HWRF USERS’ GUIDE

February 2010

The Developmental Testbed Center

Contributors to this Guide:

Shaowu Bao, NOAA/ESRL/GSD
Richard Yablonsky, University of Rhode Island
Don Stark, NCAR/RAL/JNT
Ligia Bernardet, NOAA/ESRL/GSD

Please send questions to: wrfhelp@ucar.edu

Acknowledgments: The authors also wish to thank Carol Makowski of NCAR/RAL/JNT for providing edit support for this document and addressing a number of formatting issues.
# TABLE OF CONTENTS

## Chapter 1: HWRF System Introduction ................................................. 6

1.1 HWRF System Overview ........................................................................................................ 6
1.2 HWRF Source Code Directory Structure ................................................................. 8
1.3 Input Data ................................................................................................................................. 11
1.4 Production Directory ............................................................................................................ 12

## Chapter 2: Software Installation ......................................................... 14

2.1 Introduction ............................................................................................................................... 14
2.2 Obtaining the HWRF System Code ................................................................................... 14
2.3 Building the HWRF System ................................................................................................. 16
   2.3.1 Background ......................................................................................................................... 16
   2.3.2 Internal Libraries ............................................................................................................. 16
   2.3.3 Basic System Requirements ........................................................................................ 17
   2.3.4 Required System Libraries ........................................................................................ 17
2.4 Building WRF-NMM ................................................................................................................. 20
   2.4.1 Background ........................................................................................................................ 20
   2.4.2 Configuring WRF-NMM ................................................................................................ 20
   2.4.3 Compiling WRF-NMM .................................................................................................... 21
2.5 Building HWRF-Utilities ........................................................................................................... 23
   2.5.1 Set Environment Variables ........................................................................................ 23
   2.5.2 Configure and Compile .................................................................................................. 23
2.6 Building POM-TC ....................................................................................................................... 25
   2.6.1 Set Environment Variables ................................................................................... 25
   2.6.2 Configure and Compile .................................................................................................. 26
2.7 Building GFDL Vortex Tracker ............................................................................................. 27
   2.7.1 Set the Environment Variables ........................................................................... 27
   2.7.2 Configure and Compile .................................................................................................. 28
2.8 Building NCEP Coupler ........................................................................................................... 29
   2.8.1 Configure and Compile .................................................................................................. 29
2.9 Building WPS .............................................................................................................................. 30
   2.9.1 Background ......................................................................................................................... 30
   2.9.2 Configure and Compile .................................................................................................. 31
4.4.8 hwrf_anl_bogus_10m.exe................................................................. 59
4.4.9 hwrf_create_trak_guess.exe....................................................... 60
4.4.10 hwrf_inter_4to6.exe ................................................................. 60
4.4.11 hwrf_anl_cs_10m.exe ............................................................... 61
4.4.12 hwrf_merge_nest_4x_10m2.exe ............................................... 61
4.4.13 hwrf_merge_nest_4x_step1.exe ............................................. 62
4.4.14 hwrf_inter_2to1.exe ............................................................... 63
4.4.15 hwrf_merge_nest_4x_step2.exe ............................................. 63

Chapter 5: Ocean Initialization of POM-TC ......................... 64

5.1 Introduction .................................................................................. 64
5.2 Scripts .......................................................................................... 64
5.3 Procedures in the four “kickit” Scripts ....................................... 65
  5.3.1 kickit01_sharpn.sh ................................................................. 65
  5.3.2 kickit02_getsst.sh ................................................................. 65
  5.3.3 kickit03_phase3.sh ............................................................... 65
  5.3.4 kickit04_phase4.sh ............................................................... 66
5.4 Executables .................................................................................. 66
  5.4.1 gfdl_date2day.exe ................................................................. 66
  5.4.2 gfdl_day2date.exe ................................................................. 67
  5.4.3 gfdl_getsst.exe .................................................................... 67
  5.4.4 gfdl_sharp_mcs_rf_l2m_rmy5.exe ........................................ 67
  5.4.5 gfdl_ocean_united.exe ........................................................ 68

Chapter 6: How to Run HWRF .................................................. 70

6.1 Introduction .................................................................................. 70
6.2 How to Run HWRF Using the Script wrf.ksh ........................... 70
  6.2.1 Overview of the Script ........................................................ 71
  6.3 Output Files in the Directory ..................................................... 73
  6.4 Status Check .............................................................................. 73
6.5 Executables .................................................................................. 73
  6.5.1 wrf.exe .................................................................................... 73
  6.5.2 hwrf_pwmc.exe .................................................................... 74
  6.5.3 hwrf_ocean_united.exe ........................................................ 74
  6.5.4 hwrf_swcorner_dynamic.exe ............................................. 75
Chapter 7: HWRF Post Processor ................................................. 76
  7.1 Introduction ............................................................................................................................... 76
  7.2 Script ........................................................................................................................................... 76
  7.3 Overview of the Script ............................................................................................................. 77
  7.4 Executables .................................................................................................................................. 78
    7.4.1 wrfpost.exe .......................................................................................................................... 78
    7.4.2 copygb.exe ........................................................................................................................... 79
    7.4.3 hwrf_combine.exe ............................................................................................................ 79

Chapter 8: GFDL Vortex Tracker .................................................. 81
  8.1 Introduction .................................................................................................................................. 81
  8.2 How to Run the GFDL Vortex Tracker Using Script ..................................................................... 81
  8.3 Overview of the script ................................................................................................................ 82
  8.4 Executables ................................................................................................................................... 82
    8.4.1 hwrf_gettrk.exe ................................................................................................................ 82
  8.5 How to Plot the Tracker Output Using ATCF_PLOT ..................................................................... 87
Chapter 1: HWRF System Introduction

1.1 HWRF System Overview

The Weather Research and Forecast system for hurricane prediction (HWRF) was implemented at the NWS/NCEP’s Environmental Modeling Center (EMC) to address the Nation’s next generation hurricane forecast problems.

