analyses, this time with the time interval as specified by INTF4D.

If a user wishes to include observations from outside the model domain of interest, geogrid.exe (WPS) needs to be run over a slightly larger domain than the domain of interest. Setting `trim_domain` to .TRUE. will cut all 4 directions of the input domain down by the number of grid points set in `trim_value`.

In the example below, the domain of interest is the inner white domain with a total of 100x100 grid points. geogrid.exe has been run for the outer domain (110x110 grid points). By setting the `trim_value` to 5, the output domain will be trimmed by 5 grid points in each direction, resulting in the white 100x100 grid point domain.

![Diagram showing domain trimming](attachment:image.png)

**Namelist record3**
The data in the &record3 concern space allocated within the program for observations. These are values that should not frequently need to be modified:

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_number_of_obs</td>
<td>10000</td>
<td>Anticipated maximum number of reports per time period</td>
</tr>
<tr>
<td>fatal_if_exceed_max_obs</td>
<td>.TRUE.</td>
<td>T/F flag allows the user to decide the severity of not having enough space to store all of the available observation</td>
</tr>
</tbody>
</table>

**Namelist record4**

The data in &record4 set the quality control options. There are four specific tests that may be activated by the user: An error max test; a buddy test; removal of spike, and; the removal of super-adiabatic lapse rates. For some of these tests, the user has control over the tolerances, as well.

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qc_psfc</td>
<td>.FALSE.</td>
<td>Execute error max and buddy check tests for surface pressure observations (temporarily converted to sea level pressure to run QC)</td>
</tr>
</tbody>
</table>

**Error Max Test:** For this test there is a threshold for each variable. These values are scaled for time of day, surface characteristics and vertical level.

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qc_test_error_max</td>
<td>.TRUE.</td>
<td>Check the difference between the first-guess and the observation</td>
</tr>
<tr>
<td>max_error_t</td>
<td>10</td>
<td>Maximum allowable temperature difference (K)</td>
</tr>
<tr>
<td>max_error_uv</td>
<td>13</td>
<td>Maximum allowable horizontal wind component difference (m/s)</td>
</tr>
<tr>
<td>max_error_z</td>
<td>8</td>
<td>Not used</td>
</tr>
<tr>
<td>max_error_rh</td>
<td>50</td>
<td>Maximum allowable relative humidity difference (%)</td>
</tr>
<tr>
<td>max_error_p</td>
<td>600</td>
<td>Maximum allowable sea-level pressure difference (Pa)</td>
</tr>
<tr>
<td>max_error_dewpoint</td>
<td>20</td>
<td>Maximum allowable dewpoint difference (K)</td>
</tr>
</tbody>
</table>
**Buddy Check Test:** For this test there is a threshold for each variable. These values are similar to standard deviations.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qc_test_buddy</td>
<td>.TRUE.</td>
<td>Check the difference between a single observation and neighboring observations</td>
</tr>
<tr>
<td>max_buddy_t</td>
<td>8</td>
<td>Maximum allowable temperature difference (K)</td>
</tr>
<tr>
<td>max_buddy_uv</td>
<td>8</td>
<td>Maximum allowable horizontal wind component difference (m/s)</td>
</tr>
<tr>
<td>max_buddy_z</td>
<td>8</td>
<td>Not used</td>
</tr>
<tr>
<td>max_buddy_rh</td>
<td>40</td>
<td>Maximum allowable relative humidity difference (%)</td>
</tr>
<tr>
<td>max_buddy_p</td>
<td>800</td>
<td>Maximum allowable sea-level pressure difference (Pa)</td>
</tr>
<tr>
<td>max_buddy_dewpoint</td>
<td>20</td>
<td>Maximum allowable dewpoint difference (K)</td>
</tr>
<tr>
<td>buddy_weight</td>
<td>1.0</td>
<td>Value by which the buddy thresholds are scaled</td>
</tr>
</tbody>
</table>

**Spike removal**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qc_test_vert_consistency</td>
<td>.FALSE.</td>
<td>Check for vertical spikes in temperature, dew point, wind speed and wind direction</td>
</tr>
</tbody>
</table>

**Removal of super-adiabatic lapse rates**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qc_test_convective_adj</td>
<td>.FALSE.</td>
<td>Remove any super-adiabatic lapse rate in a sounding by conservation of dry static energy</td>
</tr>
</tbody>
</table>

For satellite and aircraft observations, data are often horizontally spaced with only a single vertical level. The following entries determine how such data are dealt with and are described in more detail below the table.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_p_tolerance_one_lev</td>
<td>.FALSE.</td>
<td>Should single-level above-surface observations be directly QC'd against nearby levels (.TRUE.) or extended to nearby levels (.FALSE.)</td>
</tr>
<tr>
<td>max_p_tolerance_one_lev_qc</td>
<td>700</td>
<td>Pressure tolerance within which QC can be applied directly (Pa)</td>
</tr>
</tbody>
</table>
Dewpoint quality control:
Note that the dewpoint error max check and buddy check are using the same moisture field as the relative humidity checks. The dewpoint checks are to allow for an additional level of quality control on the moisture fields and may be helpful for dry observations where RH differences may be small but dewpoint differences are much larger. The maximum dewpoint thresholds are scaled based on the observed dewpoint to increase the threshold for dry conditions where larger dewpoint variations are expected. If the user does not wish to use dewpoint error checks, simply set the thresholds to very large values.

Quality control of single-level above-surface observations:

Option 1: use_p_tolerance_one_lev = .FALSE.;
For single-level above-surface observations marked as 'FM-88 SATOB' or 'FM-97 AIREP', the observations are adjusted to the nearest pressure level. If the observation's pressure is within max_p_extend_t Pa of the nearest first-guess level, the temperature of the observation is adjusted to the first-guess level using a standard lapse rate, otherwise the temperature is marked as missing. If the observation’s pressure is within max_p_extend_w Pa of the nearest first-guess level, the winds are used without adjustment. The dewpoint is marked as missing regardless of the pressure of the observation. The pressure of the observation is changed to be the pressure of the pressure level against which it is being quality controlled. If a single-level above-surface observation is marked as anything other than ‘FM-88 SATOB’ or ‘FM-97 AIREP’, it appears that it will not be quality controlled unless its pressure happens to exactly match one of the pressure levels in the first guess field. Note that max_p_tolerance_one_lev_qc is ignored if use_p_tolerance_one_lev = .FALSE.

Option 2: use_p_tolerance_one_lev = .TRUE.;
For all single-level above-surface observations, the observations will be quality controlled as long as the closest first-guess field is within max_p_tolerance_one_lev_qc Pa of the observation. In order to allow all single-level above-surface observations to be close enough to a first-guess pressure level that quality control directly comparing the closest pressure level to the observation is valid, the user may need to interpolate the first guess to additional pressure levels prior to ingestion into OBSGRID. OBSGRID will print out the pressure ranges for which error max quality control is not available (i.e., the pressures for which single-level above-surface observations will not be quality controlled). See max_p_tolerance_one_lev_oa in namelist record9 for the equivalent pressure tolerance for creating objective analyses. Note that max_p_extend_t and max_p_extend_w are ignored if use_p_tolerance_one_lev
Namelist record5

The data in &record5 control the enormous amount of printout that may be produced by the OBSGRID program. These values are all logical flags, where TRUE will generate output and FALSE will turn off output.

- print_obs_files
- print_found_obs
- print_header
- print_analysis
- print_qc_vert
- print_qc_dry
- print_error_max
- print_buddy
- print_oa

Namelist record7

The data in &record7 concern the use of the first-guess fields and surface FDDA analysis options. Always use the first guess.

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_first_guess</td>
<td>.TRUE.</td>
<td>Always use first guess (use_first_guess=.TRUE.)</td>
</tr>
<tr>
<td>f4d</td>
<td>.TRUE.</td>
<td>Turns on (.TRUE.) or off (.FALSE.) the creation of surface analysis files.</td>
</tr>
<tr>
<td>intf4d</td>
<td>10800</td>
<td>Time interval in seconds between surface analysis times</td>
</tr>
<tr>
<td>lagtem</td>
<td>.FALSE.</td>
<td>Use the previous time-period's final surface analysis for this time-period's first guess (lagtem=.TRUE.); or Use a temporal interpolation between upper-air times as the first guess for this surface analysis (lagtem = .FALSE.)</td>
</tr>
</tbody>
</table>

Namelist record8

The data in &record8 concern the smoothing of the data after the objective analysis. Note, only the differences fields (observation minus first-guess) of the analyzed are smoothed, not the full fields.

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smooth_type</td>
<td>1</td>
<td>1 = five point stencil of 1-2-1 smoothing; 2 = smoother-desmoother</td>
</tr>
</tbody>
</table>
smooth_sfc_wind 0 Number of smoothing passes for surface winds
smooth_sfc_temp 0 Number of smoothing passes for surface temperature
smooth_sfc_rh 0 Number of smoothing passes for surface relative humidity
smooth_sfc_slp 0 Number of smoothing passes for sea-level pressure
smooth_upper_wind 0 Number of smoothing passes for upper-air winds
smooth_upper_temp 0 Number of smoothing passes for upper-air temperature
smooth_upper_rh 0 Number of smoothing passes for upper-air relative humidity

Namelist record9

The data in &record9 concern the objective analysis options. There is no user control to select the various Cressman extensions for the radius of influence (circular, elliptical or banana). If the Cressman option is selected, ellipse or banana extensions will be applied as the wind conditions warrant.

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oa_type</td>
<td>“Cressman”</td>
<td>“MQD” for multiquadric; “Cressman” for the Cressman-type scheme, “None” for no analysis, this string is case sensitive</td>
</tr>
<tr>
<td>oa_3D_type</td>
<td>“Cressman”</td>
<td>Set upper-air scheme to “Cressman”, regardless of the scheme used at the surface</td>
</tr>
<tr>
<td>oa_3D_option</td>
<td>0</td>
<td>How to switch between “MQD” and “Cressman” if not enough observations are available to perform “MQD”</td>
</tr>
<tr>
<td>mqd_minimum_num_obs</td>
<td>30</td>
<td>Minimum number of observations for MQD</td>
</tr>
<tr>
<td>mqd_maximum_num_obs</td>
<td>1000</td>
<td>Maximum number of observations for MQD</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>radius_influence</td>
<td>5,4,3,2</td>
<td>Radius of influence in grid units for Cressman scheme</td>
</tr>
<tr>
<td>radius_influence_sfc_mult</td>
<td>1.0</td>
<td>Multiply above-surface radius of influence by this value to get surface radius of influence</td>
</tr>
<tr>
<td>oa_min_switch</td>
<td>.TRUE.</td>
<td>T = switch to Cressman if too few observations for MQD; F = no analysis if too few observations</td>
</tr>
<tr>
<td>oa_max_switch</td>
<td>.TRUE.</td>
<td>T = switch to Cressman if too many observations for MQD; F = no analysis if too many observations</td>
</tr>
<tr>
<td>scale_cressman_rh_decrease</td>
<td>.FALSE.</td>
<td>T = decrease magnitude of drying in Cressman analysis; F = magnitude of drying in Cressman analysis unmodified</td>
</tr>
<tr>
<td>oa_psfc</td>
<td>.FALSE.</td>
<td>T = perform surface pressure objective analysis; F = surface pressure only adjusted by sea level pressure analysis</td>
</tr>
<tr>
<td>max_p_tolerance_one_lev_oa</td>
<td>700</td>
<td>Pressure tolerance within which single-level above-surface observations can be used in the objective analysis (Pa)</td>
</tr>
</tbody>
</table>

When `oa_type` is set to *Cressman*, then the *Cressman* scheme will be performed on all data.

When `oa_type` is set to *None*, then no objective analysis will be performed on any data.

When `oa_type` is set to *MQD*, there are a wide variety of options available that control when the code will revert back to the *Cressman* scheme.

- `oa_max_switch ; mqd_maximum_num_obs`
  The code will revert back to *Cressman* if the switch is set to true and the maximum number of observations is exceeded.
  This is to reduce the time the code runs and not for physical reasons.
Recommended to leave switch set to true and just set the maximum number large.

- `oa_min_switch ; mqd_minimum_num_obs`
  The code will revert back to Cressman if the switch is set to true and there are too few observations. How and when the code reverts back to Cressman under these conditions are controlled by the `oa_3D_option` parameter.
  Recommended to leave switch set to true and start with the default minimum settings.

- `oa_3D_type="Cressman"
  All upper-air levels will use the Cressman scheme, regardless of other settings.

The surface will use MQD as long as there are enough observations to do so (`mqd_maximum_num_obs ; mqd_minimum_num_obs`), otherwise it will revert to the Cressman scheme.
Note that if some time periods have enough observations and others do not, the code will only revert to Cressman for the times without sufficient observations.

- `oa_3D_option`
  There are three options (0,1,2). For all these options the surface will use MQD as long as there are enough observations to do so (`mqd_maximum_num_obs ; mqd_minimum_num_obs`), otherwise it will revert to the Cressman scheme.
  Note that if some time periods have enough observations and others do not, the code will only revert to Cressman for the times without sufficient observations.

The upper-air will react as follows:
0 (default): MQD is performed in the upper-air as long as there are enough observations to do so (`mqd_maximum_num_obs ; mqd_minimum_num_obs`). As soon as this is no longer the case, the code will STOP, with suggestions as to which parameters to set to run the code correctly.

1: The code will first check to see if, for a given time, all levels and variables in the upper-air have sufficient observations for the MQD scheme. If not, the code will revert to Cressman for that time period. Note that if some time periods have enough observations and others do not, the code will only revert to Cressman for the times without sufficient observations.

2: The code will check if sufficient observations are available per time, level, and variable for the MQD scheme. If not, the code will revert to the Cressman scheme for that particular time, level and variable. Note this can result in uncontrolled switching between MQD and Cressman. Therefore this option is not recommended.

`radius_influence`
There are three ways to set the radius of influence (RIN) for the Cressman scheme:

- Manually: Set the RIN and number of scans directly. E.g., 5,4,3,2, will result in 4 scans. The first will use 5 grid points for the RIN and the last, 2 points.
• Automatically 1: Set RIN to 0 and the code will calculate the RIN based on the domain size and an estimated observation density of 325 km. By default there will be 4 scans.
• Automatically 2: Set RIN to a negative number and the code will calculate the RIN based on the domain size and an estimated observation density of 325 km. The number of scans is controlled by the value of the set number. E.g., -5 will result in 5 scans.

radius_influence_sfc_mult
The RIN calculated as described above is multiplied by this value to determine the RIN for surface observations. This allows the finer scale structures observed at the surface to be retained. If this multiplication results in a RIN greater than 100 model grid points, then the RIN on the first scan is scaled to be 100 model grid points and all subsequent scans are scale by that same ratio. This is to prevent features from being washed out on fine-scale domains. In order to minimize “spots” on the solution, any scan with a RIN less than 4.5 model grid points is skipped. If this is set to 1.0 then the RIN for surface observations will match the RIN for above-surface observations.

scale_cressman_rh_decreases
This option is meant to mitigate overdrying that can occur when the need for drying diagnosed via an observation at one point is spread to another point where the first guess is already drier than the first guess at the location of the observation. If this option is set to true then drying applied to a point where the first guess is drier than the first guess at the observation location is scaled by the ratio first guess relative humidity at the point the drying is being applied to divided by the first guess relative humidity at the location of the observation.

Note that this scaling is applied on each Cressman scan. See Reen et al. 2016 (http://dx.doi.org/10.1175/JAMC-D-14-0301.1) for further details.

oa_psfc
An objective analysis of surface pressure may allow Obsgrid surface analyses of other fields to be more effectively utilized in WRF if the first-guess surface pressure field is sufficiently coarse compared to the WRF domains (e.g., Reen 2015; http://www.arl.army.mil/arlreports/2015/ARL-TR-7447.pdf). This is because the surface pressure analysis may provide a better estimate of the pressure of the surface analyses and thus WRF is less likely to erroneously reject the surface analyses as being too distant from the actual surface. If there are an insufficient number of observations or if the first-guess surface pressure is not much coarser than WRF, this capability is less likely to add value.

max_p_tolerance_one_lev_oa
If use_p_tolerance_one_lev = .TRUE. in record4, then max_p_tolerance_one_lev_oa is the pressure tolerance (Pa) allowed between single-level above-surface observations and the pressure level they are being used in an objective analysis. If use_p_tolerance_one_lev = .FALSE. in record4, then max_p_tolerance_one_lev_oa is not used by OBSGRID.
Namelist plot_sounding

Only used for the utility plot_sounding.exe

<table>
<thead>
<tr>
<th>Namelist Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_type</td>
<td>“raw”</td>
<td>File to read to produce the plots. Options are “raw” or “used”</td>
</tr>
<tr>
<td>read_metoa</td>
<td>.TRUE.</td>
<td>If set to .TRUE., the model domain information in the metoa_em files will be used to add location information on the plot.</td>
</tr>
</tbody>
</table>
Chapter 8: WRF Software

Table of Contents
- WRF Build Mechanism
- Registry
- I/O Applications Program Interface (I/O API)
- Timekeeping
- Software Documentation
- Performance

WRF Build Mechanism

The WRF build mechanism provides a uniform apparatus for configuring and compiling the WRF model, WRF-Var system and the WRF pre-processors over a range of platforms, with a variety of options. This section describes the components and functioning of the build mechanism. For information on building the WRF code, see the chapter on Software Installation.

Required software:

The WRF build relies on Perl (version 5 or later) and a number of UNIX utilities: csh and Bourne shell, make, M4, sed, awk, and the uname command. A C compiler is needed to compile programs and libraries in the tools and external directories. The WRF code, itself, is mostly standard Fortran (and uses a few 2003 capabilities). For distributed-memory processing, MPI and related tools and libraries should be installed.

Build Mechanism Components:

Directory structure: The directory structure of WRF consists of the top-level directory, plus directories containing files related to the WRF software framework (frame), the WRF model (dyn_em, phys, chem, share), WRF-Var (da), configuration files (arch, Registry), helper and utility programs (tools), and packages that are distributed with the WRF code (external).

Scripts: The top-level directory contains three user-executable scripts: configure, compile, and clean. The configure script relies on the Perl script in arch/Config_new.pl.
Programs: A significant number of WRF lines of code are automatically generated at compile time. The program that does this is tools/registry and it is distributed as part of the source code with the WRF model.

Makefiles: The main Makefile (input to the UNIX make utility) is in the top-level directory. There are also makefiles in most of the subdirectories that come with WRF. Make is called recursively over the directory structure. Make is not directly invoked by the user to compile WRF; the compile script is provided for this purpose. The WRF build has been structured to allow “parallel make”. Before the compile command, the user sets an environment variable, J, to the number of processors to use. For example, to use two processors (in csh syntax):

```
setenv J "-j 2"
```

On some machines, this parallel make causes troubles (a typical symptom is a missing mpif.h file in the frame directory). The user can force that only a single processor to be used with the command:

```
setenv J "-j 1"
```

Configuration files: The configure.wrf contains compiler, linker, and other build settings, as well as rules and macro definitions used by the make utility. The configure.wrf file is included by the Makefiles in most of the WRF source distribution (Makefiles in tools and external directories do not include configure.wrf). The configure.wrf file, in the top-level directory, is generated each time the configure script is invoked. It is also deleted by clean -a. Thus, configure.wrf is the place to make temporary changes, such as optimization levels and compiling with debugging, but permanent changes should be made in the file arch/configure_new.defaults. The configure.wrf file is composed of three files: arch/preamble_new, arch/postamble_new and arch/configure_new.defaults.

The arch/configure_new.defaults file contains lists of compiler options for all the supported platforms and configurations. Changes made to this file will be permanent. This file is used by the configure script to generate a temporary configure.wrf file in the top-level directory. The arch directory also contains the files preamble_new and postamble_new, which constitute the generic parts (non-architecture specific) of the configure.wrf file that is generated by the configure script.

The Registry directory contains files that control many compile-time aspects of the WRF code. The files are named Registry.core (where core is, for example, EM). The configure script copies one of these to Registry/Registry, which is the file that tools/registry will use as input. The choice of core depends on settings to the configure script. Changes to Registry/Registry will be lost; permanent changes should be made to Registry.core. For the WRF ARW model, the file is typically Registry.EM. One of the keywords that the registry program understands is include. The ARW Registry files make use of the REGISTRY.EM_COMMON file.
This reduces the amount of replicated registry information. When searching for variables previously located in a Registry.EM* file, now look in Registry.EM_COMMON.

**Environment variables:** Certain aspects of the configuration and build are controlled by environment variables: the non-standard locations of NetCDF libraries or the Perl command, which dynamic core to compile, machine-specific features, and optional build libraries (such as Grib Edition 2, HDF, and parallel netCDF).

In addition to WRF-related environment settings, there may also be settings specific to particular compilers or libraries. For example, local installations may require setting a variable like MPICH_F90 to make sure the correct instance of the Fortran 90 compiler is used by the mpif90 command.

**How the WRF build works:**

There are two steps in building WRF: configuration and compilation.

**Configuration:** The configure script configures the model for compilation on your system. The configuration first attempts to locate needed libraries, such as netCDF or HDF, and tools, such as Perl. It will check for these in normal places, or will use settings from the user's shell environment. The configure file then calls the UNIX `uname` command to discover what platform you are compiling on. It then calls the Perl script `arch/Config_new.pl`, which traverses the list of known machine configurations and displays a list of available options to the user. The selected set of options is then used to create the `configure.wrf` file in the top-level directory. This file may be edited but changes are temporary, since the file will be deleted by `clean -a`, or overwritten by the next invocation of the configure script. About the only typical option that is included on the configure command is “-d” (for debug). The code builds relatively quickly and has the debugging switches enabled, but the model will run very slowly since all of the optimization has been deactivated. This script takes only a few seconds to run.