The HWRF model is a primitive equation non-hydrostatic coupled atmosphere-ocean model with the atmospheric component formulated with 42 levels in vertical. The model uses a rotated-E dynamic core with a parent and nest domain coded in the WRF framework. The model domain spans about 80° in the latitudinal direction and extends about 95° in longitudinal direction in the north and about 70° in the south depending on the selected center position of the parent domain. The boundary of the domain is determined from the initial position of the storm and NHC (National Hurricane Center) forecast 72-h position, if available. The nest domain of about 6x6° moves along with the storm and the nesting is two-way interactive. The stationary parent domain has a grid spacing of 0.18° (about 27 km) while the inner nest domain 0.06° (about 9 km). The time steps for HWRF are 54 and 18 s respectively for the parent and nest domains.

The model physics is based primarily on similar physics as the GFDL hurricane model which includes a simplified Arakawa-Schubert scheme for cumulus parameterization and Ferrier cloud microphysics package for large-scale condensation. The vertical diffusion scheme is based on Troen and Mahrt’s non-local scheme. The Monin-Obukhov scheme is used for surface flux calculations with an improved air-sea momentum flux parameterization in strong wind conditions and a one layer slab land model. Radiation effects are evaluated by the GFDL scheme, which includes diurnal variations and interactive effects of clouds. HWRF physics include parameterizations of dissipative heating.

The NCEP GFS (Global Forecast System) global analysis and the storm message provided by NHC are used to generate initial conditions for the hurricane model. The HWRF systems contains a forecast/analysis cycle in which a operational 6-h HWRF forecast provides a first guess to a 3DVAR data assimilation system for observational data. The first guess field is relocated and modified so that the initial storm position, structure and intensity conforms to that estimated from data in the NHC storm message. The initial conditions are calculated by adding the assimilated storm structure back onto the GFS environmental analysis fields. The GFS forecasted fields every 6 hours are used to provide lateral boundary analysis conditions during each forecast.

1The 3DVAR system is not included in the HWRF current release and not covered in this user’s guide.
The time integration is performed with a forward-backward scheme for fast waves, an implicit scheme for vertically propagating sound waves and the Adams-Bashforth scheme for horizontal advection and for the Coriolis force. In the vertical, the hybrid pressure-sigma coordinate (Arakawa and Lamb 1977) is used. Horizontal diffusion in based on a 2nd order Smagorinsky-type following Janjic.

The hurricane model is coupled with a version of the Princeton Ocean Model (POM-TC). In the Atlantic, the POM-TC is configured with 1/6° horizontal grid spacing and 23 vertical sigma levels. The POM-TC is initialized by a diagnostic and prognostic spinup of the ocean circulations using available climatological ocean data in combination with real-time sea surface temperature and sea surface height data. During the ocean spinup, realistic representations of the structure and positions of the Loop Current, Gulf Stream, and warm- and cold-core eddies are incorporated.

HWRF is suitable for use in tropical applications including real-time NWP, forecast research, physics parameterization research, air-sea coupling research and teaching. The HWRF system supported by the community by the Developmental Testbed Center (DTC) includes the following three main components:

- **HWRF atmospheric components**

  The general WRF code repository will be used for the development and support of the HWRF system. The atmospheric component of HWRF will be a configuration of the general WRF model.

  The HWRF atmospheric component includes the following:
  - WRF-NMM V3.2 (which has tropical physics schemes and a vortex following moving nest)
  - WRF Preprocessing System (WPS)
  - Vortex Initialization
  - HWRF Post-processing
  - GFDL vortex tracker

- **HWRF oceanic components**

  HWRF is an atmosphere-ocean coupled system. POM was developed at Princeton University. At University of Rhode Island (URI), POM model was coupled to the GFDL and HWRF models. The URI version of POM (POM-TC) has been enhanced with advanced capabilities such as feature based data assimilation techniques, in which the loop current, warm/cold eddy core and gulf stream are well represented in the ocean initial conditions. POM-TC is currently the oceanic component of the HWRF system.
The HWRF oceanic component includes the following:

- POM-TC model
- Ocean initialization
- Atmosphere-Ocean Coupler

The atmospheric and oceanic components are interactively coupled with a MPI-based coupler, which was developed at NOAA/NWS/NCEP/EMC. The atmospheric and oceanic components exchange information through the coupler: the ocean sends the sea surface temperature (SST) to the atmosphere; the atmosphere receives the SST and sends the surface fluxes, including sensible heat flux, latent heat flux and short-wave radiation to the ocean, and so on. The frequency of information exchange can be specified by the user.

- Future HWRF development

HWRF is being actively developed and advanced. In the future, more components will be coupled into the HWRF system, including wave, hydrology, storm surge and inundation components.

The HWRF modeling system software is in the public domain and is freely available for community use.

### 1.2 HWRF Source Code Directory Structure

HWRF system source code consists of the following seven parts:

1. HWRF-utilities (including vortex initialization)
2. POMTC Ocean Model
3. NCEP Atmosphere-Ocean Coupler
4. GFDL Vortex Tracker
5. WRF Atmospheric Model
6. WRF Preprocessing System (WPS)
7. WRF Post Processing (WPP)

The code of the components can be obtained by downloading the following tar files from the DTC website (see Chapter 2):

1. `HWRF-UTILITIES.tar.gz`
2. `POMTC.tar.gz`
3. `NCEP-COUPLER.tar.gz`
4. GFDL-VOXETRACKER.tar.gz
5. WRFV3.tar.gz
6. WPSV3.tar.gz
7. WPP.tar.gz

After copying these tar files to a user-defined HWRF top directory and expanding them, the user should see the following directories:

hwrf-utilities
pomtc
ncep-coupler
gfdl-vortextracker
WRFV3
WPSV3
WPPV3

For the remainder of this document, it will be assumed that the tarfiles have been expanded under \$\{HOME\}/HWRF/src

The directory trees are listed below for these seven components:

1. hwrf-utilities (HWRF utility programs and scripts)

   | | arch (compile options)
   | | exec (executables)
   | | libs (libraries including blas, sp, sfcio, bacio and bufr)
   | | parm (WPS namelist; three WRF namelists; hwrf_eta_micro_lookup.dat; all WRF lookup tables from run subdirectory, some are modified for HWRF; and WPP namelist)
   | | post (source code for the HWRF-related post processing programs)
   | | | hwrf_combine
   | | | hwrf_atcf_to_stats
   | | | hwrf_read_indi
   | | | hwrf_gridgenfine
   | | | hwrf_aux_rw
   | | | hwrf_post
   | | | hwrf_swath
   | scripts (scripts used to run HWRF system)
   | | vortex_init (source code for vortex initialization)
   | | | hwrf_anl_bogus
   | | | hwrf_create_trak_guess
   | | | hwrf_set_ijstart
   | | | hwrf_split
   | | | hwrf_create_nest
   | | | hwrf_anl_cs
2. **pomtc (Ocean model)**

- _arch_ (compile options)
- _ocean_parm_ (namelists for ocean model)
- _ocean_init_ (source code for generating ocean model initial condition)
  - _united_
  - _getsst_
  - _sharp_mcs_rf_12m_rmy5_
  - _date2day_
  - _day2date_
- _ocean_exec_ (ocean model executables)
- _ocean_main_ (source code for the ocean forecast model)
  - _ocean_united_
- _ocean_scripts_ (scripts used to generating ocean model initial condition)

3. **ncep-coupler (NCEP Coupler)**

- _arch_ (compile options)
- _cpl_exec_ (coupler executables)
- _hwrf_pwmc_ (source code for the coupler)

4. **gfdl-vortextrack (GFDL Vortex tracker)**

- _arch_ (compile options)
- _trk_exec_ (GFDL vortex tracker executables)
- _hwrf_gettrk_ (GFDL vortex tracker source codes)

5. **WRFV3 (Atmospheric model)**

- _Registry_ (WRFV3 Registry files)
- _arch_ (compile options)
- _chem_ (WRF-Chem, not used in HWRF)
- _dyn_em_ (WRF-ARW dynamic modules, not used in HWRF)
- _dyn_exp_ ('toy' dynamic core, not used by HWRF)
- _dyn_nmm_ (WRF-NMM dynamic modules, used by HWRF)
- _external_ (that contains external packages including ocean coupler interface)
1.3 Input Data

Users will need the datasets below as input to HWRF components. These datasets can be obtained from the DTC website. In order to use the DTC-supported scripts for running HWRF, these datasets must be stored following the directory structure below in a disk accessible by the HWRF scripts.

1. **Tcvitals (TC vitals data)**

2. **fix**
• *geo_static* (for WPS geogrid.exe)
• *ocean* (for ocean initialization)

```
gfdl_gdem.[00-13].ascii
fyl
gfdl_ocean_readu.dat.[01-12]
gfdl_ocean_spinup_gdem3.dat.[01-12]
gfdl_ocean_spinup_gspath.[01-12]
gfdl_ocean_spinup.BAYuf
fyl
gfdl_ocean_spinup.FSgsuf
fyl
gfdl_ocean_spinup.SGYREuf
fyl
gfdl_ocean_topo_and_mask.united
fyl
gfdl_ocean_spinup_gspath.[01-12]
```

3. **GFS_gridded** (GFS gridded data, for WRF initialization)

*GFS_gridded* has subdirectories `${YYYYMMDDHH}`, which in turn, contains GFS gridded data `gfs.${YYYYMMDDHH}.pgrbf${hhh}` where `${YYYYMMDDHH}` is the initial time and `${hhh}` is the forecast hour. For example, `gfs.2008082800.pgrbf24` is a GFS 24 hour forecast whose initial time is August 28 00Z 2008.

4. **GFS_spectral** (GFS spectral data, for ocean initialization)

*GFS_spectral* has subdirectories `${YYYYMMDDHH}`, which in turn, contains GFS spectral data `gfs.${YYYYMMDDHH}.t${hh}z.sanl` and `gfs.${YYYYMMDDHH}.t${hh}z.sfcanl`, where `${YYYYMMDDHH}` is the initial time and `${hh}` is the forecast hour. For example, `gfs.2008082800.t00z.sanl` and `gfs.2008082800.t00z.sfcanl` are initial-time GFS spectral data in a GFS forecast whose initial time is August 28 00Z 2008.

5. **Loop_current** (loop current data for ocean initialization)

`hwrf_gfdl_loop_current_wc_ring_rmy5.dat.${YYYYMMDD}` and `hwrf_gfdl_loop_current_rmy5.dat.${YYYYMMDD}`, where `${YYYYMMDD}` is the date. For example, `hwrf_gfdl_loop_current_rmy5.dat.20080828` and `hwrf_gfdl_loop_current_wc_ring_rmy5.dat.20080828` describe the loop current and warm-core ring information for August 28 2008.

### 1.4 Production Directory

If a user uses the scripts included in the released tar files to run the HWRF system, the following production directories will be created and used:

The top production directory is: `${USER}/HWRF/SID/yyyyymmddhh` (where SID is storm ID, e.g., 07L, and yyyyymmddhh is the forecast initial time).
The directory ${USER}/HWRF/SID/yyyyymmddhh has the following sub-directories:

1. messages (created by hwrfdomain.ksh)
2. geoprd (created by eogrid.ksh)
3. ungribprd (created by ungrib.ksh)
4. metgridprd (created by metgrid.ksh)
5. realprd (created by real.ksh)
6. wrfghostprd (created by wrf.ksh run in “ghost” mode)
7. wrfanalysisprd (created by wrf.ksh run in “analysis” mode)
8. relocateprd (created by vortex initialization scripts)
9. oceanprd (ocean initialization)

   • o sharpn (run the sharpening program)
   • o getsst (extract the SST from GFS)
   • o phase3 (run Phase 3)
   • o phase4 (run Phase 4)
Chapter 2: Software Installation

2.1 Introduction

The NOAA operational WRF hurricane system (HWRF) consists of seven components:

- WRF Atmospheric Model
- WRF Preprocessing System (WPS)
- WRF Post Processing (WPP)
- HWRF utilities
- POM-TC Ocean Model
- GFDL Vortex Tracker
- NCEP Atmosphere-Ocean Coupler

The first three are the traditional WRF components, while the remaining four components provide the extra functionality needed for the hurricane system.