**Compilation:** The compile script is used to compile the WRF code after it has been configured using the configure script. This csh script performs a number of checks, constructs an argument list, copies to Registry/Registry the correct Registry.core file for the core being compiled, and the invokes the UNIX `make` command in the top-level directory. The core to be compiled is determined from the user’s environment; if no core is specified in the environment (by setting WRF_core_CORE to 1) the default core is selected (currently the Eulerian Mass core for ARW). The Makefile, in the top-level directory, directs the rest of the build, accomplished as a set of recursive invocations of make in the subdirectories of WRF. Most of these makefiles include the configure.wrf file from the top-level directory. The order of a complete build is as follows:
1. Make in **external** directory
   a. make in `external/io_{grib1,grib_share,int,netcdf}` for
      Grib Edition 1, binary, and netCDF implementations of I/O API
   b. make in `RSL_LITE` directory to build communications layer
      (DM_PARALLEL only)
   c. make in `external/esmf_time_f90` directory to build ESMF time
      manager library
   d. make in `external/fftpack` directory to build FFT library for the
      global filters
   e. make in other external directories, as specified by “**external:**” target
      in the `configure.wrf` file

2. Make in the **tools** directory to build the program that reads the
   Registry/Registry file and auto-generates files in the **inc** directory

3. Make in the **frame** directory to build the WRF framework specific modules

4. Make in the **share** directory to build the non-core-specific mediation layer
   routines, including WRF I/O modules that call the I/O API

5. Make in the **phys** directory to build the WRF model layer routines for physics
   (non core-specific)

6. Make in the **dyn_core** directory for core-specific mediation-layer and model-
   layer subroutines

7. Make in the **main** directory to build the main programs for WRF, symbolic link
   to create executable files (location depending on the build case that was selected
   as the argument to the compile script)

Source files (.F and, in some of the external directories, .F90) are preprocessed to
produce .f90 files, which are input to the compiler. As part of the preprocessing,
Registry-generated files from the **inc** directory may be included. Compiling the .f90
files results in the creation of object (.o) files that are added to the library
main/libwrflib.a. Most of the external directories generate their own library
file. The linking step produces the `wrf.exe` executable and other executables,
depending on the case argument to the compile command: `real.exe` (a preprocessor
for real-data cases) or `ideal.exe` (a preprocessor for idealized cases), and the
`ndown.exe` program, for one-way nesting of real-data cases.
The .o files and .f90 files from a compile are retained until the next invocation of the clean script. The .f90 files provide the true reference for tracking down run time errors that refer to line numbers or for sessions using interactive debugging tools such as dbx or gdb.

Registry

Tools for automatic generation of application code from user-specified tables provide significant software productivity benefits in development and maintenance of large applications, such as WRF. Just for the WRF model, hundreds of thousands of lines of WRF code are automatically generated from a user-edited table, called the Registry. The Registry provides a high-level single-point-of-control over the fundamental structure of the model data, and thus provides considerable utility for developers and maintainers. It contains lists describing state data fields and their attributes: dimensionality, binding to particular solvers, association with WRF I/O streams, communication operations, and run time configuration options (namelist elements and their bindings to model control structures). Adding or modifying a state variable to WRF involves modifying a single line of a single file; this single change is then automatically propagated to scores of locations in the source code the next time the code is compiled.

The WRF Registry has two components: the Registry file (which the user may edit), and the Registry program.

The Registry file is located in the Registry directory and contains the entries that direct the auto-generation of WRF code by the Registry program. There is more than one Registry in this directory, with filenames such as Registry.EM_COMMON (for builds using the Eulerian Mass/ARW core) and Registry.NMM (for builds using the NMM core). The WRF Build Mechanism copies one of these to the file Registry/Registry and this file is used to direct the Registry program. The syntax and semantics for entries in the Registry are described in detail in “WRF Tiger Team Documentation: The Registry” on http://www2.mmm.ucar.edu/wrf/WG2/Tigers/Registry/. The use of the keyword include has greatly reduced the replicated information that was inside the Registry.EM_COMMON file. The Registry program is distributed as part of WRF in the tools directory. It is built automatically (if necessary) when WRF is compiled. The executable file is tools/registry. This program reads the contents of the Registry file, Registry/Registry, and generates files in the inc directory. These include files are inserted (with cpp #include commands) into WRF Fortran source files prior to compilation. Additional information on these is provided as an appendix to “WRF Tiger Team Documentation: The Registry (DRAFT)”. The Registry program itself is written in C. The source files and makefile are in the tools directory.
In addition to the WRF model itself, the Registry/Registry file is used to build the accompanying preprocessors such as real.exe (for real data) or ideal.exe (for ideal simulations), and the ndown.exe program (used for one-way, off-line nesting).

Every variable that is an input or an output field is described in the Registry. Additionally, every variable that is required for parallel communication, specifically associated with a physics package, or needs to provide a tendency to multiple physics or dynamics routines is contained in the Registry. For each of these variables, the index ordering, horizontal and vertical staggering, feedback and nesting interpolation requirements, and the associated IO are defined. For most users, to add a variable into the model requires, regardless of dimensionality, only the addition of a single line to the Registry (make sure that changes are made to the correct Registry.core file, as changes to the Registry file itself are overwritten). Since the Registry modifies code for compile-time options, any change to the Registry REQUIRES that the code be returned to the original unbuilt status with the clean -a command.

The other very typical activity for users is to define new run-time options, which are handled via a Fortran namelist file namelist.input in WRF. As with the model
state arrays and variables, the entire model configuration is described in the Registry. As with the model arrays, adding a new namelist entry is as easy as adding a new line in the Registry.

While the model state and configuration are, by far, the most commonly used features in the Registry, the data dictionary has several other powerful uses. The Registry file provides input to generate all of the communications for the distributed memory processing (halo interchanges between patches, support for periodic lateral boundaries, and array transposes for FFTs to be run in the X, Y, or Z directions). The Registry associates various fields with particular physics packages so that the memory footprint reflects the actual selection of the options, not a maximal value.

Together, these capabilities allow a large portion of the WRF code to be automatically generated. Any code that is automatically generated relieves the developer of the effort of coding and debugging that portion of software. Usually, the pieces of code that are suitable candidates for automation are precisely those that are fraught with “hard to detect” errors, such as communications, indexing, and IO, which must be replicated for hundreds of variables.

Registry Syntax:

Each entry in the Registry is for a specific variable, whether it is for a new dimension in the model, a new field, a new namelist value, or even a new communication. For readability, a single entry may be spread across several lines with the traditional “\” at the end of a line to denote that the entry is continuing. When adding to the Registry, most users find that it is helpful to copy an entry that is similar to the anticipated new entry, and then modify that Registry entry. The Registry is not sensitive to spatial formatting. White space separates identifiers in each entry.

Note: Do not simply remove an identifier and leave a supposed token blank, use the appropriate default value (currently a dash character “-“).

Registry Entries:

The WRF Registry has the following types of entries (not case dependent):

- **Dimspec** – Describes dimensions that are used to define arrays in the model
- **State** – Describes state variables and arrays in the domain structure
- **I1** – Describes local variables and arrays in solve
- **Typedef** – Describes derived types that are subtypes of the domain structure
- **Rconfig** – Describes a configuration (e.g. namelist) variable or array
- **Package** – Describes attributes of a package (e.g. physics)
- **Halo** – Describes halo update interprocessor communications
- **Period** – Describes communications for periodic boundary updates
- **Xpose** – Describes communications for parallel matrix transposes
- **include** – Similar to a CPP #include file
These *keywords* appear as the first word in a line of the file `Registry` to define which type of information is being provided. Following are examples of the more likely Registry types that users will need to understand.

**Registry Dimspec:**

The first set of entries in the Registry is the specifications of the dimensions for the fields to be defined. To keep the WRF system consistent between the dynamical cores and Chemistry, a unified `registry.dimspec` file is used (located in the `Registry` directory). This single file is included into each Registry file, with the keyword `include`. In the example below, three dimensions are defined: `i`, `j`, and `k`. If you do an “ncdump -h” on a WRF file, you will notice that the three primary dimensions are named as “west_east”, “south_north”, and “bottom_top”. That information is contained in this example (the example is broken across two lines, but interleaved).

```plaintext
#<Table>  <Dim>  <Order> <How defined>
dimspec  i      1     standard_domain
dimspec  j      3     standard_domain
dimspec  k      2     standard_domain

<xCoord-axis>  <Dimname in Datasets>
x          west_east
y          south_north
z          bottom_top
```

The WRF system has a notion of horizontal and vertical staggering, so the dimension names are extended with a “_stag” suffix for the staggered sizes. The list of names in the `<Dim>` column may either be a single unique character (for release 3.0.1.1 and prior), or the `<Dim>` column may be a string with no embedded spaces (such as `my_dim`). When this dimension is used later to dimension-ize a `state` or `il` variable, it must be surrounded by curly braces (such as `{my_dim}`). This `<Dim>` variable is not case specific, so for example “i” is the same as an entry for “I”.

**Registry State and IL:**

A `state` variable in WRF is a field that is eligible for IO and communications, and exists for the duration of the model forecast. The `il` variables (intermediate level one) are typically thought of as tendency terms, computed during a single model time-step, and then discarded prior to the next time-step. The space allocation and de-allocation for these `il` variables is automatic (on the stack for the model solver). In this example, for readability, the column titles and the entries are broken into multiple interleaved lines, with the user entries in a *bold font*.

Some fields have simple entries in the `Registry` file. The following is a `state` variable that is a Fortran type `real`. The name of the field inside the WRF model is...
u\_gc. It is a three dimension array (igj). This particular field is only for the ARW core (dyn\_em). It has a single time level, and is staggered in the X and Z directions. This field is input only to the real program (i1). On output, the netCDF name is UU, with the accompanying description and units provided.

<table>
<thead>
<tr>
<th>Table</th>
<th>Type</th>
<th>Sym</th>
<th>Dims</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>real</td>
<td>u_gc</td>
<td>igj</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use</th>
<th>NumTLev</th>
<th>Stagger</th>
<th>IO</th>
</tr>
</thead>
<tbody>
<tr>
<td>dyn_em</td>
<td>1</td>
<td>XZ</td>
<td>i1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DNAME</th>
<th>DESCRIP</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;UU&quot;</td>
<td>&quot;x-wind component&quot;</td>
<td>&quot;m s^-1&quot;</td>
</tr>
</tbody>
</table>

If a variable is not staggered, a “-“ (dash) is inserted instead of leaving a blank space. The same dash character is required to fill in a location when a field has no IO specification. The variable description and units columns are used for post-processing purposes only; this information is not directly utilized by the model.

When adding new variables to the Registry file, users are warned to make sure that variable names are unique. The <Sym> refers to the variable name inside the WRF model, and it is not case sensitive. The <DNAME> is quoted, and appears exactly as typed. Do not use imbedded spaces. While it is not required that the <Sym> and <DNAME> use the same character string, it is highly recommended. The <DESCRIP> and the <UNITS> are optional, however they are a good way to supply self-documentation to the Registry. Since the <DESCRIP> value is used in the automatic code generation, restrict the variable description to 40 characters or less.

From this example, we can add new requirements for a variable. Suppose that the variable to be added is not specific to any dynamical core. We would change the <Use> column entry of dyn\_em to misc (for miscellaneous). The misc entry is typical of fields used in physics packages. Only dynamics variables have more than a single time level, and this introductory material is not suitable for describing the impact of multiple time periods on the registry program. For the <Stagger> option, users may select any subset from \{X, Y, Z\} or \{-\}, where the dash character “-“ signifies “no staggering”. For example, in the ARW model, the x-direction wind component, u, is staggered in the X direction, and the y-direction wind component, v, is staggered in the Y direction.

The <IO> column handles file input and output, and it handles the nesting specification for the field. The file input and output uses three letters: i (input), r (restart), and h (history). If the field is to be in the input file to the model, the restart file from the model, and the history file from the model, the entry would be irh. To allow more flexibility, the input and history fields are associated with streams. The user may specify a digit after the i or the h token, stating that this variable is associated with a specified stream (1 through 9) instead of the default (0). A single variable may be associated with
multiple streams. Once any digit is used with the i or h tokens, the default 0 stream must be explicitly stated. For example, <IO> entry i and <IO> entry i0 are the same. However, <IO> entry h1 outputs the field to the first auxiliary stream, but does not output the field to the default history stream. The <IO> entry h01 outputs the field to both the default history stream and the first auxiliary stream. For streams larger than a single digit, such as stream number thirteen, the multi-digit numerical value is enclosed inside braces: i{13}. The maximum stream is currently 24 for both input and history.

Nesting support for the model is also handled by the <IO> column. The letters that are parsed for nesting are: u (up as in feedback up), d (down, as in downscale from coarse to fine grid), f (forcing, how the lateral boundaries are processed), and s (smoothing). As with other entries, the best coarse of action is to find a field nearly identical to the one that you are inserting into the Registry file, and copy that line. The user needs to make the determination whether or not it is reasonable to smooth the field in the area of the coarse grid, where the fine-grid feeds back to the coarse grid. Variables that are defined over land and water, non-masked, are usually smoothed. The lateral boundary forcing is primarily for dynamics variables, and is ignored in this overview presentation. For non-masked fields (such as wind, temperature, & pressure), the downward interpolation (controlled by d) and the feedback (controlled by u) use default routines. Variables that are land fields (such as soil temperature TSLB) or water fields (such as sea ice XICE) have special interpolators, as shown in the examples below (again, interleaved for readability):

```plaintext
#<Table>  <Type>  <Sym>  <Dims>
state    real   TSLB   ilj
state    real   XICE  ij

<Use>  <NumTLev>  <Stagger>
misc     1          Z
misc     1          -

<IO>
i02rhd=(interp_mask_land_field:lu_index)u=(copy_fcnm)
i0124rhd=(interp_mask_water_field:lu_index)u=(copy_fcnm)

<DNAME>  <DESCRIP>  <UNITS>
"TSLB"  "SOIL TEMPERATURE"  "K"
"SEAICE" "SEA ICE FLAG"  ""
```

Note that the d and u entries in the <IO> section are followed by an “=” then a parenthesis-enclosed subroutine, and a colon-separated list of additional variables to pass to the routine. It is recommended that users follow the existing pattern: du for non-masked variables, and the above syntax for the existing interpolators for masked variables.
Registry Rconfig:

The Registry file is the location where the run-time options to configure the model are defined. Every variable in the ARW namelist is described by an entry in the Registry file. The default value for each of the namelist variables is as assigned in the Registry. The standard form for the entry for two namelist variables is given (broken across lines and interleaved):

```
#<Table>  <Type>    <Sym>
rconfig  integer    run_days
rconfig  integer    start_year

<How set>          <Nentries>   <Default>
namelist,time_control      1           0
namelist,time_control  max_domains    1993
```

The keyword for this type of entry in the Registry file is rconfig (run-time configuration). As with the other model fields (such as state and il), the <Type> column assigns the Fortran kind of the variable: integer, real, or logical. The name of the variable in ARW is given in the <Sym> column, and is part of the derived data type structure, as are the state fields. There are a number of Fortran namelist records in the file namelist.input. Each namelist variable is a member of one of the specific namelist records. The previous example shows that run_days and start_year are both members of the time_control record. The <Nentries> column refers to the dimensionality of the namelist variable (number of entries). For most variables, the <Nentries> column has two eligible values, either 1 (signifying that the scalar entry is valid for all domains) or max_domains (signifying that the variable is an array, with a value specified for each domain). Finally, a default value is given. This permits a namelist entry to be removed from the namelist.input file if the default value is acceptable.

The registry program constructs two subroutines for each namelist variable: one to retrieve the value of the namelist variable, and the other to set the value. For an integer variable named my_nml_var, the following code snippet provides an example of the easy access to the namelist variables.

```
INTEGER :: my_nml_var, dom_id
CALL nl_get_my_nml_var ( dom_id , my_nml_var )
```

The subroutine takes two arguments. The first is the input integer domain identifier (for example, 1 for the most coarse grid, 2 for the second domain), and the second argument is the returned value of the namelist variable. The associated subroutine to set the namelist variable, with the same argument list, is nl_set_my_nml_var. For namelist variables that are scalars, the grid identifier should be set to 1.
The `rconfig` line may also be used to define variables that are convenient to pass around in the model, usually part of a derived configuration (such as the number of microphysics species associated with a physics package). In this case, the `<How set>` column entry is `derived`. This variable does not appear in the namelist, but is accessible with the same generated `nl_set` and `nl_get` subroutines.

**Registry Halo, Period, and Xpose:**

The distributed memory, inter-processor communications are fully described in the `Registry` file. An entry in the Registry constructs a code segment which is included (with `cpp`) in the source code. Following is an example of a `halo` communication (split across two lines and interleaved for readability).

```plaintext
#<Table>  <CommName>   <Core>
halo      HALO_EM_D2_3 dyn_em

<Stencil:varlist>
24:u_2,v_2,w_2,t_2,ph_2;24:moist,chem,scalar;4:mu_2,al
```

The keyword is `halo`. The communication is named in the `<CommName>` column, so that it can be referenced in the source code. The entry in the `<CommName>` column is case sensitive (the convention is to start the name with `HALO_EM`). The selected dynamical core is defined in the `<Core>` column. There is no ambiguity, as every communication in each `Registry` file will have the exact same `<Core>` column option. The last set of information is the `<Stencil:varlist>`. The portion in front of the “:” is the stencil size, and the comma-separated list afterwards defines the variables that are communicated with that stencil size. Different stencil sizes are available, and are “;”-separated in the same `<Stencil:varlist>` column. The stencil sizes 8, 24, 48 all refer to a square with an odd number of grid cells on a side, with the center grid cell removed (8 = 3x3-1, 24 = 5x5-1, 48 = 7x7-1). The special small stencil 4 is just a simple north, south, east, west communication pattern.

The convention in the WRF model is to provide a communication immediately after a variable has been updated. The communications are restricted to the mediation layer (an intermediate layer of the software that is placed between the framework level and the model level). The model level is where developers spend most of their time. The majority of users will insert communications into the `dyn_em/solve_em.F` subroutine. The `HALO_EM_D2_3` communication, defined in the `Registry` file in the example above, is activated by inserting a small section of code that includes an automatically generated code segment into the solve routine, via standard `cpp` directives.

```plaintext
#ifdef DM_PARALLEL
#    include "HALO_EM_D2_3.inc"
#endif
```
The parallel communications are only required when the ARW code is built for distributed-memory parallel processing, which accounts for the surrounding `#ifdef`.

The `period` communications are required when periodic lateral boundary conditions are selected. The Registry syntax is very similar for `period` and `halo` communications, but the stencil size refers to how many grid cells to communicate, in a direction that is normal to the periodic boundary.

```plaintext
/Table> <CommName>       <Core>       <Stencil:varlist>
period   PERIOD_EM_COUPLE_A  dyn_em   2:mub,mu_1,mu_2
```

The `xpose` (a data transpose) entry is used when decomposed data is to be re-decomposed. This is required when doing FFTs in the x-direction for polar filtering, for example. No stencil size is necessary.

```plaintext
/Table> <CommName>       <Core>       <Varlist>
xpose   XPOSE_POLAR_FILTER_T  dyn_em   t_2,t_xxx,dum_yyy
```

It is anticipated that many users will add to the parallel communications portion of the Registry file (`halo` and `period`). It is unlikely that users will add `xpose` fields.

Registry Package:

The `package` option in the Registry file associates fields with particular physics packages. Presently, it is mandatory that all 4-D arrays be assigned. Any 4-D array that is not associated with the selected physics option at run-time is neither allocated, used for IO, nor communicated. All other 2-D and 3-D arrays are eligible for use with a `package` assignment, but that is not required.

The purpose of the `package` option is to allow users to reduce the memory used by the model, since only “necessary” fields are processed. An example for a microphysics scheme is given below.

```plaintext
/Table> <PackageName>     <NMLAssociated>    <Variables>
package   kesslerscheme   mp_physics==1  - moist:qv,qc,qr
```

The entry keyword is `package`, and is associated with the single physics option listed under `<NMLAssociated>`. The package is referenced in the code in Fortran `IF` and `CASE` statements by the name given in the `<PackageName>` column, instead of the more confusing and typical `IF ( mp_physics == 1 )` approach. The `<Variables>` column must start with a dash character and then a blank “−” (for historical reasons of backward compatibility). The syntax of the `<Variables>` column then is a 4-D array name, followed by a colon, and then a comma-separated list of the 3-D arrays constituting that 4-D amalgamation. In the example above, the 4-D array is...
moist, and the selected 3-D arrays are $qv$, $qc$, and $qr$. If more than one 4-D array is required, a “;” separates those sections from each other in the <Variables> column.

In addition to handling 4-D arrays and their underlying component, 3-D arrays, the package entry is able to associate generic state variables, as shown in the example following. If the namelist variable `use_wps_input` is set to 1, then the variables $u_{gc}$ and $v_{gc}$ are available to be processed.

```
#<Table>  <PackageName> <NMLAssociated>     <Variables>
package     realonly    use_wps_input==1 - state:u_gc,v_gc
```

I/O Applications Program Interface (I/O API)

The software that implements WRF I/O, like the software that implements the model in general, is organized hierarchically, as a “software stack” (http://www2.mmm.ucar.edu/wrf/WG2/Tigers/IOAPI/IOStack.html). From top (closest to the model code itself) to bottom (closest to the external package implementing the I/O), the I/O stack looks like this:

- Domain I/O (operations on an entire domain)
- Field I/O (operations on individual fields)
- Package-neutral I/O API
- Package-dependent I/O API (external package)

The lower-levels of the stack, associated with the interface between the model and the external packages, are described in the I/O and Model Coupling API specification document on http://www2.mmm.ucar.edu/wrf/WG2/Tigers/IOAPI/index.html.