The DTC has built a utility to build and install the HWRF components and its libraries on multiple platforms. This chapter discusses how to build each of the seven components that make up the HWRF system. Section 2.2 discusses how to obtain the source code. Section 2.3 outlines the general system requirements for building the HWRF components. Section 2.4 addresses the WRF-NMM build for HWRF. Sections 2.5, 2.6, 2.7, and 2.8 discuss building the HWRF utilities, the POM-TC ocean model, the GFDL Vortex Tracker, and the NCEP coupler, respectively. The remaining two sections cover building the WPS and WPP components.

2.2 Obtaining the HWRF System Code

The HWRF system consists of seven components. The first three are WRF components and consist of the WRF model, the WRF preprocessor (WPS), and the WRF post processor (WPP). The HWRF capability is new in WRF, and is only available in version 3.2+ of the WRF components. All three of these components are available from the DTC WRF-NMM user web site. The remaining four components, the HWRF utilities, POM-TC, GFDL Vortex Tracker, and the NCEP coupler, can be obtained from the WRF for Hurricanes web site. In all cases, new users must first register before downloading the source code. Returning users will only need to provide their registered email.

The WRF, WPS, and WPP source code and documentation can be obtained from


Following registration, a successful download will produce three gzip'ed tar files, WRFV3.tar.gz and WPSV3.tar.gz and WPPV3.tar.gz. The exact file name may
differ slightly from what is shown here, since the code is under constant development. It is strongly recommended that the user always obtain the latest version of the code, unless there is a specific reason to use an older version.

The remaining HWRF components; the HWRF Utilities, the POM-TC model, the NCEP coupler, and the GFDL vortex tracker, can be obtained from

http://www.dtcenter.org/HurrWRF/users

A successful download will produce the four gzip'ed tar files, POMTC.tar.gz, HWRF-UTILITIES.tar.gz, GFDL-VORTEXTRACKER.tar.gz, and NCEP-COUPLER.tar.gz.

In order to use the DTC-provided scripts for running HWRF, it is recommended that codes be built in the directory $(HOME)/HWRF/src/. Before unpacking the tar files, create the working directory and change into it.

```
mkdir $(HOME)/HWRF/src/
cd $(HOME)/HWRF/src/
```

The tar files can be unpacked by use of the UNIX commands:

```
gunzip *.tar.gz
tar -xvf *.tar
```

or on LINUX, with the single command:

```
tar -zxvf *.tar.gz
```

Once unpacked, there should be the seven source directories:

- WRFV3 – the weather research and forecast model
- WPSV3 – the WRF preprocessor
- WPPV3 – the WRF postprocessor
- hwrf-utilities – the vortex initializer and hwrf utilities and tools
- pomtc – the tropical cyclone version of the pom ocean model
- gfdl-vortextracker – the vortex tracker for storm initialization
- ncep-coupler – the ocean/atmosphere coupler

We will now discuss how to build each component.
2.3 Building the HWRF System

2.3.1 Background

Running the HWRF system requires building seven components. Three of the components, WPS, WPP, and the HWRF-Utilities, require linking to the WRF I/O API libraries for the build. Since these I/O libraries are built as part of the WRF build, the WRF components must be built first. Once WRF is built, WPS, WPP, or the HWRF-Utilities can be built next. Since the HWRF-Utilities build produces support libraries needed by the POM-TC and the GFDL Vortex Tracker builds, the HWRF utilities must be built before these other two components. The remaining component, the NCEP Coupler can be built independently of the other components.

The component dependency is as follows:
- WRF
  - WPS
  - WPP
  - HWRF utilities
    - POM-TC
    - GFDL vortex tracker
- NCEP Coupler

2.3.2 Internal Libraries

The HWRF-Utilities directory includes five external libraries in the \texttt{libs/src/} directory, which are needed to build the two components, POM-TC and the GFDL vortex tracker. These internal libraries are provided to guarantee that the libraries will function optimally with these two components. Other versions of these libraries, such as those provided with the WPP source code, should not be used when building the POM-TC and the GFDL vortex tracker.

The included libraries are:
- BACIO
- BLAS
- SFCIO
- SP
- W3

The POM-TC component always requires the three libraries SFCIO, SP and W3. The additional library BLAS is required when the ESSL library is not included with the compiler installation. Typically this is for any system other than the IBM. The GFDL vortex tracker component requires the two libraries BACIO and W3.
2.3.3 Basic System Requirements

While the majority of the HWRF source code is written in FORTRAN, the HWRF system relies on ancillary programs written in Perl, shell script, and C. These tools, along with gnuMake, are required for the build. In addition, the HWRF I/O API requires netCDF, so that library should be available on the system.

For a basic build of the HWRF system, the following system tools/libraries are required:

- FORTRAN 90/95 compiler
- C compiler
- MPI V1+
- Perl
- netCDF V3.6+

In addition to the system tools listed above, the HWRF system also requires the wgrib program to manipulate and decode GRIB files. The source code can be obtained from NOAA at the web site:

http://www.cpc.noaa.gov/products/wesley/wgrib.html

The HWRF system has been successfully built and run on two types of computer hardware:

- IBM with xlf Fortran compiler
- Linux (both 32 and 64-bit) with
  - PGI pgf90 (v9 – v11)
  - Intel ifort (v7 - v9)

Future plans include Linux support for the GNU gfortran compiler.