Timekeeping

Starting times, stopping times, and time intervals in WRF are stored and manipulated as Earth System Modeling Framework (ESMF, http://www.cisl.ucar.edu/research/2005/esmf.jsp) time manager objects. This allows exact representation of time instants and intervals as integer numbers of years, months, hours, days, minutes, seconds, and fractions of a second (numerator and denominator are specified separately as integers). All time computations involving these objects are performed exactly by using integer arithmetic, with the result that there is no accumulated time step drift or rounding, even for fractions of a second.
The WRF implementation of the ESMF Time Manager is distributed with WRF in the `external/esmf_time_f90` directory. This implementation is entirely Fortran90 (as opposed to the ESMF implementation in C++) and it is conformant to the version of the ESMF Time Manager API that was available in 2009.

WRF source modules and subroutines that use the ESMF routines do so by use-association of the top-level ESMF Time Manager module, esmf_mod:

```
USE esmf_mod
```

The code is linked to the library file `libesmf_time.a` in the `external/esmf_time_f90` directory.

ESMF timekeeping is set up on a domain-by-domain basis in the routine `setup_timekeeping (share/set_timekeeping.F)`. Each domain keeps track of its own clocks and alarms. Since the time arithmetic is exact there is no problem with clocks on separate domains getting out of synchronization.

**Software Documentation**

Detailed and comprehensive documentation aimed at WRF software is available at [http://www2.mmm.ucar.edu/wrf/WG2/software_2.0](http://www2.mmm.ucar.edu/wrf/WG2/software_2.0).

**Performance**

Benchmark information is available at [http://www2.mmm.ucar.edu/wrf/bench](http://www2.mmm.ucar.edu/wrf/bench).
Chapter 9: Post-Processing Utilities

Table of Contents

- Introduction
- NCL
- RIP4
- ARWpost
- UPP
- VAPOR

Introduction

There are a number of visualization tools available to display WRF-ARW (http://http://www2.mmm.ucar.edu/wrf/users) model data. Model data in netCDF format can essentially be displayed using any tool capable of displaying this data format.

Currently the following post-processing utilities are supported: NCL, RIP4, ARWpost (converter to GrADS), UPP, and VAPOR.

NCL, RIP4, ARWpost and VAPOR can currently only read data in netCDF format, while UPP can read data in netCDF and binary format.

Required software

The only library that is always required is the netCDF package from Unidata (http://www.unidata.ucar.edu: login > Downloads > NetCDF - registration login required).

netCDF stands for Network Common Data Form. This format is platform independent, i.e., data files can be read on both big-endian and little-endian computers, regardless of where the file was created. To use the netCDF libraries, ensure that the paths to these libraries are set correct in your login scripts as well as all Makefiles.

Additional libraries required by each of the supported post-processing packages:

- NCL (http://www.ncl.ucar.edu)
- GrADS (http://grads.iges.org/home.html)
- GEMPAK (http://www.unidata.ucar.edu/software/gempak/)
- VAPOR (http://www.vapor.ucar.edu)
NCL

With the use of NCL Libraries (http://www.ncl.ucar.edu), WRF-ARW data can easily be displayed.

The information on these pages has been put together to help users generate NCL scripts to display their WRF-ARW model data.

Some example scripts are available online (http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_examples.htm), but in order to fully utilize the functionality of the NCL Libraries, users should adapt these for their own needs, or write their own scripts.

NCL can process WRF-ARW static, input and output files, as well as WRFDA output data. Both single and double precision data can be processed.

WRF and NCL

In July 2007, the WRF-NCL processing scripts have been incorporated into the NCL Libraries, thus only the NCL Libraries are now needed.

Major WRF-ARW-related upgrades have been added to the NCL libraries in version 6.1.0; therefore, in order to use many of the functions, NCL version 6.1.0 or higher is required.

Special functions are provided to simplify the plotting of WRF-ARW data. These functions are located in: "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl". Users are encouraged to view and edit this file for their own needs. If users wish to edit this file, but do not have write permission, they should simply copy the file to a local directory, edit and load the new version, when running NCL scripts.

Special NCL built-in functions have been added to the NCL libraries to help users calculate basic diagnostics for WRF-ARW data.

All the FORTRAN subroutines used for diagnostics and interpolation (previously located in wrf_user_fortran_util_0.f) has been re-coded into NCL in-line functions. This means users no longer need to compile these routines.
What is NCL

The NCAR Command Language (NCL) is a free, interpreted language designed specifically for scientific data processing and visualization. NCL has robust file input and output. It can read in netCDF, HDF4, HDF4-EOS, GRIB, binary and ASCII data. The graphics are world-class and highly customizable.

It runs on many different operating systems including Solaris, AIX, IRIX, Linux, MacOSX, Dec Alpha, and Cygwin/X running on Windows. The NCL binaries are freely available at: http://www.ncl.ucar.edu/Download/

To read more about NCL, visit: http://www.ncl.ucar.edu/overview.shtml

Necessary software

NCL libraries, version 6.1.0 or higher.

Environment Variable

Set the environment variable NCARG_ROOT to the location where you installed the NCL libraries. Typically (for cshrc shell):

```
setenv NCARG_ROOT /usr/local/ncl
```

.hluresfile

Create a file called .hluresfile in your $HOME directory. This file controls the color, background, fonts, and basic size of your plot. For more information regarding this file, see: http://www.ncl.ucar.edu/Document/Graphics/hlures.shtml.

NOTE: This file must reside in your $HOME directory and not where you plan on running NCL.

Below is the .hluresfile used in the example scripts posted on the web (scripts are available at: http://www2.mmm.ucar.edu/wrf/users/graphics/NCL/NCL.htm). If a different color table is used, the plots will appear different. Copy the following to your ~/.hluresfile. (A copy of this file is available at: http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_basics.htm)

```
*wkColorMap : BlAqGrYeOrReVi200
*wkBackgroundColor : white
*wkForegroundColor : black
*FuncCode : ~
*TextFuncCode : ~
```
NOTE:
If your image has a black background with white lettering, your .hluresfile has not been created correctly, or it is in the wrong location. wkColorMap, as set in your .hluresfile can be overwritten in any NCL script with the use of the function “gsn_define_colormap”, so you do not need to change your .hluresfile if you just want to change the color map for a single plot.

Create NCL scripts

The basic outline of any NCL script will look as follows:

```ncl
load external functions and procedures
begin
  ; Open input file(s)
  ; Open graphical output
  ; Read variables
  ; Set up plot resources & Create plots
  ; Output graphics
end
```

For example, let’s create a script to plot Surface Temperature, Sea Level Pressure and Wind as shown in the picture below.
; load functions and procedures
load "${NCARG_ROOT}/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "${NCARG_ROOT}/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"

begin

; WRF ARW input file (NOTE, your wrfout file does not need
; the .nc, but NCL needs it so make sure to add it in the
; line below)
a = addfile("../wrfout_d01_2000-01-24_12:00:00.nc","r")

; Output on screen. Output will be called "plt_Surface1"
type = "x11"
wks = gsn_open_wks(type,"plt_Surface1")

; Set basic resources
res = True
res@MainTitle = "REAL-TIME WRF" ; Give plot a main title
res@Footer = False ; Set Footers off
pltres = True ; Plotting resources
mpres = True ; Map resources

; Get times in the file
it = 0 ; only interested in first time
res@TimeLabel = times(it) ; keep some time information

; Get variables
slp = wrf_user_getvar(a,"slp",it) ; Get slp
wrf_smooth_2d( slp, 3 ) ; Smooth slp

t2 = wrf_user_getvar(a,"T2",it) ; Get T2 (deg K)
tc2 = t2-273.16
tf2 = 1.8*tc2+32.
tf2@description = "Surface Temperature"
tf2@units = "F"

u10 = wrf_user_getvar(a,"U10",it) ; Get U10
v10 = wrf_user_getvar(a,"V10",it) ; Get V10
u10 = u10*1.94386
v10 = v10*1.94386
u10@units = "kts"
v10@units = "kts"
; Plotting options for T
opts = res                              ; Add basic resources
opts@cnFillOn = True                    ; Shaded plot
opts@ContourParameters = (/ -20., 90., 5./) ; Contour intervals
opts@gsnSpreadColorEnd = -3
contour_tc = wrf_contour(a,wks,tf2,opts) ; Create plot
delete(opts)

; Plotting options for SLP
opts = res                              ; Add basic resources
opts@cnLineColor = "Blue"                ; Set line color
opts@cnHighLabelsOn = True              ; Set labels
opts@cnLowLabelsOn = True
opts@ContourParameters = (/ 900.,1100.,4./) ; Contour intervals
contour_psl = wrf_contour(a,wks,slp,opts) ; Create plot
delete(opts)

; Plotting options for Wind Vectors
opts = res                              ; Add basic resources
opts@FieldTitle = "Winds"                ; Overwrite the field title
opts@NumVectors = 47                    ; Density of wind barbs
vector = wrf_vector(a,wks,u10,v10,opts) ; Create plot
delete(opts)

; MAKE PLOTS
plot = wrf_map_overlays(a,wks,\ 
   (/contour_tc,contour_psl,vector/),pltres,mpres)

;---------------------------------------------------------------
end

Extra sample scripts are available at:

Run NCL scripts

1. Ensure NCL is successfully installed on your computer.

2. Ensure that the environment variable NCARG_ROOT is set to the location where
   NCL is installed on your computer. Typically (for cshrc shell), the command will
look as follows:

```
setenv NCARG_ROOT /usr/local/ncl
```

3. Create an NCL plotting script.

4. Run the NCL script you created:

```
ncl NCL_script
```

The output type created with this command is controlled by the line:

```
wks = gsn_open_wk (type,"Output")  ; inside the NCL script
```

where `type` can be `x11`, `pdf`, `ncgm`, `ps`, or `eps`

For high quality images, create pdf, ps, or eps images directly via the ncl scripts (`type = pdf / ps / eps`)

See the **Tools** section in Chapter 10 of this User’s Guide for more information concerning other types of graphical formats and conversions between graphical formats.

**Functions / Procedures under "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/"**

(WRFUserARW.ncl)

`wrf_user_getvar (nc_file, fld, it)`

**Usage:** `ter = wrf_user_getvar (a, “HGT”, 0)`

Get fields from a netCDF file for:

- Any given time by setting `it` to the time required.
- For all times in the input file(s), by setting `it = -1`
- A list of times from the input file(s), by setting `it` to `/start_time,end_time,interval/` (e.g. `/0,10,2/`).
- A list of times from the input file(s), by setting `it` to the list required (e.g. `/1,3,7,10/`).

Any field available in the netCDF file can be extracted.

`fld` is case sensitive. The policy adapted during development was to set all diagnostic variables, calculated by NCL, to lower-case to distinguish them from fields directly available from the netCDF files.
List of available diagnostics:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avo</td>
<td>Absolute Vorticity [10-5 s⁻¹]</td>
</tr>
<tr>
<td>pvo</td>
<td>Potential Vorticity [PVU]</td>
</tr>
<tr>
<td>eth</td>
<td>Equivalent Potential Temperature [K]</td>
</tr>
<tr>
<td>cape_2d</td>
<td>Returns 2D fields mcape/mcin/lcl/lfc</td>
</tr>
<tr>
<td>cape_3d</td>
<td>Returns 3D fields cape/cin</td>
</tr>
<tr>
<td>dbz</td>
<td>Reflectivity [dBZ]</td>
</tr>
<tr>
<td>mdbz</td>
<td>Maximum Reflectivity [dBZ]</td>
</tr>
<tr>
<td>geopt/geopotential</td>
<td>Full Model Geopotential [m² s⁻²]</td>
</tr>
<tr>
<td>helicity</td>
<td>Storm Relative Helicity [m²/s⁻²]</td>
</tr>
<tr>
<td>updraft_helicity</td>
<td>Updraft Helicity [m²/s⁻²]</td>
</tr>
<tr>
<td>lat</td>
<td>Latitude (will return either XLAT or XLAT_M, depending on which is available)</td>
</tr>
<tr>
<td>lon</td>
<td>Longitude (will return either XLONG or XLONG_M, depending on which is available)</td>
</tr>
<tr>
<td>omg</td>
<td>Omega</td>
</tr>
<tr>
<td>p/pres</td>
<td>Full Model Pressure [Pa]</td>
</tr>
<tr>
<td>pressure</td>
<td>Full Model Pressure [hPa]</td>
</tr>
<tr>
<td>pw</td>
<td>Precipitable Water</td>
</tr>
<tr>
<td>rh2</td>
<td>2m Relative Humidity [%]</td>
</tr>
<tr>
<td>rh</td>
<td>Relative Humidity [%]</td>
</tr>
<tr>
<td>slp</td>
<td>Sea Level Pressure [hPa]</td>
</tr>
<tr>
<td>ter</td>
<td>Model Terrain Height [m] (will return either HGT or HGT_M, depending on which is available)</td>
</tr>
<tr>
<td>td2</td>
<td>2m Dew Point Temperature [°C]</td>
</tr>
<tr>
<td>td</td>
<td>Dew Point Temperature [°C]</td>
</tr>
<tr>
<td>tc</td>
<td>Temperature [°C]</td>
</tr>
<tr>
<td>tk</td>
<td>Temperature [K]</td>
</tr>
<tr>
<td>th/theta</td>
<td>Potential Temperature [K]</td>
</tr>
<tr>
<td>tv</td>
<td>Virtual Temperature</td>
</tr>
<tr>
<td>twb</td>
<td>Wetbulb Temperature</td>
</tr>
<tr>
<td>times</td>
<td>Times in file (note this return strings - recommended)</td>
</tr>
<tr>
<td>Times</td>
<td>Times in file (note this return characters)</td>
</tr>
<tr>
<td>ua</td>
<td>U component of wind on mass points</td>
</tr>
<tr>
<td>va</td>
<td>V component of wind on mass points</td>
</tr>
<tr>
<td>wa</td>
<td>W component of wind on mass points</td>
</tr>
<tr>
<td>uvmet10</td>
<td>10m U and V components of wind rotated to earth coordinates</td>
</tr>
<tr>
<td>uvmet</td>
<td>U and V components of wind rotated to earth coordinates</td>
</tr>
<tr>
<td>z/height</td>
<td>Full Model Height [m]</td>
</tr>
</tbody>
</table>
### wrf_user_list_times (nc_file)

**Usage:**
```
times = wrf_user_list_times (a)
```

Obtain a list of times available in the input file. The function returns a 1D array containing the times (type: character) in the input file.

This is an outdated function – best to use `wrf_user_getvar(nc_file,"times",it)`

### wrf_contour (nc_file, wks, data, res)

**Usage:**
```
contour = wrf_contour (a, wks, ter, opts)
```

Returns a graphic (contour), of the data to be contoured. This graphic is only created, but not plotted to a wks. This enables a user to generate many such graphics and overlay them, before plotting the resulting picture to the wks.

The returned graphic (contour) does not contain map information, and can therefore be used for both real and idealized data cases.

This function can plot both line contours and shaded contours. **Default is line contours.**

Many resources are set for a user, and most can be overwritten. Below is a list of resources you may want to consider changing before generating your own graphics:

#### Resources unique to ARW WRF Model data

- **opts@MainTitle**: Controls main title on the plot.
- **opts@MainTitlePos**: Main title position – Left/Right/Center. Default is Left.
- **opts@NoHeaderFooter**: Switch off all Headers and Footers.
- **opts@Footer**: Add some model information to the plot as a footer. Default is True.
- **opts@InitTime**: Plot initial time on graphic. Default is True. If True, the initial time will be extracted from the input file.
- **opts@ValidTime**: Plot valid time on graphic. Default is True. A user must set `opts@TimeLabel` to the correct time.
- **opts@TimeLabel**: Time to plot as valid time.
- **opts@TimePos**: Time position – Left/Right. Default is “Right”.
- **opts@ContourParameters**: A single value is treated as an interval. Three values represent: Start, End, and Interval.
- **opts@FieldTitle**: Overwrite the field title - if not set the field description is used for the title.
- **opts@UnitLabel**: Overwrite the field units - seldom needed as the units associated with the field will be used.
- **opts@PlotLevelID**: Use to add level information to the field title.

#### General NCL resources (most standard NCL options for cn and lb can be set by the user to overwrite the default values)

- **opts@cnFillOn**: Set to True for shaded plots. Default is False.
- **opts@cnLineColor**: Color of line plot.
opts@lbTitleOn : Set to False to switch the title on the label bar off. Default is True.

opts@cnLevelSelectionMode ; opts@cnLevels ; opts@cnFillColors ;

optr@cnConstFLabelOn : Can be used to set contour levels and colors manually.

wrf_vector (nc_file, wks, data_u, data_v, res)
Usage: vector = wrf_vector (a, wks, ua, va, opts)

Returns a graphic (vector) of the data. This graphic is only created, but not plotted to a wks. This enables a user to generate many graphics, and overlay them, before plotting the resulting picture to the wks.

The returned graphic (vector) does not contain map information, and can therefore be used for both real and idealized data cases.

Many resources are set for a user, and most can be overwritten. Below is a list of resources you may want to consider changing before generating your own graphics:

Resources unique to ARW WRF Model data

opts@MainTitle : Controls main title on the plot.
opts@MainTitlePos : Main title position – Left/Right/Center. Default is Left.
opts@NoHeaderFooter : Switch off all Headers and Footers.
opts@Footer : Add some model information to the plot as a footer. Default is True.
opts@InitTime : Plot initial time on graphic. Default is True. If True, the initial time will be extracted from the input file.
opts@ValidTime : Plot valid time on graphic. Default is True. A user must set
opts@TimeLabel to the correct time.
opts@TimeLabel : Time to plot as valid time.
opts@TimePos : Time position – Left/Right. Default is “Right”.
opts@ContourParameters : A single value is treated as an interval. Three values represent: Start, End, and Interval.
opts@FieldTitle : Overwrite the field title - if not set the field description is used for the title.
opts@UnitLabel : Overwrite the field units - seldom needed as the units associated with the field will be used.
opts@PlotLevelID : Use to add level information to the field title.
opts@NumVectors : Density of wind vectors.

General NCL resources (most standard NCL options for vc can be set by the user to overwrite the default values)

opts@vcGlyphStyle : Wind style. “WindBarb” is default.
wrf_map_overlays (nc_file, wks, (/graphics/), pltres, mpres)
Usage: plot = wrf_map_overlays (a, wks, (/contour,vector/), pltres, mpres)

Overlay contour and vector plots generated with wrf_contour and wrf_vector. Can overlay any number of graphics. Overlays will be done in the order given, so always list shaded plots before line or vector plots, to ensure the lines and vectors are visible and not hidden behind the shaded plot.

A map background will automatically be added to the plot. Map details are controlled with the mpres resource. Common map resources you may want to set are:

mpres@mpGeophysicalLineColor; mpres@mpNationalLineColor;
mpres@mpUSStateLineColor; mpres@mpGridLineColor;
mpres@mpLimbLineColor; mpres@mpPerimLineColor

If you want to zoom into the plot, set mpres@ZoomIn to True, and mpres@Xstart, mpres@Xend, mpres@Ystart, and mpres@Yend to the corner x/y positions of the zoomed plot.

pltres@NoTitles : Set to True to remove all field titles on a plot.
pltres@CommonTitle : Overwrite field titles with a common title for the overlaid plots.
Must set pltres@PlotTitle to desired new plot title.

If you want to generate images for a panel plot, set pltres@PanelPot to True.

If you want to add text/lines to the plot before advancing the frame, set pltres@FramePlot to False. Add your text/lines directly after the call to the wrf_map_overlays function. Once you are done adding text/lines, advance the frame with the command “frame (wks)”.

wrf_overlays (nc_file, wks, (/graphics/), pltres)
Usage: plot = wrf_overlays (a, wks, (/contour,vector/), pltres)

Overlay contour and vector plots generated with wrf_contour and wrf_vector. Can overlay any number of graphics. Overlays will be done in the order given, so always list shaded plots before line or vector plots, to ensure the lines and vectors are visible and not hidden behind the shaded plot.

Typically used for idealized data or cross-sections, which does not have map background information.

pltres@NoTitles : Set to True to remove all field titles on a plot.
pltres@CommonTitle : Overwrite field titles with a common title for the overlaid plots.
Must set pltres@PlotTitle to desired new plot title.

If you want to generate images for a panel plot, set pltres@PanelPot to True.
If you want to add text/lines to the plot before advancing the frame, set \texttt{pltres@FramePlot} to False. Add your text/lines directly after the call to the \texttt{wrf_overlays} function. Once you are done adding text/lines, advance the frame with the command \texttt{frame (wks)}.

\textbf{wrf\_map (nc\_file, wks, res)}

Usage: \texttt{map = wrf\_map (a, wks, opts)}

Create a map background.
As maps are added to plots automatically via the \texttt{wrf\_map\_overlays} function, this function is seldom needed as a stand-alone.

\textbf{wrf\_user\_intrp3d (var3d, H, plot\_type, loc\_param, angle, res)}

This function is used for both horizontal and vertical interpolation.

\textbf{var3d}: The variable to interpolate. This can be an array of up to 5 dimensions. The 3 right-most dimensions must be \texttt{bottom\_top x south\_north x west\_east}.

\textbf{H}: The field to interpolate to. Either pressure (hPa or Pa), or z (m). Dimensionality must match \texttt{var3d}.

\textbf{plot\_type}: “h” for horizontally- and “v” for vertically-interpolated plots.

\textbf{loc\_param}: Can be a scalar, or an array, holding either 2 or 4 values.

\textit{For plot\_type = “h”}:

This is a scalar representing the level to interpolate to.
Must match the field to interpolate to (H).
When interpolating to pressure, this can be in hPa or Pa (\textit{e.g.} 500., \textit{to interpolate to 500 hPa}). When interpolating to height this must in in m (\textit{e.g.} 2000., \textit{to interpolate to 2 km}).

\textit{For plot\_type = “v”}:

This can be a pivot point though which a line is drawn – in this case a single x/y point (2 values) is required. Or this can be a set of x/y points (4 values), indicating start x/y and end x/y locations for the cross-section.