2.3.4 Required System Libraries

The HWRF system requires numerous support libraries to build. Many of these libraries are part of the compiler installation, and are subsequently referred to as system libraries. For our needs, the most important of these libraries are MPI and netCDF.

The netCDF V3.6+ library is required for the HWRF system I/O. It can be downloaded from the Unidata web site:

http://www.unidata.ucar.edu

Typically, the netCDF library is installed in a directory that is included in the users path such as /usr/local/lib. When this is not the case, the environment variable NETCDF, can be set to point to the location of the library. For csh/tcsh, the path can be set with the command:
setenv NETCDF /path_to_netcdf_library/

For bash/ksh, the path can be set with the command:

export NETCDF=/path_to_netcdf_library/

It is crucial that libraries, such as netCDF, are built with the same FORTRAN compiler, and with the same compiler flags, as will be used to compile the source code. This is often an issue on systems with multiple FORTRAN compilers, or when the option to build with multiple word sizes (e.g. 32-bit vs. 64-bit addressing) is available.

For example, a 64-bit Linux system with the two FORTRAN compilers; PGI, and Intel, might need to have netCDF built four ways:

- Intel ifort with 32-bit addressing
- Intel ifort with 64-bit addressing
- PGI pgf90 with 32-bit addressing
- PGI pgf90 with 64-bit addressing

The correct choice of library depends on which compiler combination will be used to build the source code.

As another example, consider NCAR's IBM supercomputers. Typically, both the 32-bit and 64-bit versions of the netCDF library will be installed. Traditionally, compilation would select the 32-bit version of the library by default. Under these conditions, it was necessary to specify in the compilation utility the 64-bit version by setting the environment variable (for csh/tcsh)

setenv OBJECT_MODE 64

or (for bash/ksh)

export OBJECT_MODE=64

prior to compilation. System defaults may vary, ask your system administrator for details.

Running the HWRF system as a distributed memory parallel job requires that a version of the MPI library be installed. Just as with the netCDF library, the MPI library must be built with the same FORTRAN compiler and word size option as the remainder of the source code.

Installing MPI on a system is typically a job for the system administrator. To determine whether MPI is available on your computer system, try the following UNIX commands:
- `which mpif90`
- `which mpicc`
- `which mpirun`

If all of these executables are defined, MPI is probably already installed. All that remains is to confirm that the MPI `bin/`, `include/`, and base directories are included in the user's path.

To see the complete environmental settings, type `env` at the prompt. Scroll through the entries and confirm that both the MPI and netCDF directories are included in the `PATH` variable.

If your path variable is not set up as discussed above, the shell configuration files may need to be modified. Modification of your path variables can easily lead to damaging your user account, so proceed with caution. If you are unfamiliar with modifying your dot files, ask your system administrator for help before making any changes.

The path names for the compilers and libraries listed above should be defined in shell configuration files (such as `.cshrc` or `.login`). For example, the path for a Linux system, running `csh/tcsh`, with a PGI compiler installed may look something like this:

```
set path = (
    /usr/pgi/bin /usr/pgi/lib /usr/local/ncarg/bin \
    /usr/local/mpich-pgi /usr/local/mpich-pgi/bin \
    /usr/local/netcdf-pgi/bin /usr/local/netcdf-pgi/include)
```

In addition, the following environmental variables may need to be set:

```
setenv PGI /usr/pgi
setenv NETCDF /usr/local/netcdf-pgi
setenv LD_LIBRARY_PATH /usr/lib:/usr/local/lib:/usr/pgi/linux86/lib:\
                       /usr/local/netcdf-pgi/lib
```

Your system administrator can assist with questions concerning the environment setup.
2.4 Building WRF-NMM

2.4.1 Background

The WRF code has a fairly sophisticated build mechanism. The package attempts to
determine the architecture on which the code is being built, and then presents the user
with options to allow the user to select the preferred build method. For example, on a
Linux machine, the build mechanism determines whether the machine is 32-bit or 64-bit,
and then prompts the user for the desired usage of processors (such as serial, shared
memory, or distributed memory), and presents a selection of possible compiler choices.

A helpful guide to building WRF using PGI compilers on a 32-bit or 64-bit LINUX
system can be found at:

http://www.pgroup.com/resources/tips.htm

2.4.2 Configuring WRF-NMM

To configure WRF-NMM, go to the top of the WRF directory (`cd
$(HOME)/HWRF/src/WRFV3`) and type:

    ./configure

You will be given a list of choices for your computer. These choices range from
compiling for a single processor job (serial), to using OpenMP shared-memory (SM) or
distributed-memory (DM) parallelization options for multiple processors.

The choices for IBM architectures are as follows:

1. AIX xlf compiler with xlc (serial)
2. AIX xlf compiler with xlc (smpar)
3. AIX xlf compiler with xlc (dmpar)
4. AIX xlf compiler with xlc (dm+sm)

For the HWRF system on IBM architectures, option 3 is recommended.

The choices for Linux architectures are as follows:

1. Linux x86_64, PGI compiler with gcc  (serial)
2. Linux x86_64, PGI compiler with gcc  (smpar)
3. Linux x86_64, PGI compiler with gcc  (dmpar)
4. Linux x86_64, PGI compiler with gcc  (dm+sm)
5. Linux x86_64 i486 i586 i686, ifort compiler with icc  (serial)
6. Linux x86_64 i486 i586 i686, ifort compiler with icc (smpar)
7. Linux x86_64 i486 i586 i686, ifort compiler with icc (dmpar)
8. Linux x86_64 i486 i586 i686, ifort compiler with icc (dm+sm)
9. Linux i486 i586 i686 x86_64, PathScale compiler with pathcc (serial)
10. Linux i486 i586 i686 x86_64, PathScale compiler with pathcc (dmpar)
11. x86_64 Linux, gfortran compiler with gcc (serial)
12. x86_64 Linux, gfortran compiler with gcc (smpar)
13. x86_64 Linux, gfortran compiler with gcc (dmpar)
14. x86_64 Linux, gfortran compiler with gcc (dm+sm)
15. Cray XT Catamount/Linux x86_64, PGI compiler with gcc (serial)
16. Cray XT Catamount/Linux x86_64, PGI compiler with gcc (dmpar)
17. Cray XT CNL/Linux x86_64, PGI compiler with gcc (serial)
18. Cray XT CNL/Linux x86_64, PGI compiler with gcc (smpar)
19. Cray XT CNL/Linux x86_64, PGI compiler with gcc (dmpar)
20. Cray XT CNL/Linux x86_64, PGI compiler with gcc (dm+sm)
21. Cray XT CNL/Linux x86_64, Pathscale compiler with gcc (serial)
22. Cray XT CNL/Linux x86_64, Pathscale compiler with gcc (smpar)
23. Cray XT CNL/Linux x86_64, Pathscale compiler with gcc (dmpar)
24. Cray XT CNL/Linux x86_64, Pathscale compiler with gcc (dm+sm)

At this time, only the distributed memory (dmpar) builds are recommended for the HWRF system on Linux systems. Thus for Linux operating systems, only options 3 or 7 are recommended.

Once a build option is selected, the type of nesting is chosen next:

*Compile for nesting? (1=basic, 2=preset moves, 3=vortex following) [default 1]:*

For HWRF, only option 2 is supported. This would appear counter intuitive, since option 3 is listed as the vortex following option. Option 3 is only appropriate for the ARW hurricane configuration.

The configure step is now completed. A file has been created called `configure.wrf`. The compile options and paths in the `configure.wrf` file can be edited for further customization of the build process.

### 2.4.3 Compiling WRF-NMM

To correctly compile WRF-NMM for the HWRF system, the following environment variables must be set:

In C-Shell use the commands:

```bash
setenv HWRF 1
setenv WRF_NMM_CORE 1
setenv WRF_NMM_NEST 1
```
In Bash Shell use the commands:
```bash
export HWRF=1
export WRF_NMM_CORE=1
export WRF_NMM_NEST=1
```

Note that setting the environment variable `WRF_NMM_NEST` to 1 does not preclude running with a single domain.

Once these environment variables are set, enter the following command, for csh/tcsh,
```bash
./compile nmm_real |& tee wrf-build.log
```

to initiate the compile and save the standard out and standard error to a log file.

Be aware that the commands `./compile -h` and `./compile` produce a listing of all of the available compile options (only the `nmm_real` option is relevant to the HWRF system).

To remove all object files (except those in `external/`), type:
```bash
./clean
```

To conduct a complete clean which removes all built files in all directories, as well as the `configure.wrf`, type:
```bash
./clean -a
```

A complete clean is strongly recommended if the compilation failed, the registry has been changed, or if the configuration file is changed.

A successful compilation produces two executables in the directory `main/`:

- `real_nmm.exe`: WRF initialization
- `wrf.exe`: WRF model integration

These executables are linked to the `run/` and `test/nmm_real/` directories, which can be used for model runs.

Further details on the HWRF atmospheric model, physics options, and running the model can be found in the Running HWRF chapter of the user’s guide. The HWRF input data must be created using the WPS code (see the WPS chapter of the HWRF user’s guide for more information).

Complete details on building and running the WRF-NMM model, can be downloaded from the WRF-NMM user’s guide, which is available at


2.5 Building HWRF-Utilities

The HWRF utilities directory consists of an eclectic collection of source code and libraries. The libraries, which are provided to support the POM-TC and vortex tracker builds, include the BACIO, BLAS, BUFR, SFCIO, SP, and W3 libraries. In addition the source code includes source for the vortex initialization routines, software tools such as the grbindex, and post processing utilities.

2.5.1 Set Environment Variables

The HWRF utilities build requires that two path variables $\text{NETCDF}$ and $\text{WRF\_DIR}$ be set to the appropriate paths. The netCDF library path $\text{NETCDF}$ is required for building the WRF-NMM component, therefore it should be appropriately set if that component compiled successfully. The $\text{WRF\_DIR}$ path should point to the WRF directory compiled in the previous section. If the WRF-NMM model has not yet been built, you must first build WRF before compiling any of the other components.

In csh/tcsh, the variables are set as follows:

```
setenv NETCDF /absolute_path_to_appropriate_netCDF_library/
setenv WRF_DIR /absolute_path_to_the_WRF_directory/WRFV3
```

In bash/ksh, the variables are set as follows:

```
export NETCDF=/absolute_path_to_appropriate_netCDF_library/
export WRF_DIR=/absolute_path_to_the_WRF_directory/WRFV3
```

It is crucial that the Fortran compiler used to build the libraries (Intel, PGI, XLF, etc.) be the same as the compiler used to compile the source code. Typically, this is only an issue in two situations; on Linux systems having multiple compilers installed, and on systems where there is a choice between building with either 32-bit or 64-bit addressing.

2.5.2 Configure and Compile

To configure HWRF-Utilities for compilation, from within the hwrf-utilities directory, type:

```
./configure
```

The configure script checks the system hardware, and if the path variables are not set, asks for the correct paths to the netCDF libraries and the WRF build. It concludes by asking the user to choose a configuration supported by current machine architecture.

For Linux the options are:
1. Linux x86_64, PGI compiler (dmpar)
2. Linux x86_64, Intel compiler (dmpar)

For the IBM, only one choice is available:

AIX xlff compiler with xlC

The configure script creates a file called configure.hwrf. This file contains compilation options, rules, and paths specific to the current machine architecture, and can be edited to change compilation options, if desired.

At this time, HWRF has only been tested on the IBM, Linux PGI (v7, v8, v9) and Linux Intel (9.0, 10.0, 11.0). Additional build platforms and updates will be released as they become available.

In csh/tcsh, to compile the HWRF utilities and save the build output to a log file, type:

`./compile |& tee compile.log`

To get help about compilation, type:

`./compile -h`

To remove all object files, type:

`./clean`

To conduct a complete clean which removes ALL built files, including the executables, libraries, and the configure.hwrf, type:

`./clean -a`

A complete clean is strongly recommended if the compilation failed or if the configuration file is changed.