\textbf{angle}:

Set to 0., for \texttt{plot\_type = “h”}, or for \texttt{plot\_type = “v”} when start and end locations of cross-section are supplied in \texttt{loc\_param}.
If a single pivot point was supplied in \texttt{loc\_param}, angle is the angle of the line that will pass through the pivot point. Where: 0. is SN, and 90. is WE.

\textbf{res}:

Set to False for \texttt{plot\_type = “h”}, or for \texttt{plot\_type = “v”} when a single pivot point is supplied. Set to True if start and end locations are supplied.
wrf_user_intrp2d (var2d, loc_param, angle, res)

This function interpolates a 2D field along a given line.

var2d: The 2D field to interpolate. This can be an array of up to 3 dimensions. The 2 right-most dimensions must be south_north x west_east.

loc_param:
An array holding either 2 or 4 values.
This can be a pivot point through which a line is drawn - in this case a single x/y point (2 values) is required. Or this can be a set of x/y points (4 values), indicating start x/y and end x/y locations for the cross-section.

angle:
Set to 0 when start and end locations of the line are supplied in loc_param.
If a single pivot point is supplied in loc_param, angle is the angle of the line that will pass through the pivot point. Where: 0. is SN, and 90. is WE.

res:
Set to False when a single pivot point is supplied. Set to True if start and end locations are supplied.

wrf_user_ll_to_ij (nc_file, lons, lats, res)
Usage: loc = wrf_user_latlon_to_ij (a, 100., 40., res)
Usage: loc = wrf_user_latlon_to_ij (a, (/100., 120./), (/40., 50./), res)

Converts a lon/lat location to the nearest x/y location. This function makes use of map information to find the closest point; therefore this returned value may potentially be outside the model domain.

lons/lats can be scalars or arrays.

Optional resources:
res@returnInt - If set to False, the return values will be real (default is True with integer return values)
res@useTime - Default is 0. Set if you want the reference longitude/latitudes to come from a specific time - one will only use this for moving nest output, which has been stored in a single file.

loc(0,:) is the x (WE) locations, and loc(1,:) the y (SN) locations.

wrf_user_ij_to_ll (nc_file, i, j, res)
Usage: loc = wrf_user_latlon_to_ij (a, 10, 40, res)
Usage: loc = wrf_user_latlon_to_ij (a, (/10, 12/), (/40, 50/), res)
Convert an i/j location to a lon/lat location. This function makes use of map information to find the closest point, so this returned value may potentially be outside the model domain.

i/j can be scalars or arrays.

**Optional resources:**

res@useTime - Default is 0. Set if you want the reference longitude/latitudes to come from a specific time - one will only use this for moving nest output, which has been stored in a single file.

loc(0,:) is the lons locations, and loc(1,:) the lats locations.

**wrf_user_unstagger (varin, unstagDim)**

This function unstaggers an array, and returns an array on ARW WRF mass points.

varin: Array to be unstaggered.

unstagDim: Dimension to unstagger. Must be either "X", "Y", or "Z". This is case sensitive. If you do not use one of these strings, the returning array will be unchanged.

**wrf_wps_dom (wks, mpres, lnres, txres)**

A function has been built into NCL to preview where a potential domain will be placed (similar to plotgrids.exe from WPS).

The lnres and txres resources are standard NCL Line and Text resources. These are used to add nests to the preview.

The mpres are used for standard map background resources like:

mpres@mpFillOn ; mpres@mpFillColors ; mpres@mpGeophysicalLineColor ;
mpres@mpNationalLineColor ; mpres@mpUSStateLineColor ;
mpres@mpGridLineColor ; mpres@mpLimbLineColor ;
mpres@mpPerimLineColor

Its main function, however, is to set map resources to preview a domain. These resources are similar to the resources set in WPS. Below is an example of how to display 3 nested domains on a Lambert projection. *(The output is shown below).*

mpres@max_dom           = 3
mpres@parent_id         = (/ 1, 1, 2 /)
mpres@parent_grid_ratio = (/ 1, 3, 3 /)
mpres@i_parent_start    = (/ 1, 31, 15 /)
mpres@j_parent_start    = (/ 1, 17, 20 /)
mpres@e_we = (/ 74, 112, 133/)
mpres@e_sn = (/ 61, 97, 133 /)
mpres@dx = 30000.
mpres@dy = 30000.
mpres@map_proj = "lambert"
mpres@ref_lat = 34.83
mpres@ref_lon = -81.03
mpres@truelat1 = 30.0
mpres@truelat2 = 60.0
mpres@stand_lon = -98.0

NCL built-in Functions

A number of NCL built-in functions have been created to help users calculate simple diagnostics. Full descriptions of these functions are available on the NCL web site (http://www.ncl.ucar.edu/Document/Functions/wrf.shtml).

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wrf_avo</td>
<td>Calculates absolute vorticity.</td>
</tr>
<tr>
<td>wrf_cape_2d</td>
<td>Computes convective available potential energy (CAPE), convective inhibition (CIN), lifted condensation level (LCL), and level of free convection (LFC).</td>
</tr>
<tr>
<td>wrf_cape_3d</td>
<td>Computes convective available potential energy (CAPE) and convective inhibition (CIN).</td>
</tr>
<tr>
<td>wrf_dbz</td>
<td>Calculates the equivalent reflectivity factor.</td>
</tr>
<tr>
<td>wrf_eth</td>
<td>Calculates equivalent potential temperature</td>
</tr>
<tr>
<td>wrf_helicity</td>
<td>Calculates storm relative helicity</td>
</tr>
<tr>
<td>wrf_ij_to_ll</td>
<td>Finds the longitude, latitude locations to the specified model grid indices (i,j).</td>
</tr>
</tbody>
</table>
wrf_ll_to_ij | Finds the model grid indices (i,j) to the specified location(s) in longitude and latitude.
wrf_omega | Calculates omega
wrf_pvo | Calculates potential vorticity.
wrf_rh | Calculates relative humidity.
wrf_slp | Calculates sea level pressure.
wrf_smooth_2d | Smooth a given field.
wrf_td | Calculates dewpoint temperature in [C].
wrfifikasi | Calculates temperature in [K].
wrf_updraft_helicity | Calculates updraft helicity
wrf_uvmet | Rotates u,v components of the wind to earth coordinates.
wrf_virtual_temp | Calculates virtual temperature
wrf_wetbulb | Calculates wetbulb temperature

Adding diagnostics using FORTRAN code

It is possible to link your favorite FORTRAN diagnostics routines to NCL. It is easier to use FORTRAN 77 code, but NCL also recognizes basic FORTRAN 90 code.

Let’s use a routine that calculates temperature (K) from theta and pressure.

**FORTRAN 90 routine called myTK.f90**

```fortran
subroutine compute_tk (tk, pressure, theta, nx, ny, nz)
implicit none

!! Variables
  integer :: nx, ny, nz
  real, dimension (nx,ny,nz) :: tk, pressure, theta

!! Local Variables
  integer :: i, j, k
  real, dimension (nx,ny,nz) :: pi

  pi(:,:,)= (pressure(:,:,)/1000.)**(287./1004.)
  tk(:,:,)= pi(:,:,)*theta(:,:,)

return
end subroutine compute_tk
```

For simple routines like this, it is easiest to re-write the routine into a FORTRAN 77 routine.
**FORTRAN 77 routine called myTK.f**

```fortran
subroutine compute_tk (tk, pressure, theta, nx, ny, nz)
implicit none

C Variables
integer nx, ny, nz
real tk(nx,ny,nz) , pressure(nx,ny,nz), theta(nx,ny,nz)

C Local Variables
integer i, j, k
real pi

DO k=1,nz
  DO j=1,ny
    DO i=1,nx
      pi=(pressure(i,j,k) / 1000.)**(287./1004.)
      tk(i,j,k) = pi*theta(i,j,k)
    ENDDO
  ENDDO
ENDDO
return
end
```

Add the markers **NCLFORTSTART** and **NCLEND** to the subroutine as indicated below. Note, that local variables are outside these block markers.

**FORTRAN 77 routine called myTK.f, with NCL markers added**

```fortran
C NCLFORTSTART
subroutine compute_tk (tk, pressure, theta, nx, ny, nz)
implicit none

C Variables
integer nx, ny, nz
real tk(nx,ny,nz) , pressure(nx,ny,nz), theta(nx,ny,nz)

C NCLEND

C Local Variables
integer i, j, k
real pi

DO k=1,nz
  DO j=1,ny
    DO i=1,nx
      pi=(pressure(i,j,k) / 1000.)**(287./1004.)
      tk(i,j,k) = pi*theta(i,j,k)
    ENDDO
  ENDDO
ENDDO
return
end
```
Now compile this code using the NCL script WRAPIT.

```
WRAPIT myTK.f
```

**NOTE:** If WRAPIT cannot be found, make sure the environment variable `NCARG_ROOT` has been set correctly.

If the subroutine compiles successfully, a new library will be created, called `myTK.so`. This library can be linked to an NCL script to calculate TK. See how this is done in the example below:

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
external myTK "/myTK.so"

begin

    t = wrf_user_getvar (a,"T",5)
    theta = t + 300
    p = wrf_user_getvar (a,"pressure",5)

    dim = dimsizes(t)
    tk = new ( (/ dim(0), dim(1), dim(2) /), float)

    myTK :: compute_tk (tk, p, theta, dim(2), dim(1), dim(0))

end
```

Want to use the FORTRAN 90 program? It is possible to do so by providing an interface block for your FORTRAN 90 program. Your FORTRAN 90 program may also not contain any of the following features:
- pointers or structures as arguments,
- missing/optional arguments,
- keyword arguments, or
- if the procedure is recursive.

**Interface block for FORTRAN 90 code, called myTK90.stub**

```c
C NCLFORTSTART
subroutine compute_tk (tk, pressure, theta, nx, ny, nz)

integer nx, ny, nz
real tk(nx,ny,nz), pressure(nx,ny,nz), theta(nx,ny,nz)

C NCLEND
```

Now compile this code using the NCL script WRAPIT.

```
WRAPIT myTK90.stub myTK.f90
```
NOTE: You may need to copy the WRAPIT script to a location and edit it to point to a FORTRAN 90 compiler.

If the subroutine compiles successfully, a new library will be created, called **myTK90.so** *(note the change in name from the FORTRAN 77 library).* This library can similarly be linked to an NCL script to calculate TK. See how this is done in the example below:

```ncl
load "SNCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "SNCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl"
external myTK90 "/myTK90.so"

begin
    t = wrf_user_getvar(a,"T",5)
    theta = t + 300
    p = wrf_user_getvar(a,"pressure",5)

    dim = dimsizes(t)
    tk = new( (/ dim(0), dim(1), dim(2) /), float)

    myTK90 :: compute_tk(tk, p, theta, dim(2), dim(1), dim(0))
end
```
RIP4

RIP (which stands for Read/Interpolate/Plot) is a Fortran program that invokes NCAR Graphics routines for the purpose of visualizing output from gridded meteorological data sets, primarily from mesoscale numerical models. It was originally designed for sigma-coordinate-level output from the PSU/NCAR Mesoscale Model (MM4/MM5), but was generalized in April 2003 to handle data sets with any vertical coordinate, and in particular, output from the Weather Research and Forecast (WRF) modeling system. It can also be used to visualize model input or analyses on model grids. It has been under continuous development since 1991, primarily by Mark Stoelinga at both NCAR and the University of Washington.

The RIP users' guide (http://www2.mmm.ucar.edu/wrf/users/docs/ripug.htm) is essential reading.

Code history

Version 4.0: reads WRF-ARW real output files
Version 4.1: reads idealized WRF-ARW datasets
Version 4.2: reads all the files produced by WPS
Version 4.3: reads files produced by WRF-NMM model
Version 4.4: add ability to output different graphical types
Version 4.5: add configure/compiler capabilities
Version 4.6: current version – only bug fix changes between 4.5 and 4.6

(This document will only concentrate on running RIP4 for WRF-ARW. For details on running RIP4 for WRF-NMM, see the WRF-NMM User's Guide: http://www.dtcenter.org/wrf-nmm/users/docs/overview.php)

Necessary software

RIP4 only requires low-level NCAR Graphics libraries. These libraries have been merged with the NCL libraries since the release of NCL version 5 (http://www.ncl.ucar.edu/), so if you don’t already have NCAR Graphics installed on your computer, install NCL version 5.

Obtain the code from the WRF-ARW user’s web site:

http://www2.mmm.ucar.edu/wrf/users/download/get_source.html

Unzip and untar the RIP4 tar file. The tar file contains the following directories and files:

- CHANGES, a text file that logs changes to the RIP tar file.
- Doc/, a directory that contains documentation of RIP, most notably the Users' Guide (ripug).
- **README**, a text file containing basic information on running RIP.
- **arch/**, directory containing the default compiler flags for different machines.
- **clean**, script to clean compiled code.
- **compile**, script to compile code.
- **configure**, script to create a configure file for your machine.
- **color.tbl**, a file that contains a table, defining the colors you want to have available for RIP plots.
- **eta_micro_lookup.dat**, a file that contains "look-up" table data for the Ferrier microphysics scheme.
- **psadilookup.dat**, a file that contains "look-up" table data for obtaining temperature on a pseudoadiabat.
- **sample_infiles/**, a directory that contains sample user input files for RIP and related programs.
- **src/**, a directory that contains all of the source code files for RIP, RIPDP, and several other utility programs.
- **stationlist**, a file containing observing station location information.

### Environment Variables

An important environment variable for the RIP system is **RIP_ROOT**. RIP_ROOT should be assigned the path name of the directory where all your RIP program and utility files (**color.tbl**, **stationlist**, lookup tables, etc.) reside. Typically (for cshrc shell):

```
setenv RIP_ROOT /my-path/RIP4
```

The RIP_ROOT environment variable can also be overwritten with the variable **rip_root** in the RIP user input file (UIF).

A second environment variable you need to set is **NCARG_ROOT**. Typically (for cshrc shell):

```
setenv NCARG_ROOT /usr/local/ncarg     ! for NCARG V4
setenv NCARG_ROOT /usr/local/ncl       ! for NCL V5
```

### Compiling RIP and associated programs

Since the release of version 4.5, the same configure/compile scripts available in all other WRF programs have been added to RIP4. To compile the code, first configure for your machine by typing:

```
./configure
```

You will see a list of options for your computer (*below is an example for a Linux machine*):
POST-PROCESSING

Will use NETCDF in dir: /usr/local/netcdf-pgi

Please select from among the following supported platforms.
1. PC Linux i486 i586 i686 x86_64, PGI compiler
2. PC Linux i486 i586 i686 x86_64, g95 compiler
3. PC Linux i486 i586 i686 x86_64, gfortran compiler
4. PC Linux i486 i586 i686 x86_64, Intel compiler

Enter selection [1-4]

Make sure the netCDF path is correct.
Pick compile options for your machine.

This will create a file called configure.rip. Edit compile options/paths, if necessary.

To compile the code, type:

    ./compile

After a successful compilation, the following new files should be created.

<table>
<thead>
<tr>
<th>rip</th>
<th>RIP post-processing program.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ripcomp</td>
<td>This program reads-in two rip data files and compares their content.</td>
</tr>
<tr>
<td>ripdp_mm5</td>
<td>RIP Data Preparation program for MM5 data</td>
</tr>
<tr>
<td>ripdp_wrfarw</td>
<td>RIP Data Preparation program for WRF data</td>
</tr>
<tr>
<td>ripinterp</td>
<td>This program reads-in model output (in rip-format files) from a coarse domain and from a fine domain, and creates a new file which has the data from the coarse domain file interpolated (bi-linearly) to the fine domain. The header and data dimensions of the new file will be that of the fine domain, and the case name used in the file name will be the same as that of the fine domain file that was read-in.</td>
</tr>
<tr>
<td>ripshow</td>
<td>This program reads-in a rip data file and prints out the contents of the header record.</td>
</tr>
<tr>
<td>showtraj</td>
<td>Sometimes, you may want to examine the contents of a trajectory position file. Since it is a binary file, the trajectory position file cannot simply be printed out. showtraj reads the trajectory position file and prints out its contents in a readable form. When you run showtraj, it prompts you for the name of the trajectory position file to be printed out.</td>
</tr>
</tbody>
</table>
**Preventing data with RIPDP**

RIP does not ingest model output files directly. First, a preprocessing step must be executed that converts the model output data files to RIP-format data files. The primary difference between these two types of files is that model output data files typically contain all times and all variables in a single file (or a few files), whereas RIP data has each variable at each time in a separate file. The preprocessing step involves use of the program RIPDP (which stands for RIP Data Preparation). RIPDP reads-in a model output file (or files), and separates out each variable at each time.

**Running RIPDP**

The program has the following usage:

```
rippo XXX [-n namelist file] model-data-set-name [basic|all] data_file_1 data_file_2 data_file_3 ...
```

Above, the "XXX" refers to "mm5", "wrfarw", or "wrfnmm".

The argument model-data-set-name can be any string you choose, that uniquely defines this model output data set.

The use of the namelist file is optional. The most important information in the namelist is the times you want to process.

As this step will create a large number of extra files, creating a new directory to place these files in will enable you to manage the files easier (mkdir RIPDP).

**e.g.** `rippo_wrfarw RIPDP/arw all wrfout_d01_*`
The RIP user input file

Once the RIP data has been created with RIPDP, the next step is to prepare the user input file (UIF) for RIP (see Chapter 4 of the RIP users’ guide for details). This file is a text file, which tells RIP what plots you want, and how they should be plotted. A sample UIF, called rip_sample.in, is provided in the RIP tar file. This sample can serve as a template for the many UIFs that you will eventually create.

A UIF is divided into two main sections. The first section specifies various general parameters about the set-up of RIP, in a namelist format (userin - which controls the general input specifications; and trajcalc - which controls the creation of trajectories). The second section is the plot specification section, which is used to specify which plots will be generated.

**namelist: userin**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idotitle</td>
<td>1</td>
<td>Controls first part of title.</td>
</tr>
<tr>
<td>title</td>
<td>‘auto’</td>
<td>Defines your own title, or allow RIP to generate one.</td>
</tr>
<tr>
<td>titlecolor</td>
<td>‘def.foreground’</td>
<td>Controls color of the title.</td>
</tr>
<tr>
<td>iinittime</td>
<td>1</td>
<td>Prints initial date and time (in UTC) on plot.</td>
</tr>
<tr>
<td>ifcsttime</td>
<td>1</td>
<td>Prints forecast lead-time (in hours) on plot.</td>
</tr>
<tr>
<td>ivalidtime</td>
<td>1</td>
<td>Prints valid date and time (in both UTC and local time) on plot.</td>
</tr>
<tr>
<td>inearesth</td>
<td>0</td>
<td>This allows you to have the hour portion of the initial and valid time be specified with two digits, rounded to the nearest hour, rather than the standard 4-digit HHMM specification.</td>
</tr>
<tr>
<td>timezone</td>
<td>-7.0</td>
<td>Specifies the offset from Greenwich time.</td>
</tr>
<tr>
<td>iusdaylightrule</td>
<td>1</td>
<td>Flag to determine if US daylight saving should be applied.</td>
</tr>
<tr>
<td>ptimes</td>
<td>9.0E+09</td>
<td>Times to process. This can be a string of times (e.g. 0,3,6,9,12,) or a series in the form of A,-B,C, which means &quot;times from hour A, to hour B, every C hours&quot; (e.g. 0,-12,3,). Either ptimes or iptimes can be used, but not both. You can plot all available times, by omitting both ptimes and iptimes from the namelist, or by setting the first value negative.</td>
</tr>
<tr>
<td>ptimeunits</td>
<td>‘h’</td>
<td>Time units. This can be ‘h’ (hours), ‘m’ (minutes), or ‘s’ (seconds). Only valid with ptimes.</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>iptimes</td>
<td>99999999</td>
<td>Times to process. This is an integer array that specifies desired times for RIP to plot, but in the form of 8-digit &quot;mdate&quot; times (i.e. YYYMMDDHH). Either ptimes or iptimes can be used, but not both. You can plot all available times by omitting both ptimes and iptimes from the namelist, or by setting the first value negative.</td>
</tr>
<tr>
<td>tacc</td>
<td>1.0</td>
<td>Time tolerance in seconds. Any time in the model output that is within tacc seconds of the time specified in ptimes/iptimes will be processed.</td>
</tr>
<tr>
<td>flmin, flmax, fbmin, fmax</td>
<td>.05, .95, .10, .90</td>
<td>Left, right, bottom and top frame limit</td>
</tr>
<tr>
<td>ntextq</td>
<td>0</td>
<td>Text quality specifier (0=high; 1=medium; 2=low).</td>
</tr>
<tr>
<td>ntextcd</td>
<td>0</td>
<td>Text font specifier [0=complex (Times); 1=duplex (Helvetica)].</td>
</tr>
<tr>
<td>fcoffset</td>
<td>0.0</td>
<td>This is an optional parameter you can use to &quot;tell&quot; RIP that you consider the start of the forecast to be different from what is indicated by the forecast time recorded in the model output. Examples: fcoffset=12 means you consider hour 12 in the model output to be the beginning of the true forecast.</td>
</tr>
<tr>
<td>idotser</td>
<td>0</td>
<td>Generates time-series output files (no plots); only an ASCII file that can be used as input to a plotting program.</td>
</tr>
<tr>
<td>idescriptive</td>
<td>1</td>
<td>Uses more descriptive plot titles.</td>
</tr>
<tr>
<td>icgmsplit</td>
<td>0</td>
<td>Splits metacode into several files.</td>
</tr>
<tr>
<td>maxfld</td>
<td>10</td>
<td>Reserves memory for RIP.</td>
</tr>
<tr>
<td>ittrajcalc</td>
<td>0</td>
<td>Generates trajectory output files (use namelist trajcalc when this is set).</td>
</tr>
<tr>
<td>imakev5d</td>
<td>0</td>
<td>Generate output for Vis5D</td>
</tr>
<tr>
<td>istopmiss</td>
<td>1</td>
<td>This switch determines the behavior for RIP when a user-requested field is not available. The default is to stop. Setting the switch to 0 tells RIP to ignore the missing field and to continue plotting.</td>
</tr>
<tr>
<td>rip_root</td>
<td>‘/dev/null’</td>
<td>Overwrites the environment variable RIP_ROOT.</td>
</tr>
</tbody>
</table>
Plot Specification Table

The second part of the RIP UIF consists of the Plot Specification Table. The PST provides all of the user control over particular aspects of individual frames and overlays.

The basic structure of the PST is as follows:
- The first line of the PST is a line of consecutive equal signs. This line, as well as the next two lines, is ignored by RIP. It is simply a banner that says this is the start of the PST section.
- After that, there are several groups of one or more lines, separated by a full line of equal signs. Each group of lines is a frame specification group (FSG), and it describes what will be plotted in a single frame of metacode. Each FSG must end with a full line of equal signs, so that RIP can determine where individual frames start and end.
- Each line within a FGS is referred to as a plot specification line (PSL). An FSG that consists of three PSL lines will result in a single metacode frame with three over-laid plots.

Example of a frame specification groups (FSG's):