If the compilation is successful, it will create twenty two executables in the directory exec/:

`diffwrf_3dvar.exe`
`grbindx.exe`
`hwrf_anl_4x_step2.exe`
`hwrf_anl_bogus_10m.exe`
`hwrf_anl_cs_10m.exe`
`hwrf_atcf_to_stats.exe`
`hwrf_combine.exe`
`hwrf_create_nest_1x_10m.exe`
and eight libraries in the directory *libs/*.

*libbacio.a* - BACIO library
*libblas.a* - BLAS library
*libbufr_i4r8.a* - BUFR library built with -i4 -r8 flags
*libsfcio_i4r4.a* - SFCIO library built with -i4 -r4 flags
*libsp_i4r8.a* - SP library built with -i4 -r8 flags
*libsp_i4r4.a* - SP library built with -i4 -r4 flags
*libw3_i4r8.a* - W3 library built with -i4 -r8 flags
*libw3_i4r4.a* - W3 library built with -i4 -r4 flags

These libraries will be needed by the GFDL vortex tracker and the POM-TC ocean model. The configuration step for these components will require setting a path variable to point to the *hwrf-utilities/libs/* directory in the HWRF utilities directory.

## 2.6 Building POM-TC

### 2.6.1 Set Environment Variables

The Tropical Cyclone version of the Princeton Ocean Model (POM-TC), requires three external libraries; SFCIO, SP, W3, and a fourth library which is platform dependent. The first three of these libraries are located in the *hwrf-utilities/libs/* directory and should be available if the HWRF utilities component is built successfully. If the HWRF utilities have not yet been built, you must first build them before trying to build POM-TC.

In csh/tcsh, the first three library paths can be set with the commands:

```
setenv LIB_W3_PATH /absolute_path_to_hwrf-utilities/libs/
setenv LIB_SP_PATH /absolute_path_to_hwrf-utilities/libs/
```
In bash/ksh, the first three library paths can be set with the commands:

```bash
export LIB_W3_PATH=/absolute_path_to_hwrf-utilities/libs/
export LIB_SP_PATH=/absolute_path_to_hwrf-utilities/libs/
export LIB_SFCIO_PATH=/absolute_path_to_hwrf-utilities/libs/
```

In addition to the three previous libraries, POM-TC requires routines from the ESSL mathematical library. Typically, these libraries are only available on IBM machines. As an alternative to the ESSL library, the build system includes the option to build the BLAS mathematical library. When the ESSL library is not provided as part of the system software, such as on Linux platforms, the BLAS library may be used in place of it. In such a case, the fourth and final path is set to:

```bash
setenv LIB_BLAS_PATH /absolute_path_to_hwrf-utilities/libs/
```

for csh/tcsh. For bash/ksh the path is set by:

```bash
export LIB_BLAS_PATH=/absolute_path_to_hwrf-utilities/libs/
```

### 2.6.2 Configure and Compile

To configure POM-TC for compilation, from within the pomtc directory, type:

```
./configure
```

The configure script checks the system hardware, and if the path variables are not set, asks for software paths to the **W3**, **SP**, **SFCIO**, and for Linux, the **BLAS** libraries. It concludes by asking the user to choose a configuration supported by current machine architecture.

For Linux, the options are:

1. Linux x86_64, PGI compiler (dmpar)
2. Linux x86_64, Intel compiler (dmpar)

For the IBM, only one choice is available:

```
AIX xlf compiler with xlc
```

The configure script creates a file called `configure.pom`. This file contains compilation options, rules, and paths specific to the current machine architecture, and can be edited to change compilation options, if desired.
Builds on IBM architectures with xlf, Linux with PGI (v7, v8, v9) and Linux with Intel (9.0, 10.0, 11.0) are well tested. Additional platforms will be added as they become available.

In csh/tcsh, to compile the POM-TC and save the build output to a log file, type:

```
./compile |& tee ocean.log
```

To get help about compilation, type:

```
./compile -h
```

To remove all the object files, type:

```
./clean
```

To conduct a complete clean which removes ALL built files, object, executables and the configuration file `configure.pom`, type:

```
./clean -a
```

A complete clean is strongly recommended if the compilation failed to build, or if the configuration file is changed.

If the compilation is successful, six executables are created in `ocean_exec/`,

- `gfdd_date2day.exe`
- `gfdd_day2date.exe`
- `gfdd_getsst.exe`
- `gfdd_ocean_united.exe`
- `gfdd_sharp_mcs_rf_l2m_rmy5.exe`
- `hwrf_ocean_united.exe`

The first of these, `hwrf_ocean_united.exe`, is the actual ocean model executable. The remaining five executables are used for the ocean initialization.

## 2.7 Building GFDL Vortex Tracker

### 2.7.1 Set the Environment Variables

The GFDL Vortex Tracker, requires two external libraries; W3, and BACIO. These libraries are located in the `hwrf-utility/libs/` directory and should be available if the HWRF utilities are successfully built. If the HWRF utilities have not yet been built, you must first build them before trying to build the vortex tracker.
In csh/tcsh, the library paths to be set are:

```bash
setenv LIB_W3_PATH /absolute_path_to_hwrf-utilities/libs/
setenv LIB_BACIO_PATH /absolute_path_to_hwrf-utilities/libs/
```

In bash/ksh, the library paths to be set are:

```bash
export LIB_W3_PATH=/absolute_path_to_hwrf-utilities/libs/
export LIB_BACIO_PATH=/absolute_path_to_hwrf-utilities/libs/
```

### 2.7.2 Configure and Compile

To configure the vortex tracker for compilation, from within the tracker directory, type:

```bash
./configure
```

The configure script checks the system hardware, and if the path variables are not set, asks for software paths to the W3, and BACIO libraries. It concludes by asking the user to choose a configuration supported by current machine architecture.