```
feld=tmc; ptyp=hc; vcor=p; levs=850; >
cint=2; cmth=fill; cosq=-32,light.violet,-24,
violet,-16,blue,-8,green,0,yellow,8,red,>
16,orange,24,brown,32,light.gray
feld=ght; ptyp=hc; cint=30; linw=2
feld=uuu,vvv; ptyp=hv; vcmx=-1; colr=white; intv=5
feld=map; ptyp=hb
feld=tic; ptyp=hb
```

This FSG will generate 5 frames to create a single plot (as shown below):
- Temperature in degrees C (feld=tmc). This will be plotted as a horizontal contour plot (ptyp=hc), on pressure levels (vcor=p). The data will be interpolated to 850 hPa. The contour intervals are set to 2 (cint=2), and shaded plots (cmth=fill) will be generated with a color range from light violet to light gray.
- Geopotential heights (feld=ght) will also be plotted as a horizontal contour plot. This time the contour intervals will be 30 (cint=30), and contour lines with a line width of 2 (linw=2) will be used.
- Wind vectors (feld=uuu,vvv), plotted as barbs (vcmx=-1).
- A map background will be displayed (feld=map), and
- Tic marks will be placed on the plot (feld=tic).
Running RIP

Each execution of RIP requires three basic things: a RIP executable, a model data set and a user input file (UIF). The syntax for the executable, rip, is as follows:

\[
\text{rip} \ [-f] \ \text{model-data-set-name} \ \text{rip-execution-name}
\]

In the above, model-data-set-name is the same model-data-set-name that was used in creating the RIP data set with the program ripdp.

rip-execution-name is the unique name for this RIP execution, and it also defines the name of the UIF that RIP will look for.

The –f option causes the standard output (i.e., the textual print out) from RIP to be written to a file called rip-execution-name.out. Without the –f option, the standard output is sent to the screen.

\[\text{e.g. rip} \ -f \ \text{RIPDP/arw} \ \text{rip_sample}\]

If this is successful, the following files will be created:

- rip_sample.TYPE - metacode file with requested plots
- rip_sample.out - log file (if –f used); view this file if a problem occurred
The default output TYPE is a ‘cgm’, metacode file. To view these, use the command ‘idt’.

   e.g.     idt     rip_sample.cgm

For high quality images, create pdf or ps images directly (ncarg_type = pdf / ps).

See the Tools section in Chapter 10 of this User’s Guide for more information concerning other types of graphical formats and conversions between graphical formats.

Examples of plots created for both idealized and real cases are available from:
http://www2.mmm.ucar.edu/wrf/users/graphics/RIP4/RIP4.htm
ARWpost

The ARWpost package reads-in WRF-ARW model data and creates GrADS output files. *Since version 3.0 (released December 2010), vis5D output is no longer supported. More advanced 3D visualization tools, like VAPOR and IDV, have been developed over the last couple of years, and users are encouraged to explore those for their 3D visualization needs.*

The converter can read-in WPS geogrid and metgrid data, and WRF-ARW input and output files in netCDF format. *Since version 3.0 the ARWpost code is no longer dependant on the WRF IO API. The advantage of this is that the ARWpost code can now be compiled and executed anywhere without the need to first install WRF. The disadvantage is that GRIB1 formatted WRF output files are no longer supported.*

Necessary software

GrADS software - you can download and install GrADS from [http://grads.iges.org/](http://grads.iges.org/). The GrADS software is not needed to compile and run ARWpost, but is needed to display the output files.

Obtain the ARWpost TAR file from the WRF Download page ([http://www2.mmm.ucar.edu/wrf/users/download/get_source.html](http://www2.mmm.ucar.edu/wrf/users/download/get_source.html))

Unzip and untar the ARWpost tar file.

The tar file contains the following directories and files:

- `README`, a text file containing basic information on running ARWpost.
- `arch/`, directory containing configure and compilation control.
- `clean`, a script to clean compiled code.
- `compile`, a script to compile the code.
- `configure`, a script to configure the compilation for your system.
- `namelist.ARWpost`, namelist to control the running of the code.
- `src/`, directory containing all source code.
- `scripts/`, directory containing some grads sample scripts.
- `util/`, a directory containing some utilities.
**Environment Variables**

Set the environment variable NETCDF to the location where your netCDF libraries are installed. Typically *(for cshrc shell)*:

```bash
setenv NETCDF /usr/local/netcdf
```

**Configure and Compile ARWpost**

To configure - Type:

```bash
./configure
```

You will see a list of options for your computer *(below is an example for a Linux machine)*:

Will use NETCDF in dir: /usr/local/netcdf-pgi

Please select from among the following supported platforms.
1. PC Linux i486 i586 i686, PGI compiler
2. PC Linux i486 i586 i686, Intel compiler

Enter selection [1-2]

Make sure the netCDF path is correct.
Pick the compile option for your machine

To compile - Type:

```bash
./compile
```

If successful, the executable `ARWpost.exe` will be created.
Edit the namelist.ARWpost file

Set input and output file names and fields to process (**io**)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&amp;datetime</strong></td>
<td></td>
<td>Start and end dates to process. Format: YYYY-MM-DD HH:00:00</td>
</tr>
<tr>
<td>start_date;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interval_seconds</td>
<td>0</td>
<td>Interval in seconds between data to process. If data is available every hour,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and this is set to every 3 hours, the code will skip past data not required.</td>
</tr>
<tr>
<td>tacc</td>
<td>0</td>
<td>Time tolerance in seconds. Any time in the model output that is within tacc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>seconds of the time specified will be processed.</td>
</tr>
<tr>
<td>debug_level</td>
<td>0</td>
<td>Set this higher for more print-outs that can be useful for debugging later.</td>
</tr>
</tbody>
</table>

| **&io**        |       |                                                                            |
| input_root_name | ./    | Path and root name of files to use as input. All files starting with the root|
|                |       | name will be processed. Wild characters are allowed.                      |
| output_root_name | ./   | Output root name. When converting data to GrADS, output_root_name.ctl and  |
|                 |       | output_root_name.dat will be created.                                     |
| output_title   |       | Use to overwrite title used in GrADS .ctl file.                           |
| mercator_defs  | .False.| Set to true if mercator plots are distorted.                              |
| split_output   | .False.| Use if you want to split our GrADS output files into a number of smaller fil|
|                |       | es (a common .ctl file will be used for all .dat files).                   |
| frames_per_outfile | 1 | If split_output is .True., how many time periods are required per output (.dat) file. |
**Post-processing**

| plot | ‘all’ | Which fields to process.
|------|-------|-----------------------------
|      |       | ‘all’ – all fields in WRF file
|      |       | ‘list’ – only fields as listed in the ‘fields’ variable.
|      |       | ‘all_list’ – all fields in WRF file and all fields listed in the ‘fields’ variable.
|      |       | Order has no effect, i.e., ‘all_list’ and ‘list_all’ are similar.
|      |       | If ‘list’ is used, a list of variables must be supplied under ‘fields’. Use ‘list’ to calculate diagnostics.

| fields | Fields to plot. Only used if ‘list’ was used in the ‘plot’ variable.

| interp.method | 0 | 0 - sigma levels,
|              |   | -1 - code-defined "nice" height levels,
|              |   | 1 - user-defined height or pressure levels

| interp.levels | Only used if interp.method=1
|              | Supply levels to interpolate to, in hPa (pressure) or km (height). Supply levels bottom to top.

| extrapolate | .false. | Extrapolate the data below the ground if interpolating to either pressure or height.

*Available diagnostics:*

- **cape** - 3d cape
- **cin** - 3d cin
- **mcape** - maximum cape
- **mcin** - maximum cin
- **clfr** - low/middle and high cloud fraction
- **dbz** - 3d reflectivity
- **max_dbz** - maximum reflectivity
- **geopt** - geopotential
- **height** - model height in km
- **lcl** - lifting condensation level
- **lfc** - level of free convection
- **pressure** - full model pressure in hPa
- **rh** - relative humidity
- **rh2** - 2m relative humidity
- **theta** - potential temperature
- **tc** - temperature in degrees C
- **tk** - temperature in degrees K
- **td** - dew point temperature in degrees C
- **td2** - 2m dew point temperature in degrees C
**slp** - sea level pressure
**umet** and **vmet** - winds rotated to earth coordinates
**u10m** and **v10m** - 10m winds rotated to earth coordinates
**wdir** - wind direction
**wspd** - wind speed coordinates
**wd10** - 10m wind direction
**ws10** - 10m wind speed

**Run ARWpost**

Type:

```bash
./ARWpost.exe
```

This will create the `output_root_name.dat` and `output_root_name.ctl` files required as input by the GrADS visualization software.

**NOW YOU ARE READY TO VIEW THE OUTPUT**

For general information about working with GrADS, view the GrADS home page: [http://grads.iges.org/grads/](http://grads.iges.org/grads/)

To help users get started, a number of GrADS scripts have been provided:

- The scripts are all available in the `scripts/` directory.
- The scripts provided are only examples of the type of plots one can generate with GrADS data.
- The user will need to modify these scripts to suit their data (e.g., if you do not specify 0.25 km and 2 km as levels to interpolate to when you run the "bwave" data through the converter, the "bwave.gs" script will not display any plots, since it will specifically look for these levels).
- Scripts must be copied to the location of the input data.

**GENERAL SCRIPTS**

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cbar.gs</td>
<td>Plot color bar on shaded plots (from GrADS home page)</td>
</tr>
<tr>
<td>rgbset.gs</td>
<td>Some extra colors <em>(Users can add/change colors from color number 20 to 99)</em></td>
</tr>
</tbody>
</table>
skew.gs  Program to plot a skewT

TO RUN TYPE: run skew.gs (needs pressure level TC,TD,U,V as input)
User will be prompted if a hardcopy of the plot must be created (- 1 for
yes and 0 for no).
If 1 is entered, a GIF image will be created.
Need to enter lon/lat of point you are interested in
Need to enter time you are interested in
Can overlay 2 different times

plot_all.gs  Once you have opened a GrADS window, all one needs to do is run this
script.
It will automatically find all .ctl files in the current directory and list them
so one can pick which file to open.
Then the script will loop through all available fields and plot the ones a
user requests.

SCRIPTS FOR REAL DATA

real_surf.gs  Plot some surface data
Need input data on model levels
plevels.gs   Plot some pressure level fields
Need model output on pressure levels
rain.gs      Plot total rainfall
Need a model output data set (any vertical coordinate), that contain fields
"RAINC" and "RAINNC"
cross_z.gs   Need z level data as input
Will plot a NS and EW cross section of RH and T (C)
Plots will run through middle of the domain
zlevels.gs   Plot some height level fields
Need input data on height levels
Will plot data on 2, 5, 10 and 16km levels
input.gs     Need WRF INPUT data on height levels

SCRIPTS FOR IDEALIZED DATA

bwave.gs     Need height level data as input
Will look for 0.25 and 2 km data to plot
grav2d.gs    Need normal model level data
hill2d.gs    Need normal model level data
qss.gs       Need height level data as input.
Will look for heights 0.75, 1.5, 4 and 8 km to plot
sqx.gs       Need normal model level data a input
sqy.gs       Need normal model level data a input
Examples of plots created for both idealized and real cases are available from:
http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/ARWpost/

**Trouble Shooting**

*The code executes correctly, but you get "NaN" or "Undefined Grid" for all fields when displaying the data.*

Look in the .ctl file.

a) If the second line is:

```plaintext
options byteswapped
```

Remove this line from your .ctl file and try to display the data again.
If this SOLVES the problem, you need to remove the `-Dbytesw` option from `configure.arwp`

b) If the line below does NOT appear in your .ctl file:

```plaintext
options byteswapped
```

ADD this line as the second line in the .ctl file.
Try to display the data again.
If this SOLVES the problem, you need to ADD the `-Dbytesw` option for `configure.arwp`

The line "options byteswapped" is often needed on some computers (DEC alpha as an example). It is also often needed if you run the converter on one computer and use another to display the data.
NCEP Unified Post Processor (UPP)

UPP Introduction

The NCEP Unified Post Processor has replaced the WRF Post Processor (WPP). The UPP software package is based on WPP but has enhanced capabilities to post-process output from a variety of NWP models, including WRF-NMM, WRF-ARW, Non-hydrostatic Multi-scale Model on the B grid (NMMB), Global Forecast System (GFS), and Climate Forecast System (CFS). At this time, community user support is provided for the WRF-based systems and NMMB.

In addition to the option to output fields on the model’s native vertical levels, UPP interpolates output from the model’s native grids to National Weather Service (NWS) standard levels (pressure, height, etc.) and standard output grids (AWIPS, Lambert Conformal, polar-stereographic, etc.) in NWS and World Meteorological Organization (WMO) GRIB format. With the release of UPPv3.0, preliminary capabilities to output in GRIB Edition 2 (GRIB2) format for select models has been included and a simple template is available for users to modify to fit their needs. Caution should be taken when utilizing GRIB2; exhaustive testing has not been conducted and it is recommend to use this feature in testing/exploratory mode at this time. Updates will be provided as GRIB2 output capabilities become available and more comprehensive information will be included in the Users’ Guide.

UPP incorporates the Joint Center for Satellite Data Assimilation (JCSDA) Community Radiative Transfer Model (CRTM) to compute model derived brightness temperature ($T_B$) for various instruments and channels. This additional feature enables the generation of a number of simulated satellite products including GOES and AMSRE products for WRF-NMM, Hurricane WRF (HWRF), WRF-ARW and GFS. For CRTM documentation, refer to http://www.orbit.nesdis.noaa.gov/smcd/spb/CRTM.

UPP Software Requirements

The Community Unified Post Processor requires the same Fortran and C compilers used to build the WRF model. In addition, the netCDF library, the JasPer library, the PNG library, Zlib, and the WRF I/O API libraries, which are included in the WRF model tar file, are also required. UPP uses WRF I/O libraries for data processing of all models and as a result UPP is dependent on a WRF build. The JasPer library, PNG library, and Zlib are new requirements with the release of UPPv2.0 and higher, due to the addition GRIB2 capabilities. NCEP provides these necessary codes for download: http://www.nco.ncep.noaa.gov/pmb/codes/GRIB2/
The UPP has some sample visualization scripts included to create graphics using either GrADS (http://grads.iges.org/grads/grads.html) or GEMPAK (http://www.unidata.ucar.edu/software/gempak/index.html). These are not part of the UPP installation and need to be installed separately if one would like to use either plotting package.

UPP has been tested on LINUX platforms (with PGI, Intel and GFORTRAN compilers).

Obtaining the UPP Code

The UPP package can be downloaded from:

UPP Functionalities

The UPP,

- is compatible with WRF v3.3 and higher.
- can be used to post-process WRF-ARW, WRF-NMM, NMMB, GFS, and CFS forecasts (community support provided for WRF-based and NMMB forecasts).
- can ingest WRF history files (wrfout*) in two formats: netCDF and binary.
- can ingest NMMB history files (nmmb_hist*) in binary.

The UPP is divided into two parts:

Unipost

i. Interpolates the forecasts from the model’s native vertical coordinate to NWS standard output levels (e.g., pressure, height) and computes mean sea level pressure. If the requested parameter is on a model’s native level, then no vertical interpolation is performed.
ii. Computes diagnostic output quantities (e.g., convective available potential energy, helicity, relative humidity).
iii. Outputs the results in NWS and WMO standard GRIB1 format (for GRIB documentation, see http://www.nco.ncep.noaa.gov/pmb/docs/).
iv. Destaggers the WRF-ARW forecasts from a C-grid to an A-grid.
v. Outputs two navigation files, copygb_nav.txt (for WRF-NMM output only) and copygb_hwrf.txt (for WRF-ARW and WRF-NMM). These files can be used as input for copygb.

1. copygb_nav.txt: This file contains the GRID GDS of a Lambert Conformal Grid similar in domain and grid spacing to the one used to run the WRF-NMM. The Lambert Conformal map projection works well for mid-latitudes.
2. copygb_hwrf.txt: This file contains the GRID GDS of a Latitude-Longitude Grid similar in domain and grid spacing to the one used to
run the WRF model. The latitude-longitude grid works well for tropics.

Copygb
1. Destaggers the WRF-NMM forecasts from the staggered native E-grid to a regular non-staggered grid. (Since unipost destaggers WRF-ARW output from a C-grid to an A-grid, WRF-ARW data can be displayed directly without going through copygb.)
2. Destaggers the NMMB forecasts from the staggered native B-grid to a regular non-staggered grid.
3. Interpolates the forecasts horizontally from their native grid to a standard AWIPS or user-defined grid (for information on AWIPS grids, see http://www.nco.ncep.noaa.gov/pmb/docs/on388/tableb.html).
4. Outputs the results in NWS and WMO standard GRIB1 format (for GRIB documentation, see http://www.nco.ncep.noaa.gov/pmb/docs/).

Full UPP documentation available at:


VAPOR

VAPOR is the Visualization and Analysis Platform for Ocean, Atmosphere, and Solar Researchers. VAPOR was developed at NCAR to provide interactive visualization and analysis of numerically simulated fluid dynamics. The current (2.3) version of VAPOR has many capabilities for 3D visualization of WRF-ARW simulation output, including the ability to directly import wrfout files, and support for calculating derived variables that are useful in visualizing WRF output.

Basic capabilities of VAPOR with WRF-ARW output

- Direct Volume rendering (DVR)
  Any 3D variable in the WRF data can be viewed as a density. Users control transparency and color to view temperature, water vapor, clouds, etc. in 3D.

- Flow
  - Display barbs associated with 2D or 3D field magnitudes. Barbs can also be positioned at a specified height above the terrain and aligned to the WRF data grid.
  - Draw 2D and 3D streamlines and flow arrows, showing the wind motion and direction, and how wind changes in time.
- Path tracing (unsteady flow) enables visualization of trajectories that particles take over time. Users control when and where the particles are released.
- Flow images (image based flow visualization) can be used to provide an animated view of wind motion in a planar section, positioned anywhere in the scene.
- Field line advection can be used to animate the motion of streamlines of any vector field in a moving wind field.

- **Isosurfaces**
  The isosurfaces of variables are displayed interactively. Users can control iso-values, color and transparency of the isosurfaces. Isosurfaces can be colored according to the values of another variable.

- **Contour planes and Probes**
  3D variables can be intersected with arbitrarily oriented planes. Contour planes can be interactively positioned. Users can interactively pinpoint the values of a variable and establish seed points for flow integration. Wind and other vector fields can be animated in the probe plane.

- **Two-dimensional variable visualization**
  2D (horizontal) WRF variables can be color-mapped and visualized in the 3D scene. They can be viewed on a horizontal plane in the scene, or mapped onto the terrain surface.

- **Animation**
  Control the time-stepping of the data, for interactive replaying and for recording animated sequences.

- **Image display**
  Tiff images can be displayed in the 3D scene. If the images are georeferenced (i.e. geotiffs) then they can be automatically positioned at the correct latitude/longitude coordinates. Images can be mapped to the terrain surface, or aligned to an axis-aligned plane. Several useful georeferenced images are preinstalled with VAPOR, including political boundary maps, and the NASA Blue Marble earth image. VAPOR also provides several utilities for obtaining geo-referenced images from the Web. Images with transparency can be overlaid on the terrain images, enabling combining multiple layers of information.

- **Analysis capabilities**
  VAPOR (versions 2.1+) has an embedded Python calculation engine. Derived variables can be easily calculated with Python expressions or programs and these will be evaluated as needed for use in visualization. VAPOR provides Python scripts to calculate the following variables from WRF output:
  - CTT: Cloud-top temperature
  - DBZ: 3D radar reflectivity
  - DBZ_MAX: radar reflectivity over vertical column
ETH: equivalent potential temperature
RH: relative humidity
PV: potential vorticity
SHEAR: horizontal wind shear
SLP: 2D sea-level pressure
SHEAR: horizontal wind shear
TD: dewpoint temperature
TK: temperature in degrees Kelvin

Instructions for calculating and visualizing these and other variables are provided in the document, “Using Python with VAPOR” on the VAPOR website.

Derived variables can also be calculated in IDL and imported into the current visualization session. Variables can also be calculated in other languages (e.g. NCL) and adjoined to the Vapor Data Collection. Documentation of these capabilities can be found in the Documentation menu on the VAPOR website http://www.vapor.ucar.edu.

VAPOR requirements

VAPOR is supported on Linux, Mac, and Windows systems. VAPOR works best with a recent graphics card (say 1-2 years old). The advanced features of VAPOR perform best with nVidia™, ATI™ or AMD™ graphics accelerators.

VAPOR is installed on NCAR visualization systems. Users with UCAR accounts can connect their (Windows, Linux or Mac) desktops to the NCAR visualization systems using NCAR’s VNC-based remote visualization services, to run VAPOR and visualize the results remotely. Instructions for using NCAR visualization services are at: https://www2.cisl.ucar.edu/resources/geyser_caldera/visualization
Contact dasg@ucar.edu or vapor@ucar.edu for assistance.

VAPOR support resources

The VAPOR website: http://www.vapor.ucar.edu includes software, documentation, example data, and links to other resources. The document "Getting started with VAPOR and WRF" (http://www.vapor.ucar.edu/docs/getting-started-vapor/getting-started-vapor-and-wrf) has an overview of the various capabilities that are useful in visualizing WRF data with VAPOR.

The VAPOR Sourceforge website (http://sourceforge.net/projects/vapor) enables users to post bugs, request features, download software, etc.
Users of VAPOR on NCAR visualization systems should contact dasg@ucar.edu for support.

Users are encouraged to provide feedback. Questions, problems, bugs etc. should be reported to vapor@ucar.edu. The VAPOR development priorities are set by users as well as by the VAPOR steering committee, a group of turbulence researchers who are interested in improving the ability to analyze and visualize time-varying simulation results. Post a feature request to the VAPOR SourceForge website (http://sourceforge.net/projects/vapor), or e-mail vapor@ucar.edu if you have requests or suggestions about improving VAPOR capabilities.

Basic steps for using VAPOR to visualize WRF-ARW data

1. Install VAPOR

VAPOR installers for Windows, Macintosh and Linux are available on the VAPOR home page, http://www.vapor.ucar.edu/.

For most users, a binary installation is fine. Installation instructions are also provided in the VAPOR documentation pages, http://www.vapor.ucar.edu/docs/install.

After VAPOR is installed, it is necessary to perform user environment setup on Unix or Mac, before executing any VAPOR software. These setup instructions are provided on the VAPOR binary install documentation pages: http://www.vapor.ucar.edu/docs/install.

2. (Optional) Convert WRF output data to VAPOR Data Collection

Starting with VAPOR 2.0, you can directly load WRF-ARW output files into VAPOR. From the VAPOR menu select “Import data into current session---WRF-ARW”. Alternately, if your data is very large, you will be able to visualize it more interactively by converting it to a Vapor Data Collection (VDC).

A VAPOR VDC consists of (1) a metadata file (file type .vdf) that describes an entire VAPOR data collection, and (2) a directory of multi-resolution data files where the actual data is stored. The metadata file is created by the command wrfvdfcreate, and the multi-resolution data files are written by the command wrf2vdf. The simplest way to create a VAPOR data collection is using the vdcwizard application, which is installed with VAPOR. Also there are command-line tools wrfvdfcreate and wrf2vdf that can be used to convert the WRF-ARW output data to a VDC.

Using vdcwizard, you specify the name of the .vdf file and all the wrfout files that are to be used. First the .vdf file is created, then all the wrfout files are processed creating a VDC.
3. Visualize the WRF data

From the command line, issue the command “vaporgui”, or double-click the VAPOR desktop icon (on Windows or Mac). This will launch the VAPOR user interface.

To directly import WRF-ARW (NetCDF) output files, click on the Data menu, and select “Import WRF output files into default session”. Then select all the wrfout files you want to visualize and click “open”. If instead you converted your data to a VAPOR Data Collection, then, from the Data menu, choose “Load a dataset into default session”, and select the metadata file that you associated with your converted WRF data.
To visualize the data, select a renderer tab (DVR, Iso, Flow, 2D, Image, Barbs, or Probe), chose the variable(s) to display, and then, at the top of the tab, check the box labeled “Instance:1”, to enable the renderer. For example, the above top image combines volume, flow and isosurface visualization with a terrain image. The bottom image illustrates hurricane Ike, as it made landfall in 2008. The Texas terrain has a map of US Counties applied to it, and an NCL image of accumulated rainfall is shown at ground level in the current region.

4. Read the VAPOR Documentation


A short tutorial, showing how to use VAPOR to visualize hurricane Katrina WRF output files, is provided at [http://docs.vapor.ucar.edu/tutorials/hurricane-katrina](http://docs.vapor.ucar.edu/tutorials/hurricane-katrina).

Additional documents on the VAPOR website ([http://www.vapor.ucar.edu](http://www.vapor.ucar.edu)) provide more information about visualization of WRF data. Information is also available in the VAPOR user interface to help users quickly get the information they need, and showing how to obtain the most useful visualizations. Note the following resources:

- The Georgia Weather Case Study ([http://www.vapor.ucar.edu/sites/default/files/docs/GeorgiaCaseStudy.pdf](http://www.vapor.ucar.edu/sites/default/files/docs/GeorgiaCaseStudy.pdf)) provides a step-by-step tutorial, showing how to use most of the VAPOR features that are useful in WRF visualization. However this material is based on an older version of VAPOR.
- Creation of geo-referenced images to use with WRF data is discussed in the web document “Preparation of Georeferenced Images”.
  (http://www.vapor.ucar.edu/docs/vapor-data-preparation/preparation-georeferenced-images)

- "Using NCL with VAPOR to visualize WRF-ARW data":
  (http://www.vapor.ucar.edu/sites/default/files/docs/VAPOR-WRF-NCL.pdf)
  is a tutorial that shows how to create geo-referenced images from NCL plots, and to insert them in VAPOR scenes.

- Fuller documentation of the capabilities of the VAPOR user interface is provided in the VAPOR GUI General Guide:
  (http://www.vapor.ucar.edu/docs/vaporgui-help).

- The VAPOR Users' Guide for WRF Typhoon Research:
  (http://www.vapor.ucar.edu/sites/default/files/docs/Typhoon.pdf)
  provides a tutorial for using VAPOR on typhoon data, including instructions for preparing satellites images and NCL plots to display in the scene. This document is also fairly old.

To understand the meaning or function of an element in the VAPOR user interface:

**Tool tips**: Place the cursor over a widget for a couple of seconds and a one-sentence description will be provided.

**Context-sensitive help**: From the Help menu, click on “?Explain This”, and then click with the left mouse button on a GUI element, to obtain a longer technical explanation of the functionality.

**Web help**: From the vaporgui Help menu, various help topics associated with the current context can be selected. These will launch a Web browser displaying detailed information about the selected topic.
Chapter 10: Utilities and Tools

Table of Contents

- Introduction
- read_wrf_nc
- iowrf
- p_interp
- TC Bogus Scheme
- v_interp
- proc_oml.f
- Tools

Introduction

This chapter contains a number of short utilities to read and manipulate WRF-ARW data.

Also included in this chapter are references to some basic third party software, which can be used to view/change input and output data files.

read_wrf_nc

This utility allows a user to look at a WRF netCDF file at a glance.

What is the difference between this utility and the netCDF utility ncdump?

- This utility has a large number of options, to allow a user to look at the specific part of the netCDF file in question.
- The utility is written in Fortran 90, which will allow users to add options.
- This utility can be used for both WRF-ARW and WRF-NMM cores. It can be used for geogrid, metgrid and wrf input / output files. Only 3 basic diagnostics are available, pressure / height / tk, these can be activated with the -diag option (these are only available for wrfout files)

Obtain the read_wrf_nc utility from the WRF Download page (http://www2.mmm.ucar.edu/wrf/users/download/get_source.html)
Compile

The code should run on any machine with a netCDF library (*If you port the code to a different machine, please forward the compile flags to wrfhelp@ucar.edu*).

To compile the code, use the compile flags at the top of the utility.

e.g., for a LINUX machine you need to type:

```
pgf90 read_wrf_nc.f -L/usr/local/netcdf/lib
       -lnetcdf -lm -I/usr/local/netcdf/include
       -Mfree -o read_wrf_nc
```

If successful, this will create the executable: `read_wrf_nc`

Run

```
./read_wrf_nc  wrf_data_file_name  [-options]
```

**options :**  
[-h / help] [-att] [-m] [-M z] [-s]  
[-S x y z] [-v VAR] [-V VAR] [-w VAR]  
[-t t1 [t2]] [-times]  
[-ts xy X Y VAR VAR ....]  
[-ts ll lat lon VAR VAR ....]  
[-lev z] [-rot] [-diag]  
[-EditData VAR]

**Options:**  
(Note: options [-att] ; [-t] and [-diag] can be used with other options)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h / help</td>
<td>Print help information.</td>
</tr>
<tr>
<td>-att</td>
<td>Print global attributes.</td>
</tr>
<tr>
<td>-m</td>
<td>Print list of fields available for each time, plus the min and max values for each field.</td>
</tr>
<tr>
<td>-M z</td>
<td>Print list of fields available for each time, plus the min and max values for each field. The min and max values of 3d fields will be for the z level of the field.</td>
</tr>
<tr>
<td>-s</td>
<td>Print list of fields available for each time, plus a sample value for each field. Sample value is taken from the middle of model domain.</td>
</tr>
<tr>
<td>-S x y z</td>
<td>Print list of fields available for each time, plus a sample value for each field. Sample value is at point x y z in the model domain.</td>
</tr>
<tr>
<td>-t t1 [t2]</td>
<td>Apply options only to times t1 to t2. t2 is optional. If not set, options will only apply to t1.</td>
</tr>
<tr>
<td>-times</td>
<td>Print only the times in the file.</td>
</tr>
</tbody>
</table>
**-ts** Generate time series output. A full vertical profile for each variable will be created.
-ts xy X Y VAR VAR ..... will generate time series output for all VAR’s at location X/Y
-lev z Work only with option -ts
Will only create a time series for level z
-rot Work only with option -ts
Will rotate winds to Earth coordinates
-diag Add if you want to see output for the diagnostics temperature (K), full model pressure and model height (tk, pressure, height)
-v VAR Print basic information about field VAR.
-V VAR Print basic information about field VAR, and dump the full field out to the screen.
-w VAR Write the full field out to a file VAR.out
 Default Options are [-att –s]

**SPECIAL option: -EditData VAR**

This option allows a user to **read** a WRF netCDF file, **change** a specific field, and **write** it BACK into the WRF netCDF file.

This option will **CHANGE** your CURRENT WRF netCDF file so **TAKE CARE** when using this option.

ONLY one field at a time can be changed; therefore, if you need 3 fields changed, you will need to run this program 3 times, each with a different "VAR"

IF you have multiple times in your WRF netCDF file – **by default ALL times** for variable "VAR" WILL be changed. **If you only want to change one time period, also use the “-t” option.**

**HOW TO USE THIS OPTION:**

**Make a COPY of your WRF netCDF file before using this option**

**EDIT the subroutine USER_CODE**

ADD an IF-statement block for the variable you want to change. This is to prevent a variable getting overwritten by mistake.

For REAL data arrays, work with the array "data_real" and for INTEGER data arrays, work with the array "data_int".
Example 1:
If you want to change all (all time periods too) values of U to a constant 10.0 m/s, you would add the following IF-statement:
   else if ( var == 'U') then
   data_real = 10.0

Example 2:
If you want to change a section of the LANDMASK data to SEA points:
   else if ( var == 'LANDMASK') then
   data_real(10:15,20:25,1) = 0

Example 3:
Change all ISLTYP category 3 values into category 7 values (NOTE this is an INTEGER field):
   else if ( var == 'ISLTYP') then
   where (data_int == 3 )
   data_int = 7
   end where

Compile and run the program.
You will be asked if this is really what you want to do.
ONLY the answer "yes" will allow the change to take effect.
iowrf

This utility allows a user to do some basic manipulation on WRF-ARW netCDF files.

- The utility allows a user to thin the data; de-stagger the data; or extract a box from the data file.

Obtain the **iowrf utility** from the WRF Download page: (http://www2.mmm.ucar.edu/wrf/users/download/get_source.html).

**Compile**

The code should run on any machine with a netCDF library *(If you port the code to a different machine, please forward the compile flags to wrfhelp@ucar.edu)*.

To compile the code, use the compile flags at the top of the utility.

*e.g., for a LINUX machine you need to type:*

```
pgf90 iowrf.f -L/usr/local/netcdf/lib -lnetcdf -lm
-I/usr/local/netcdf/include -Mfree -o iowrf
```

If successful, this will create the executable: **iowrf**

**Run**

```
./iowrf  wrf_data_file_name  [-options]
```


<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-thina X</td>
<td>Thin the data with a ratio of 1:X Data will be averaged before being fed back</td>
</tr>
<tr>
<td>-thin X</td>
<td>Thin the data with a ratio of 1:X No averaging will be done</td>
</tr>
<tr>
<td>-box {}</td>
<td>Extract a box from the data file. X/Y/Z can be controlled independently. e.g.,</td>
</tr>
<tr>
<td></td>
<td>-box x 10 30 y 10 30 z 5 15</td>
</tr>
<tr>
<td></td>
<td>-box x 10 30 z 5 15</td>
</tr>
<tr>
<td></td>
<td>-box y 10 30</td>
</tr>
<tr>
<td></td>
<td>-box z 5 15</td>
</tr>
<tr>
<td>-A</td>
<td>De-stagger the data – no thinning will take place</td>
</tr>
<tr>
<td>-64bit</td>
<td>Allow large files (&gt; 2GB) to have read / write access</td>
</tr>
</tbody>
</table>
**p_interp**

This utility interpolates WRF-ARW netCDF output files to user-specified pressure levels. Several new capabilities have been supported in **p_interp** since October 2010. These includes:

- The ability to output fields needed to create met_em files, which can be used as input to real.exe. This output can be used to change the vertical resolution of WRF input files. Output from **p_interp** can also be used as input to TC bogusing or OBSGRID.
- A new namelist option is included to split input files containing multiple times into multiple output files, each with a separate time.
- **p_interp** can be compiled and ran in parallel to improve the time needed to processes large input files.
- Output from **p_interp** can now also be read directly by MET ([http://www.dtcenter.org/met/users/index.php](http://www.dtcenter.org/met/users/index.php)), removing the requirement to first run WPP before WRF-ARW data can be processed by the MET toolkit.

Obtain the **p_interp utility** from the WRF Download page: ([http://www2.mmm.ucar.edu/wrf/users/download/get_source.html](http://www2.mmm.ucar.edu/wrf/users/download/get_source.html)).

**Compile**

The code should run on any machine with a netCDF library. *If you port the code to a different machine, please forward the compile flags to wrfhelp@ucar.edu*

To compile the code, use the compile flags at the top of the utility.

*e.g., for a serial compile on a LINUX machine you need to type:*

```bash
pgf90 p_interp.F90 -L/usr/local/netcdf/lib
   -lnetcdf -lm -I/usr/local/netcdf/include
   -Mfree -o p_interp
```

*e.g., for a parallel compile on an IBM machine you need to type:*

```bash
mpxlf_r -qfree=f90 -L/usr/local/netcdf/lib -lnetcdf
   -lm -I/usr/local/netcdf/include -o p_interp
   p_interp.F90 -WF,-D_MPI
```

If successful, this will create the executable: **p_interp**
Edit the Namelist

Edit the associated namelist.pinterp file. (see namelist options below).

<table>
<thead>
<tr>
<th><strong>&amp;io</strong></th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path_to_input</td>
<td>./</td>
<td>Path to input data</td>
</tr>
<tr>
<td>input_name</td>
<td>None – must be set in namelist</td>
<td>File name(s) of wrfout files. Use wild character if more than one file is processed.</td>
</tr>
<tr>
<td>path_to_output</td>
<td>./</td>
<td>Path where output data will be written</td>
</tr>
<tr>
<td>output_name</td>
<td>' '</td>
<td>If no name is specified, the output will be written to input_name_PLEV</td>
</tr>
<tr>
<td>process</td>
<td>‘all’</td>
<td>Indicate which fields to process. ‘all’ fields in wrfout file (diagnostics PRES, TT, HGT &amp; RH will automatically be calculated); ‘list’ of fields as indicated in ‘fields’</td>
</tr>
<tr>
<td>fields</td>
<td>' '</td>
<td>List of fields to process, if ‘list’ is used in parameter ‘process’</td>
</tr>
<tr>
<td>debug</td>
<td>.false.</td>
<td>Set to .true. for more debugging</td>
</tr>
<tr>
<td>mpi_debug</td>
<td>.false.</td>
<td>Set to .true. for additional output that may be helpful when debugging parallel code.</td>
</tr>
<tr>
<td>bit64</td>
<td>.false.</td>
<td>Allow large files (&gt; 2GB) to have read / write access.</td>
</tr>
<tr>
<td>met_em_output</td>
<td>.false.</td>
<td>Set to .true. to calculate the output fields needed in a met_em file. These files are used as input to real.exe.</td>
</tr>
<tr>
<td>split_output</td>
<td>.false.</td>
<td>.true. will output each time in the input file to a separate output file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>&amp;interp_in</strong></th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interp_levels</td>
<td>-99999.</td>
<td>List of pressure levels to interpolate data to</td>
</tr>
<tr>
<td>extrapolate</td>
<td>0</td>
<td>0 - set values below ground and above model top to</td>
</tr>
</tbody>
</table>
missing values (default)  
1 - extrapolate below ground, and set above model top to model top values

| interp_method | 1 | 1 - linear in p-interpolation (default)  
2 - linear in log-p-interpolation
| unstagger_grid | .false. | Set to .true. to unstagger the data on output

If `met_em_output` is set to .true. in the namelist, other options also need to be set:

```
split_output   = .true.
unstagger_grid = .false.
extrapolate   = 1
process       = 'all'
```

If you do not set any of the first 3 options as shown above, they will be reset automatically in the code. If `process` is set to 'list', the code will stop and the user will have to set `process` to 'all'.

Also note that `p_interp` will stop if `met_em*` files already exist in the `path_to_output` directory. This is to reduce the change of overwriting any `met_em*` files created by `metgrid.exe`.

**Run**

To run `p_interp` compiled with the serial options, type

```
./p_interp
```

For distributed memory systems, some form of `mpirun` will be needed to run the executable. To run `p_interp` (compiled with parallel options) interactively, and using $x$ processors, the command may look like:

```
mpirun -np x ./p_interp
```

On some systems, parallel interactive jobs may not be an option, in which case the command would be

```
mpirun ./p_interp
```

to run in a batch script. On some IBM systems, the parallel job launcher may be `poe` or `mpirun.lsf`, rather than `mpirun`. 
TC Bogus Scheme

The ARW core for the WRF modeling system provides a simple Tropical Cyclone (TC) Bogussing scheme. It can remove an existing tropical storm, and may optionally bogus in a Rankine vortex for the new tropical storm. The input to the program is a single time-period and single domain of metgrid data, and a few namelist variables from the namelist.input file that describes the bogus TC’s location and strength. The output is also a metgrid-like file. The scheme is currently only set up to process isobaric data. After running the tc.exe program, the user must manually rename the files so that the real.exe program can read the modified input.

Namelist Options

The namelist information for the TC scheme is located in an optional namelist record &tc. Only a single domain is processed. Users with multiple domains should horizontally-interpolate the generated meteorological fields to the fine-grid domains. Alternatively, users may run the tc.exe program on separate metgrid output files for different domains, though this is not recommended.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert_bogus_storm</td>
<td>logical, insert a bogus storm</td>
<td></td>
</tr>
<tr>
<td>remove_storm</td>
<td>logical, removes an existing storm</td>
<td></td>
</tr>
<tr>
<td>num_storm</td>
<td>integer</td>
<td>number of storms to bogus, currently must be set to 1</td>
</tr>
<tr>
<td>latc_loc</td>
<td>real</td>
<td>latitude of bogus storm (+ north, - south)</td>
</tr>
<tr>
<td>lonc_loc</td>
<td>real</td>
<td>longitude of bogus storm (+ east, - west)</td>
</tr>
<tr>
<td>vmax_meters_per_second</td>
<td>real</td>
<td>maximum observed sustained wind speed (m/s)</td>
</tr>
<tr>
<td>rmax</td>
<td>real</td>
<td>radius from the cyclone center to where the maximum wind speed occurs (m)</td>
</tr>
<tr>
<td>vmax_ratio</td>
<td>real</td>
<td>scale factor for model’s Rankine vortex</td>
</tr>
</tbody>
</table>

Note: If insert_bogus_storm is set to true then remove_storm should be set to false. If remove_storm is set to true then insert_bogus_storm should be set to false.

The value for vmax_ratio should be about 0.75 for a 45-km domain and about 0.90 for a 15-km domain (use these values to interpolate other for other resolutions). This is a representativeness scale factor. The observed maximum wind speed is not appropriate for an entire grid cell when the domain is fairly coarse.

For example, assume that a cyclone report came in with the storm centered at 25° N and 75° W, where the maximum sustained winds were observed to be 120 kts, with the maximum winds about 90 km from the storm center. With a 45-km coarse grid model domain, the namelist.input file would be:
&tc
  insert_bogus_storm = .true.
  remove_storm = .false.
  latc_loc = 25.0
  lonc_loc = -75.0
  vmax_meters_per_second = 61.7
  rmax = 90000.0
  vmax_ratio = 0.75
/

Program tc.exe

The program tc.exe is automatically built along with the rest of the ARW executables. This, however, is a serial program. For the time being, it is the best to build this program using serial and no-nesting options.

Running tc.exe

1) Run all of the WPS programs as normal (geogrid, ungrib, and metgrid).
2) As usual, link-in the metgrid output files into either the test/em_real or the run directory.
3) Edit the namelist.input file for usage with the tc.exe program. Add-in the required fields from the &tc record, and only process a single time period.
4) Run tc.exe
5) Rename the output file, auxinput1_d01_<date> to the name that the real.exe program expects, met_em.d01.<date>, note that this will overwrite your original metgrid.exe output file for the initial time period.
6) Edit the namelist.input file to process all of the time periods for the real.exe program.
v_interp

This utility can be used to add vertical levels in WRF-ARW netCDF input. An example of the usage would be one-way nesting, via the program ndown. Since the program ndown does not do ‘vertical nesting’ prior to Version 3.2, namely adding vertical levels, this program can be used after running ndown to achieve the same results. Starting from Version 3.2, vertical levels may be added in the program ndown, via the namelist option ‘vert_refine_fact’, which allows one to refine vertical levels by an integer factor.

The v_interp utility program can be obtained from the WRF Download page:
(http://www2.mmm.ucar.edu/wrf/users/download/get_source.html)

Compile

The code should be easily built and ran on any machine with a netCDF library. To compile the code, use the compile flags shown at the top of the utility program.

e.g., for a LINUX machine and pgf90 compiler, one may type:

```
pgf90 v_interp.f -L/usr/local/netcdf/lib -lnetcdf \
   -I/usr/local/netcdf/include \
   -Mfree -o v_interp
```

If successful, this will create the executable: v_interp

Run

Edit the namelist file namelist.v_interp (see namelist options below) for the number of new vertical levels (nvert) and the new set of levels (nlevels). To find out the existing model levels, check the original WRF namelist.input file used to create the input files, or type the following:

```
ncdump -v ZNW wrfinput_d01
```

The executable takes two arguments on the command line:

```
./v_interp file file_new
```
where file is the input file you want to add the vertical levels to, and file_new is the output file that contains more vertical levels. To run the program for wrfinput file, type

```
./v_interp wrfinput_d01 wrfinput_d01_new
```

For the wrfbdy file, type

```
./v_interp wrfbdy_d01 wrfbdy_d01_new
```

**namelists:**

<table>
<thead>
<tr>
<th>&amp;newlevels</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvert</td>
</tr>
<tr>
<td>nlevels</td>
</tr>
</tbody>
</table>

**Program Notes:**

When adding vertical levels, please keep the first- and the last-half levels the same as in the input file, itself. A problem may occur if levels are added outside the range.

For the wrfbdy file, please keep the input file name as wrfbdy_*. Since the program keys-in on the file name in order to do the interpolation for special boundary arrays.
proc_oml.f

This utility may be used to process 3D HYCOM (http://www.hycom.org) ocean model temperature data in netCDF format to produce initial ocean mixed layer depth field (HOML) for use in a WRF simulation that uses the simple ocean mixed layer model option (omlcall = 1, and oml_hml0 < 0). The program estimates two fields from the HYCOM data: 1) effective mixed layer depth based on the idea of ocean heat content (HOML); and 2) mean ocean temperature in the top 200 m depth (TMOML). This is used as the lower limit for cooling SST’s in the wake of a hurricane.

To download the proc_oml.f utility, please see:
http://www2.mmm.ucar.edu/wrf/users/hurricanes/util.html

Compile

To compile the code, use the compile flags shown at the top of the utility program. For example, for a LINUX machine and pgf90 compiler one may type:

```
pgf90 proc_oml.f -L/usr/local/netcdf/lib -lnetcdf \ 
              -I/usr/local/netcdf/include -Mfree -o proc_oml.f
```

If successful, this will create the executable: proc_oml

Run

To run the program, type

```
./proc_oml ocean-data-file.nc yyyymmddhh
```

where ‘ocean-data-file.nc’ is the HYCOM ocean data file, and yyyymmddhh is the 10-digit date when the data is valid for (e.g. 2005082700). Successfully running the program will produce an output file, MLD, which is in intermediate format as if it were produced by the WPS/ungrib program.

To use this field in WPS/metgrid, add it to ‘constant_name’ as below:

```
constant_name = ‘MLD’,
```

V3.2 WPS/metgrid has the additional fields in METGRID.TBL for proper horizontal interpolation. For more information, please refer to the following presentation, at http://www2.mmm.ucar.edu/wrf/users/tutorial/hurricanes/AHW_nest_ocean.pdf
Tools

Below is a list of tools that are freely available, and can be used very successfully to manipulate model data (both WRF model data, as well as other GRIB and netCDF datasets).

Converting Graphics

**ImageMagick**

ImageMagick is a software suite to create, edit, and compose bitmap images. It can read, convert and write images in a variety of formats (over 100) including DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PhotoCD, PNG, Postscript, SVG, and TIFF. Use ImageMagick to translate, flip, mirror, rotate, scale, shear and transform images, adjust image colors, apply various special effects, or draw text, lines, polygons, ellipses and Bzier curves.

The software package is freely available from, [http://www.imagemagick.org](http://www.imagemagick.org). Download and installation instructions are also available from this site.

Examples of converting data with ImageMagick software:

```
convert file.pdf file.png
convert file.png file.bmp
convert file.pdf file.gif
convert file.ras file.png
```

ImageMagick cannot convert ncgm (NCAR Graphics) file format to other file formats.

**Converting ncgm (NCAR Graphics) file format**

NCAR Graphics has tools to convert ncgm files to raster file formats. Once files are in raster file format, ImageMagick can be used to translate the files into other formats.

For `ncgm` files containing a single frame, use `ctrans`.

```
ctrans -d sun file.ncgm file.ras
```

For `ncgm` files containing multiple frames, first use `med` (metafile frame editor) and then `ctrans med` will create multiple single frame files called `medxxx.ncgm`

```
med -e '1, $ split $' file.ncgm
```

```
ctrans -d sun med001.ncgm > med001.ras
```
Basic Unix Commands

The WRF model is run on any Unix/Linux machine. Some basic Unix commands are required to work in this environment. There are numerous web sites one can visit to learn more about basic and advanced Unix commands. A couple of basic Unix commands are listed below, as well as some web sites where users can obtain more information.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mkdir</code></td>
<td>To make (mkdir) or remove (rmdir) directories.</td>
</tr>
<tr>
<td><code>cd</code></td>
<td>To change to a new directory.</td>
</tr>
<tr>
<td><code>ls</code></td>
<td>List the files and directories in a directory.</td>
</tr>
<tr>
<td><code>ls -l</code></td>
<td>Lists your files in 'long format', which contains lots of useful information, e.g. the exact size of the file, who owns the file and who has the right to look at it, and when it was last modified.</td>
</tr>
<tr>
<td><code>ls --lrt</code></td>
<td>Lists your files in 'long format', in order of time stamp, and reverse order.</td>
</tr>
<tr>
<td><code>rm</code></td>
<td>Remove files.</td>
</tr>
<tr>
<td><code>more</code></td>
<td>Shows the first part of a file, just as much as will fit on one screen. Just hit the space bar to see more or <code>q</code> to quit.</td>
</tr>
<tr>
<td><code>cat</code></td>
<td>Shows the entire file on the screen.</td>
</tr>
<tr>
<td><code>head</code></td>
<td>Shows the first couple of lines of a file on screen.</td>
</tr>
<tr>
<td><code>tail</code></td>
<td>Shows the last couple of lines of a file on screen.</td>
</tr>
<tr>
<td><code>grep</code></td>
<td>Find lines that match patterns in files.</td>
</tr>
<tr>
<td><code>mv</code></td>
<td>Rename or move a file.</td>
</tr>
<tr>
<td><code>cp</code></td>
<td>Copy a file to a different name or location.</td>
</tr>
<tr>
<td><code>pwd</code></td>
<td>Shows the directory path you are currently in.</td>
</tr>
<tr>
<td><code>ln -sf</code></td>
<td>Makes a symbolic (-s) link (ln) of a file. The file will appear to be in two locations, but is only physically in one location. <code>The -f option ensures that if the target file already exists, then it will first be unlink so that the link may occur correctly.</code></td>
</tr>
<tr>
<td><code>vi / emacs</code></td>
<td>File editors. For new users, <code>emacs</code> may be an easier editor to work with, as <code>vi</code> requires some extra understanding to navigate between the <code>command</code> and <code>insert</code> modes, whereas <code>emacs</code> functions more like a conventional editor.</td>
</tr>
</tbody>
</table>

http://mally.stanford.edu/~sr/computing/basic-unix.html
http://www.math.harvard.edu/computing/unix/unixcommands.html
http://www.washington.edu/computing/unix/unixqr.html
http://www.cs.colostate.edu/helpdocs/vi.html
Design WRF model domains

**WPS/util/plotgrids.ncl**
Is an NCL script, which can either plot the domain on screen, or create a variety of different output types (pdf, ps, ncgm). This script must be ran in the same directory where the namelist.wps resides. This script only works with NCL version 6.1.0 or newer. If you still have an older version of NCL you can still use the plotgrids_old.ncl script.

Read more about this utility in Chapter 3 of this Users Guide.

Display ungrib (intermediate) files

**WPS/util/plotfmt.ncl**
Is an NCL script that can be used to display intermediate files created by WPS/ungrib.exe.

If you have created intermediate files manually, it is a very good practice to use this utility to display the data in your files first before running WPS/metgrid.exe.

**Note:** If you plan on manually creating intermediate files, refer to [http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Basics/IM_files/index.html](http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Basics/IM_files/index.html) for detailed information about the file formats and sample programs.

This script reads intermediate files and can output the graphics in a variety of different output formats (on the screen, pdf, ps, ncgm). The script requires NCL version 6.2.0 or newer. An input file much be supplied, e.g:

```
nc1 plotfmt.ncl 'filename="FILE:2005-06-01_00"
```

**WPS/util/int2nc.exe**, can be used to convert intermediate files created by WPS/ungrib.exe into netCDF files.

**WPS/util/plotfmt_nc.ncl**
Is an NCL script, which can plot the netCDF output files created by int2nc.exe. This script must be run in the same directory where the netCDF files reside. The file to be plotted should be entered on the command line, e.g.,

```
ncl plotfmt_nc.ncl ‘inputFILE="FILE:2005-06-01_00.nc"
```

Read more about these utilities in **Chapter 3** of this Users Guide.
netCDF data

netCDF stands for network Common Data Form. Most of the information below can be used for WRF netCDF data, as well as other netCDF datasets. netCDF is one of the current supported data formats chosen for WRF I/O API.

Advantages of using netCDF?
Most graphical packages support netCDF file formats
netCDF files are platform-independent (big-endian / little-endian)
A lot of software already exists that can be used to process/manipulate netCDF data

Documentation:
http://www.unidata.ucar.edu/ (General netCDF documentation)

Utilities:
ncdump
This is part of the netCDF libraries. Reads a netCDF file and prints information about the dataset. e.g.
ncdump -h file (print header information)
ncdump -v VAR file (print header information and the full field VAR)
ncdump -v Times file (a handy way to see how many times are available in a WRF output file)

ncview
Displays netCDF data graphically. No overlays, no maps and no manipulation of data possible.
http://meteora.ucsd.edu/~pierce/ncview_home_page.html

ncBrowse
Displays netCDF data graphically. Some overlays, maps and manipulation of data are possible.
http://www.epic.noaa.gov/java/ncBrowse/

read_wrf_nc
A utility to display basic information about WRF netCDF files.

iowrf
A utility to do some basic file manipulation on WRF-ARW netCDF files.
**p_interp**
A utility to interpolate WRF-ARW netCDF output files to user specified pressure levels.

**netCDF operators**
http://nco.sourceforge.net/
Stand-alone programs that can be used to manipulate data (by performing grid point averaging / file differencing / file ‘appending’). *A couple of available programs are listed below, see the above link for a list of all available programs.*

**ncdiff**
Difference between two files; e.g.
```
ncdiff  input1.nc input2.nc output.nc
```

**ncrcat**
Writes specified variables / times to a new file; e.g.
```
ncrcat -v RAINNC wrfout* RAINNC.nc
ncrcat -d Time,0,231 -v RAINNC wrfout* RAINNC.nc
```

**ncra**
Averages variables and writes to a new file; e.g.
```
ncra -v OLR wrfout* OLR.nc
```

**ncs (nc kitchen sink)**
Combination of NCO tools all in one (handy: one tool for multiple operations). An especially handy use of this tool is to split large files into smaller files, e.g.
```
ncs -A -F -d Time,1,1 wrfout* -o wrfout_timel.nc
```
GRIB data

**Documentation and Decoders**
Documentation and decoders for both GRIB1 and GRIB2 can be found here: [http://rda.ucar.edu/#!GRIB](http://rda.ucar.edu/#!GRIB)
Some of the utilities that are worth looking at is the unpackgrib2.c and grib2to1.c code.

**GRIB codes**

It is important to understand the GRIB codes to know which fields are available in your dataset. For instance, NCEP uses the GRIB1 code 33 for the U-component of the wind, and 34 for the V-component. *Other centers may use different codes, so always obtain the GRIB codes from the center you get your data from.*

GRIB2 uses 3 codes for each field - **product**, **category** and **parameter**. We would most often be interested in **product 0** (*Meteorological products*). **Category** refers to the type of field; e.g., category 0 is temperature, category 1 is moisture and category 2 is momentum. **Parameter** is the field number.
So whereas GRIB1 only uses code 33 for the U-component of the wind, GRIB2 will use 0,2,2, for the U-component, and 0,2,3 for the V-component.

**Display GRIB header/field information**

- **GRIB1 data**
  - WPS/util/g1print.exe
  - wgrrib

- **GRIB2 data**
  - WPS/util/g2print.exe
  - wgrrib2

Both wgrrib and wgrrib2 are available from the [http://rda.ucar.edu/#!GRIB](http://rda.ucar.edu/#!GRIB) web site.

**Convert GRIB data to netCDF format**


**Displaying GRIB files**

Model Verification

MET is designed to be a highly configurable, state-of-the-art suite of verification tools. It was developed using output from the Weather Research and Forecasting (WRF) modeling system, but may be applied to the output of other modeling systems as well.

MET provides a variety of verification techniques, including:
- Standard verification scores, comparing gridded model data to point-based observations
- Standard verification scores, comparing gridded model data to gridded observations
- Object-based verification method, comparing gridded model data to gridded observations

http://www.dtcenter.org/met/users/index.php
Appendix A: WRF-Fire

Table of Contents

- Introduction
- WRF-Fire in idealized cases
- Fire variables in namelist.input
- namelist.fire
- Running WRF-Fire on real data
  - Building the code
  - Fire variables in namelist.wps
  - Geogrid
  - Conversion to geogrid format
  - Editing GEOGRID.TBL
  - Ungrib and Metgrid
  - Running real case and WRF-Fire
- Fire state variables
- WRF-Fire software
  - WRF-Fire coding conventions
  - Parallel execution
  - Software layers
  - Initialization in idealized case

Introduction

A wildland fire module named WRF-Fire has been added to WRF ARW to allow users to model the growth of a wildland fire and the dynamic feedbacks with the atmosphere. It is implemented as a physics package with two-way coupling between the fire behavior and the atmospheric environment allowing the fire to alter the atmosphere surrounding it, i.e. ‘create its own weather’. Here we address the mechanics, options, parameters, and datasets for using this module.

The wildland fire module is currently a simple two-dimensional model of a surface fire, that is, a fire that spreads through fuels on the ground, such as grass, shrubs, and the litter from trees that has fallen to the surface. It does not yet contain the algorithms needed to represent crown fires, which consume and spread through the tree canopies. The user specifies the time, location, and shape of a fire ignition. The evolution of the fireline, the interface enclosing the burning region, is implemented by the level set method. The level set function is advanced by the Runge-Kutta method of order 2, with spatial discretization by the Godunov method. The rate at which this interface expands is calculated at all
points along it using a point-based semi-empirical formula for estimating the rate of spread of the surface fire based upon the Rothermel (1972) formula, which calculates the fire rate of spread as a function of local fuel conditions, wind, and terrain slope. A semi-empirical formula is used as a parameterization since turbulent combustion cannot be resolved at the spatial scales of atmospheric models; thus, all physical processes involved in propagating the fire are assumed to be represented in this relationship. Importantly, the winds used to drive the fire are interpolated from nearby low-level wind velocities, which are themselves perturbed by the fire. Once the fireline has passed by, the ignited fuel continues to burn - the mass of fuel is assumed to decay exponentially with time after ignition, the rate depending on the size of the fuel particles making up the fuel complex: fine fuels such as grass are consumed rapidly, while fuels with larger diameters such as twigs and downed logs are consumed slowly. The fuel burned in each time step is converted to sensible and latent heat source terms for the lowest levels of the WRF atmospheric model state, where the water vapor source arises from the release of the intrinsic moisture in cellulosic fuels and the additional moisture absorbed by fuels from their environment, the fuel moisture content. The e-folding depth over which the heat and vapor distributed is set by the user, based on results from wildland fire measurements. The fire may not progress to locations where the local fuel moisture content is greater than the moisture content of extinction.

Additional parameters and datasets beyond a standard WRF atmospheric simulation are required and are described here. The surface fuel available to be burned at each point is categorized using the Anderson classification system for “fuel models” (3 grass-dominated types, 4 shrub-dominated types, 3 types of forest litter, and 3 levels of logging slash) which we will henceforth refer to as “fuel categories” to limit confusion. Each of these fuel categories is assigned a set of typical properties consisting of the fuel load (the mass per unit area) and numerous physical properties having to do with fuel geometry, arrangement, and physical makeup. The user may make the fuels spatially homogeneous by using one fuel category for the whole domain, alter fuel properties, add custom fuel categories, or (for real data experiments) project a spatially heterogeneous map of fuel categories onto the domain from fuel mapping datasets. The user also sets the number of ignitions, their time, location, and shape, and the fuel moisture content, an important factor in fire behavior.

One time step of the fire model is performed for every WRF time step. The fire model runs on a refined grid that covers the same region as the innermost WRF domain. The fire module supports both distributed and shared memory parallel execution.

Other References


- The original report introducing Rothermel’s semi-empirical formulas (Rothermel,

- The following paper describes the WRF-Fire module and applies WRF with WRF-Fire in simulations to test the sensitivity of fire growth to environmental factors such as wind speed, fuel load and moisture, and fuel model in the daytime convective boundary layer:


### WRF-Fire in idealized cases

To perform a simulation including a fire, follow the installation instructions in Chapter 5 to configure WRF and set up the environment. For an idealized case, use

```
./compile em_fire
```

to build WRF for one of the several supplied ideal examples. This will create the links `wrf.exe` and `ideal.exe` in the directory `test/em_fire`. The examples are in its subdirectories. The links `wrf.exe` and `ideal.exe` in the subdirectories point to the parent directory.

The directory `test/em_fire` contains the directories `hill_simple` and `two_fires`. Each directory contains all files needed to run the example, namely `namelist.input`, `namelist.fire`, and `input_sounding`.

To run the `hill_simple` idealized example, type

```
cd test/em_fire

cp -f /hill_simple/* .

./ideal.exe

./wrf.exe
```

The file `namelist.input` contains an additional section `&fire` with parameters of the fire model and ignition coordinates. The file `namelist.fire` contains an additional namelist used to enter custom fuel properties.
## Fire variables in namelist.input

<table>
<thead>
<tr>
<th>Variable names</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;domains</td>
<td></td>
<td>Domain definition</td>
</tr>
<tr>
<td>sr_x</td>
<td>10</td>
<td>The fire mesh is 10 times finer than the innermost atmospheric mesh in the ( x ) direction. This number must be even.</td>
</tr>
<tr>
<td>sr_y</td>
<td>10</td>
<td>The fire mesh is 10 times finer than the innermost atmospheric mesh in the ( y ) direction. This number must be even.</td>
</tr>
<tr>
<td>&amp;fire</td>
<td></td>
<td>Fire ignition and fuel parameters</td>
</tr>
<tr>
<td>ifire</td>
<td>0</td>
<td>No fires will be simulated.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Fires will be simulated, using the tracer scheme to represent the flaming front interface (not active).</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Fires will be simulated, using the level set method to represent the movement of the interface.</td>
</tr>
<tr>
<td>fire_fuel_read</td>
<td>0</td>
<td>How to set the fuel data -1: real data from WPS 0: set to a homogeneous distribution of fire_fuel_cat everywhere 1: The spatial distribution of fuel categories is to be specified as a function of terrain altitude. (The user specifies a custom function.)</td>
</tr>
<tr>
<td>fire_num_ignitions</td>
<td>3</td>
<td>Number of ignition lines, max. 5 allowed</td>
</tr>
<tr>
<td>fire_ignition_start_x1</td>
<td>1000.</td>
<td>( x ) coordinate of the start point of the ignition line 1. All ignition coordinates are given in m from the lower left corner of the innermost domain</td>
</tr>
<tr>
<td>fire_ignition_start_y1</td>
<td>500.</td>
<td>( x ) coordinate of the start point of the ignition line 1</td>
</tr>
<tr>
<td>fire_ignition_end_x1</td>
<td>1000.</td>
<td>( y ) coordinate of the end point of the ignition line 1. Point ignition (actually a small circle) is obtained by specifying the end point the same as the start point.</td>
</tr>
<tr>
<td>fire_ignition_end_y1</td>
<td>1900.</td>
<td>( y ) coordinate of the end point of the ignition line 1</td>
</tr>
</tbody>
</table>
fire_ignition_radius1 18.  Everything within
fire_ignition_radius1 meters from
the ignition location will be ignited.

fire_ignition_time1 2.  Time of ignition in s since the start of the run

fire_ignition_start_x2 ...  Up to 5 ignition lines may be given. Ignition
parameters with the number higher than
fire_num_ignitions are ignored.

fire_ignition_time5
fire_print_msg 1 0: no messages from the fire scheme
1: progress messages from the fire scheme

fire_print_file 0 0: no files written (leave as is)
1: fire model state written every 10 s into
files that can be read in Matlab.

There are several more variables in the namelist for developers’ use only to further
develop and tune the numerical methods. Do not alter unless directed to do so.

**namelist.fire**

This file serves to redefine the fuel categories if the user wishes to alter the default fuel
properties.

<table>
<thead>
<tr>
<th>Variable names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;fuel_scalars</td>
<td>Scalar fuel constants</td>
</tr>
<tr>
<td>cmbcnst</td>
<td>The energy released per unit fuel burned for cellulosic fuels (constant, 1.7433e7 J kg(^{-1})).</td>
</tr>
<tr>
<td>hfgl</td>
<td>The threshold heat flux from a surface fire at which point a canopy fire is ignited above (in W m(^{-2})).</td>
</tr>
<tr>
<td>fuelmc_g</td>
<td>Surface fuel, fuel moisture content (in percent expressed in decimal form, from 0.00 – 1.00).</td>
</tr>
<tr>
<td>nfuelcats</td>
<td>Number of fuel categories defined (default: 13)</td>
</tr>
<tr>
<td>no_fuel_cat</td>
<td>The number of the dummy fuel category specified to be used where there is no fuel</td>
</tr>
<tr>
<td>&amp;fuel_categories</td>
<td>Domain specifications</td>
</tr>
<tr>
<td>fgi</td>
<td>The initial mass loading of surface fuel (in kg m(^{-2})) in each fuel category</td>
</tr>
<tr>
<td>fueldepthm</td>
<td>Fuel depth (m)</td>
</tr>
<tr>
<td>savr</td>
<td>Fuel surface-area-to-volume-ratio (m(^{-1}))</td>
</tr>
<tr>
<td>fuelmce</td>
<td>Fuel moisture content of extinction (in percent expressed in decimal form, from 0.00 – 1.00).</td>
</tr>
</tbody>
</table>
fueldens | Fuel particle density lb ft$^{-3}$ (32 if solid, 19 if rotten)
---|---
st | Fuel particle total mineral content. (kg minerals/kg wood)
se | Fuel particle effective mineral content. (kg minerals – kg silica)/kg wood
weight | Weighting parameter that determines the slope of the mass loss curve. This can range from about 5 (fast burn up) to 1000 (i.e. a 40% decrease in mass over 10 minutes).
ichap | Is this a chaparral category to be treated differently using an empirical rate of spread relationship that depends only on wind speed? (1: yes, this is a chaparral category and should be treated differently; 0: no, this is not a chaparral category or should not be treated differently). Primarily used for Fuel Category 4.

---

**Running WRF-Fire on real data**

**Building the code**

Running WRF with real data is a complicated process of converting data formats and interpolating to the model grid. This process is simplified by the WRF Preprocessing System (WPS). The purpose of this section is to summarize the use of this system and to highlight the differences in its use between fire and ordinary atmospheric simulations. For more complete documentation of WPS, see Chapter 3 of the WRF-ARW User’s Guide.

WPS consists of three utility programs: geogrid.exe, ungrirb.exe, and metgrid.exe. Each program is designed to take existing data sets and convert/interpolate them into an intermediate format. The build system for WPS is similar to that of WRF. NetCDF must be installed and the environment variable NETCDF should be set to the installation prefix. Run the configure script in the main WPS directory, pick a configuration option from the list, and then run compile. Note that WRF itself must be built prior to compiling WPS. In addition, the build process assumes that WRF exists in ../WRFV3/. WRF should be configured as described in Section 3 and compiled with the command

```
./compile em_real >& compile.log
```

The WPS can be configured from inside the top level directory wrf-fire/WPS with the command

```
```
and compiled in the same directory with the command

```bash
./compile >& compile.log
```

Upon successful completion the three binaries listed above should exist in the current directory.

Because the WPS programs are, for the most part, not processor intensive, it is not generally necessary to compile these programs for parallel execution, even if they do support it. Typical usage of WRF with real data involves doing all of the preprocessing work either locally on a workstation or on the head node of a supercomputer. The intermediate files are all architecture independent, so they can be transferred between computers safely. If you intend to use a supercomputer for the main simulation, it is advisable to generate the WPS output locally and transfer the met_em files to the computer you will be using for WRF-Fire. The met_em files are much smaller than the wrfinput and wrfbdy files and can be transported easily. This also eases the process of dealing with the dependencies of the python scripts described below because it may not be easy or even possible to meet these requirements on a shared parallel computer.

### Fire variables in namelist.wps

The simulation domain is described in the file `namelist.wps`. This namelist contains four sections, one for each of the main binaries created in WPS and one shared among them all. This file, as distributed with WRF-Fire, is set up for a test case useful for testing, but in general one needs to modify it for each simulation domain. The following table lists namelist options that can be modified. Other options in this file are generally not necessary to change for WRF-Fire simulations. See the WRF-ARW User’s Guide for more information.

<table>
<thead>
<tr>
<th>Variable names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;share</td>
<td>Shared name list options</td>
</tr>
<tr>
<td>max_dom</td>
<td>Number of nested domains to use</td>
</tr>
<tr>
<td>start_date/end_date</td>
<td>Starting/ending date and time to process atmospheric data in the format YYYY-MM-DD hh:mm:ss. These times should coincide with reanalysis cycles for your atmospheric data (hours 00,03,06,09,12, etc. for 3 hour NARR data). The simulation window in which you are interested in running must be inside this interval.</td>
</tr>
<tr>
<td>Subgrid_ratio_[xy]</td>
<td>The refinement ratio from the atmospheric grid to the fire grid.</td>
</tr>
<tr>
<td>interval_seconds</td>
<td>Number of seconds between each atmospheric dataset. (10800 for 3 hour NARR data)</td>
</tr>
</tbody>
</table>
Domain specifications

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent_id</td>
<td>When using nested grids, the parent of the current grid, or 0 if it is the highest level.</td>
</tr>
<tr>
<td>parent_grid_ratio</td>
<td>The refinement ratio from the parent grid (ignored for top level grid) (only 3 or 5 is supported by WRF)</td>
</tr>
<tr>
<td>[ij]_parent_start</td>
<td>The indices on the parent grid of the lower left corner of the current grid (ignored for top-level grid)</td>
</tr>
<tr>
<td>E_we/e_sn</td>
<td>The size of the grid in the x/y axis</td>
</tr>
<tr>
<td>dx/dy</td>
<td>Resolution of the grid in the x/y axis</td>
</tr>
<tr>
<td>map_proj, true_lat[12], stand_lon</td>
<td>Projection specifications. Lambert is typically used for central latitudes such as the continental US. For small domains, the projection used does not matter much.</td>
</tr>
<tr>
<td>ref_x/ref_y</td>
<td>Grid index of a reference point with known geographic location. Defaults to the center of the domain.</td>
</tr>
<tr>
<td>ref_lon/ref_lat</td>
<td>The location (longitude/latitude) of the reference point.</td>
</tr>
<tr>
<td>geog_data_path</td>
<td>Absolute or relative path to geogrid data released with WPS (<a href="http://www2.mmm.ucar.edu/wrf/src/wps_files/geog_v3.1.tar.gz">http://www2.mmm.ucar.edu/wrf/src/wps_files/geog_v3.1.tar.gz</a>)</td>
</tr>
</tbody>
</table>

**Geogrid**

The geogrid executable acts exclusively on static datasets (those that don’t change from day to day) such as surface elevation and land use. Because these datasets are static, they can be obtained as a single set of files from the main WPS distribution website in resolutions of 10 minutes, 2 minutes, and 30 seconds. The geogrid executable extracts from these global data sets what it needs for the current domain. While resolutions of this magnitude are acceptable for ordinary atmospheric simulations, these datasets are too coarse for a high-resolution fire simulation. In particular, a WRF-Fire simulation will require two additional data sets not present in the standard data.

**NFUEL_CAT**

The variable NFUEL_CAT contains Anderson 13 fuel category data. This data can be obtained for the US from the USGS seamless data access server at: http://landfire.cr.usgs.gov/viewer/. Using the zooming and panning controls, the user can select the desired region with LANDFIRE 13 Anderson Fire Behavior Fuel Models box selected. This will open a new window where the user can request the data in specific projections and data formats.

**ZSF**

The variable ZSF contains high resolution terrain height information similar to that in the HGT variable present in atmospheric simulations; however, the standard topographical data set is only available at a maximum resolution of 30 arc seconds (about 900 meters). For a simulation using the WRF-Fire routines, data resolution of at least 1/3 of an arc second is desirable to include the effect of local terrain slope on the rate of spread. Such a dataset is available for the US at http://seamless.usgs.gov/. This is another USGS seamless data access server similar to that of LANDFIRE. The desired dataset on this
server is listed under elevation and is called 1/3” NED.

**Conversion to geogrid format**

Once one has collected the necessary data from USGS servers or elsewhere, it is necessary to convert it from the given format (such as geotiff, Arcgrid, etc.) into geogrid format. The format specification of the geogrid format is given in the WPS section of the WRF users guide. The process of this conversion is somewhat technical; however, work is in progress to automate it.

**Editing GEOGRID.TBL**

In order to include your custom data into the WPS output, you must add a description of it in the GEOGRID.TBL file, which is located, by default, in the geogrid subdirectory of the main WPS distribution. In addition to the standard options described in the WPS users guide, there is one additional option that is necessary for defining data for fire grid variables. For them, there is a subgrid option, which is off by default. For fire grid data, one should add the option subgrid=yes to indicate that the variable should be defined on a refined subgrid with a refinement ratio defined by the subgrid_ratio_[xy] option in the WPS namelist. For example, typical table entries would appear as follows:

```plaintext
name=NFUEL_CAT
  priority=1
  dest_type=categorical
  dominant_only=NFUEL_CAT
  z_dim_name=fuel_cat
  halt_on_missing=yes
  interp_option=default:nearest_neighbor+average_16pt+search
    rel_path=default:landfire/
    subgrid=yes

name = ZSF
  priority = 1
  dest_type = continuous
  halt_on_missing=yes
  interp_option = default:four_pt
    rel_path=default:highres_elev/
    subgrid=yes
```

This table assumes that the converted data resides as a subdirectory of the standard data directory given in the namelist under the option geog_data_path. The NFUEL_CAT data should reside in landfire/ and ZSF in highres_elev/. In general, the only options that should be modified by the user are the rel_path or abs_path options. Once the data has been obtained and converted and the geogrid table has been properly set up, the user can run:

```
./geogrid.exe
```

which will create files such as geo_em.d01.nc that contain the interpolated static data fields.
Ungrib and Metgrid

The ungrib executable performs initial processing on atmospheric data. There are many different datasets that can be used as input to ungrib. One must obtain this data manually for a given simulation. Because fire simulations will be at a much higher resolution than most atmospheric simulations, it is advisable to get as high resolution data as possible. The 32 km resolution data from the North American Regional Reanalysis (NARR) is likely a good choice. This data is available freely from https://dss.ucar.edu/datazone/dsszone/ds608.0/NARR/3HRLY_TAR/. For real data WRF runs, three individual datasets from this website are required: 3d, flx, and sfc. To use them, download the files for the appropriate date/time and extract them somewhere on your file system. The files have the naming convention, NARR3D_200903_0103.tar. NARR indicates it comes from the NARR model, 3D indicates that it is the atmospheric data fields, and 200903_0103 indicates that it contains data from March 1\textsuperscript{st} through 3\textsuperscript{rd} of 2009. Once these files are extracted, they must be linked into the main WPS directory with the command link_grib.csh. It takes as arguments all of the files extracted from the dataset. For example, if you extracted these files to /home/joe/mydata, then you would issue the command:

```
./link_grib.csh /home/joe/mydata/*
```

into the top level WPS directory. Each atmospheric dataset requires a descriptor table known as a variable table to be present. WPS comes with several variable tables that work with most major data sources. These files reside in WPS/ungrib/Variable_Tables/. The appropriate file must be linked into the top level WPS directory as the file Vtable. For NARR data, type:

```
ln -sf ungrib/Variable_Tables/Vtable.NARR Vtable
```

Once this has been done, everything should be set up properly to run the ungrib command:

```
./ungrib.exe
```

Finally, the program metgrid combines the output of ungrib and geogrid to create a series of files, which can be read by WRF’s real.exe. This is accomplished by

```
./metgrid.exe
```

Assuming everything completed successfully, you should now have a number of files named something like met_em.d01.2009-03-01_12:00:00.nc. These should be copied or linked to your WRFV3/test/em_real directory. If any errors occur during execution of ungrib or metgrid, then make sure you have downloaded all of the necessary atmospheric data and that the variable table and namelist are configured properly.
Running real case and WRF-Fire

First copy or link the met_em files generated by metgrid into test/em_real. If the simulation is being done locally, this can be accomplished by running in \texttt{wrf-fire/WRFV3/test/em_real}\ln -sf ../..

The namelist for WRF in the file \texttt{namelist.input} must now be edited to reflect the domain configured in WPS. In addition to the fire-specific settings listed in Section 4.3 regarding the ideal simulation, a number of other settings must be considered as listed below. See Chapter 5 for more details on these settings.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;time_control</td>
<td></td>
</tr>
<tr>
<td>start_xxx/end_xxx</td>
<td>These describe the starting and ending date and time of the simulation. They must coincide with the start_date/end_date given in namelist.wps.</td>
</tr>
<tr>
<td>run_xxx</td>
<td>The amount of time to run the simulation.</td>
</tr>
<tr>
<td>interval_seconds</td>
<td>Must coincide with interval seconds from namelist.wps.</td>
</tr>
<tr>
<td>restart_interval</td>
<td>A restart file will be generated every x minutes. The simulation can begin from a restart file rather than wrfinput. This is controlled by the namelist variable ‘restart’.</td>
</tr>
<tr>
<td>&amp;domains</td>
<td>All grid settings must match those given in the geogrid section of namelist.wps.</td>
</tr>
<tr>
<td>num_metgrid_levels</td>
<td>The number of vertical levels of the atmospheric data being used. This can be determined from the met_em files: \texttt{nccdump -h met_em*</td>
</tr>
<tr>
<td>sr_x/sr_y</td>
<td>Fire grid refinement. This must match that given in namelist.wps as subgrid_ratio_x/subgrid_ratio_y.</td>
</tr>
<tr>
<td>p_top_requested</td>
<td>The default is 5000, but may need to be edited if there is an error executing real. If so, just set this to whatever it tells you in the error message.</td>
</tr>
</tbody>
</table>

Once the namelist is properly configured, run the real executable:

\texttt{./real.exe}\n
and then run wrf:

\texttt{./wrf.exe}
Fire state variables

A number of array variables were added to the registry to the WRF state in order to support the fire model. They are available in the wrfout* files created when running WRF. All fire array variables are based at the centers of the fire grid cells. Their values in the strips at the upper end of width sr_x in the x direction and sr_y in the y direction are unused and are set to zero by WRF.

The following variables can be used to interpret the fire model output.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFN</td>
<td>level set function. Node (i,j) is on fire if LFN(i,j) (\leq 0)</td>
</tr>
<tr>
<td>FXLONG, FXLAT</td>
<td>longitude and latitude of the nodes</td>
</tr>
<tr>
<td>FGRNHFX</td>
<td>ground heat flux from the fire (W/m(^2)), averaged over the cell</td>
</tr>
<tr>
<td>FGRNQFX</td>
<td>ground heat flux from the fire (W/m(^2)), averaged over the cell</td>
</tr>
<tr>
<td>ZSF</td>
<td>terrain elevation above sea level (m)</td>
</tr>
<tr>
<td>UF, VF</td>
<td>surface wind</td>
</tr>
<tr>
<td>FIRE_AREA</td>
<td>approximate part of the area of the cell that is on fire, between 0 and 1</td>
</tr>
</tbody>
</table>

WRF-Fire software

This section is intended for programmers who wish to modify or extend the fire module.

WRF-Fire coding conventions

The fire module resides in WRF physics layer and conforms to WRF Coding Conventions. The wildland fire-related subroutines maintain the conventions as they apply to on atmospheric grids, adapts them to 2D surface-based computations, and follows analogous conventions on the fire grid. In particular, these routines may not maintain any variables or arrays that persist between calls, and may not use common blocks, allocatable variables, or pointer variables. Work arrays with variable bounds may be declared only as automatic; thus, they are freed between on exit from the subroutine where they are declared. All grid-sized arrays that should persist between calls to the wildland fire-related subroutines must be created in WRF through the registry mechanism, and passed to these as arguments.

In addition, the wildland fire-related subroutines may not call any WRF routines directly but only through a utility layer. All variables in the wildland fire-related subroutines are based at grid centers. Grid dimensions are passed in argument lists as
ifds, ifde, jfds, jfde, & ! fire domain dims
ifms, ifme, jfms, jfme, & ! fire memory dims
ifps, ifpe, jfps, jfpe, & ! fire patch dims (may be omitted)
ifts, ifte, jfts, jfte, & ! fire tile dims

Atmosphere grid 2D variables are declared with dimension(ims:ime, jms:jme).
Fire grid variables are declared with dimension(ifms:ifme, jfms:jfme).
Loops on the fire grid are always over a tile. The index variable names, the order of the
loops, and the bounds are required exactly as in the code fragment below.

```
do j=jfts,jfte
   do i=ifts,ifte
      fire_variable(i,j)=...
   enddo
endo```

In loops that need to index more than one grid at the same time (such as computations on
a submesh, or interpolation between atmosphere and fire) the index variable names must
always begin with i j.

**Parallel execution**

In these routines, all computational subroutines are called from a thread that services a
single tile. There is no code running on a patch. Loops may update only array entries
within in the tile but they may read other array entries in adjacent tiles, for example for
interpolation or finite differences. The values of arrays that may be read between adjacent
tiles are synchronized outside of the computational routines. Consequently, the values of
a variable that was just updated may be used from an adjacent tile only in the next call to
the computational subroutines, after the required synchronization was done outside.
Synchronization within a patch is by exiting the OpenMP loop. Synchronization of the
values between patches is by explicit HALO calls on the required variables and with the
required width. HALOs are provided by the WRF infrastructure and specified in the
registry.

The overall structure of the parallelism is spread over multiple software layers,
subroutines and source files. The computation is organized in stages, controlled by the
value of ifun.

```
! the code executes on a single patch
! if distributed memory, we are one of the MPI processes
do ifun=ifun_start,ifun_end ! what to do
   if(ifun.eq.1)then  ! this HALO needed before stage ifun=1
      #include "SOME_HALO.inc" ! communicate between patches
   endif
...```
!$OMP PARALLEL DO  
do ij=1,num_tiles ! parallel loop over tiles  
   if(ifun.eq.1)then  ! one of the initialization stages  
      call some_atmosphere_to_fire_interpolation(...)  
   endif  
   ...
   call fire_model(...) ! call the actual model  
   ! for some values of ifun, fire_model may do nothing  
   if(ifun.eq.6)then  ! fire step done  
      call some_fire_to_atmosphere_computation(...)  
   endif  
endo ! end parallel loop over tiles  
   ! array variables are synchronized between tiles now  
endo ! end ifun loop

Software layers

The wildland fire-related subroutines are called from WRF file \texttt{dyn\_em/module\_first\_rk\_step\_part1}. The output of these routines (the heat and moisture tendencies) are stored on exit from these routines and added to the tendencies in WRF later in a call to \texttt{update\_phy\_ten} from \texttt{dyn\_em/module\_first\_rk\_step\_part2}. The wildland fire-related subroutines themselves consist of the following files in the \texttt{phys} directory, each constituting a distinct software layer:

\texttt{module\_fr\_fire\_driver.F} \textbf{Fire driver} layer. These subroutines are called directly from WRF. All parallelism is contained here. The rest of the routines are called on a single tile.

\texttt{module\_fr\_fire\_atm.F} \textbf{Atmosphere-fire interaction} layer. These routines are the interface between the fire and the atmosphere and interpolate between them.

\texttt{module\_fr\_fire\_model.F} \textbf{Fire front representation and advancement} layer. This routine calls the core and the physics layers. Formulated in terms of the fire grid only, it is intended to be independent of particular mathematical methods used in the core layer.

\texttt{module\_fr\_fire\_core.F} \textbf{Core} layer: This contains numerical algorithms for fire front advancement and the rate of fuel consumption calculation. It calls the physics layer for the fire spread rate.

\texttt{module\_fr\_fire\_phys.F} \textbf{Fire physics} layer. This contains algorithms for
calculating the rate of spread of the fire front in terms of the fire environment and associated initialization.

module_fr_fire_util.F Utilities layer. This layer is used by all other layers. It declares scalar switches and parameters and contains all interpolation and other service routines that may be general in nature and the interface to WRF routines such as messages and error exits. To maintain independence in WRF, this is the only layer that may call any WRF routines.

fr_fire_params_args.h This include file contains subroutine argument lists to pass through all arguments that are needed in the fire rate of spread algorithm in the physics layer. It is only necessary to write this long argument list once given the WRF requirement that arrays may be passed as arguments only, and not shared globally, say, as pointers. Also, it maintains the independence of the core layer from the physics layer and the modularity of the wildland fire-related subroutines in WRF.

fr_fire_params_decl.h Include file with the matching declarations.

Initialization in idealized case

The initialization of model arrays in the idealized case is done in the file dyn_em/module_initialize_fire.F

This file was adapted from other initialization files in the same directory and extended to deal with wildland fire-related variables.

a. Vertically stretched grid

Because of the fine meshes used in fire modeling, the user may wish to search for the text grid%znw(k) and modify the following loop to assure a desired refinement of the vertical atmospheric grid near the Earth surface:

\[
\text{DO k=1, kde}
\text{grid%znw(k) = (exp((-k-1)/float(kde-1)/z_scale) &}
\text{- exp(-1./z_scale))/(1.-exp(-1./z_scale))}
\text{ENDDO}
\]

b Topography

The relevant code is found by searching for the text

!***** set terrain height

The terrain height needs to be set consistently in the atmosphere model in the array
grid%ht and in the fire model array grid%zsf at the finer resolution. In the supplied examples, controlled by namelist.input variables fire_mountain_type, fire_mountain_start_x, fire_mountain_start_y, fire_mountain_end_x, fire_mountain_end_y, and fire_mountain_height, both arrays are set consistently from an algebraic formula (a cosine hill or a cosine ridge).

It is possible, though not recommended, to set only grid%ht and have the fire module interpolate the terrain height from the atmosphere mesh by specifying fire_topo_from_atm=1 in namelist.input. This will result in blocky terrain with discontinuous terrain gradients, which will affect fire spread patterns.

Note that in a real run, the user should leave fire_topo_from_atm=0 and both terrain height arrays are set consistently at the best available resolution from the WPS.

The user should not modify the code immediately after the setting of the terrain height arrays, which initializes a number of atmosphere variables consistently with the terrain height.