For Linux, the options are:

1. Linux x86_64, PGI compiler (dmpar)
2. Linux x86_64, Intel compiler (dmpar)

For the IBM, only one choice is available:

*AIX xl compiler with xlc*

The configure script creates a file called `configure.trk`. This file contains compilation options, rules, and paths specific to the current machine architecture. The configure file can be edited to change compilation options, if desired.

Three build configurations are well tested; IBM xl, Linux with PGI (v7, v8, v9), and Linux with Intel (9.0, 10.0, 11.0). Additional platforms will be released as they become available.

In csh/tcsh, to compile the vortex tracker and save the build output to a log file, type:

```bash
./compile |& tee tracker.log
```

To get help about compilation, type:
To remove all object files, type:

`./compile -h`

To completely clean ALL built files, object, executable, and `configure.trk`, type:

`./clean`

To completely clean ALL built files, object, executable, and `configure.trk`, type:

`./clean -a`

A complete clean is strongly recommended if the compilation failed, or if the configuration file is changed.

If the compilation was successful, the executable `hwrf_gettrak.exe` will be created in the directory `trk_exec/`.

### 2.8 Building NCEP Coupler

#### 2.8.1 Configure and Compile

To configure the NCEP Coupler for compilation, from within the Coupler directory, type:

`./configure`

The configure script checks the system hardware, asks the user to choose a configuration supported by current machine architecture, and creates a configure file called `configure.cpl`

For Linux, the options are:

1. Linux x86_64, PGI compiler (dmpar)
2. Linux x86_64, Intel compiler (dmpar)

For the IBM, only one choice is available:

 `AIX xlf compiler with xlc`

The configure file `configure.cpl` contains compilation options, rules, and paths specific to the current machine architecture, and can be edited to change compilation options, if desired. At this time, the IBM, Linux PGI (v7, v8, v9) and Linux Intel (9.0, 10.0, 11.0) are well tested. Additions and updates for other platforms will be released as they become available.
In csh/tcsh, to compile the coupler and save the build output to a log file, type:

```
./compile |& tee coupler.log
```

To get help about compilation, type:

```
./compile -h
```

To remove all the object files, type:

```
./clean
```

To completely clean ALL built files, object, executable, and `configure.cpl`, type:

```
./clean -a
```

A complete clean is strongly recommended if the compilation failed, or if the configuration file is changed.

If the compilation is successful, it will create the single executable `hwrf_pwmc.exe` in the `cpl_exec/` directory.

### 2.9 Building WPS

#### 2.9.1 Background

The WRF WPS requires the same build environment as the WRF-NMM model, including netCDF library and MPI libraries. Since the WPS makes direct calls to the WRF I/O API libraries included with the WRF-NMM model, the WRF-NMM model must be built prior to building the WPS. In order to run the WRF Domain Wizard, an optional tool to assist in creating simulation domains, Java 1.5 or later is needed.

Further details on using the WPS to create HWRF input data can be found in Chapter 3 of the HWRF user’s guide.

Complete details on building and running the WPS and the Domain Wizard, are available from the WRF-NMM user’s guide, and can be downloaded from:

`http://www.dtcenter.org/wrf-nmm/users/docs/overview.php`
2.9.2 Configure and Compile

Following the compilation of the WRF-NMM executables, change to the WPS directory and issue the configure command.

`.configure`

Choose one of the configure options listed.

For example, on IBM computers the listed options are:

1. AIX serial, NO GRIB2
2. AIX serial
3. AIX DM parallel, NO GRIB2
4. AIX DM parallel

Choose any of the options. On Linux computers, the listed options are:

1. PC Linux x86_64, Intel compiler serial, NO GRIB2
2. PC Linux x86_64, Intel compiler serial
3. PC Linux x86_64, Intel compiler DM parallel, NO GRIB2
4. PC Linux x86_64, Intel compiler DM parallel
5. PC Linux x86_64 (IA64 and Opteron), PGI compiler 5.2 or higher, serial, NO GRIB2
6. PC Linux x86_64 (IA64 and Opteron), PGI compiler 5.2 or higher, serial
7. Cray XT Linux x86_64 (IA64 and Opteron), PGI compiler 5.2 or higher, DM parallel, NO GRIB2
8. PC Linux x86_64 (IA64 and Opteron), PGI compiler 5.2 or higher, DM parallel, NO GRIB2
9. PC Linux x86_64 (IA64 and Opteron), PGI compiler 5.2 or higher, DM parallel
10. PC Linux x86_64 (IA64 and Opteron), PathScale compiler 2.1 or higher, serial, NO GRIB2
11. PC Linux x86_64 (IA64 and Opteron), PathScale compiler 2.1 or higher, DM parallel, NO GRIB2
12. PC Linux x86_64, g95 compiler, serial, NO GRIB2
13. PC Linux x86_64, g95 compiler, serial
14. PC Linux x86_64, g95 compiler, DM PARALLEL, NO GRIB2
15. PC Linux x86_64, g95 compiler, DM PARALLEL

Any of these options are acceptable. After selecting the proper option:

`.compile |& tee wps.log`

To conduct a complete clean which removes ALL built files in ALL directories, as well as the `configure.wps`, type:

`.clean -a`
A complete clean is strongly recommended if the compilation failed, the registry has been changed, or if the configuration file is changed.

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`, `metgrid.exe`, if the WPS software was successfully installed. If any of these links do not exist, check the compilation log file to determine what went wrong.

For full details on the operation of WPS, see the WPS chapter of the HWRF user’s guide.

### 2.10 Building WPP

The NCEP WRF Post-processor was designed to interpolate HWRF output from native coordinates and variables to coordinates and variables more useful for analysis. Specifically WPP de-staggers the HWRF output, interpolates the data from its native vertical grid to standard levels, and creates additional diagnostic variables.

#### 2.10.1 WPP Software Requirements

The WRF Post-processor requires the same Fortran and C compilers used to build the WRF model. In addition to the netCDF library, the WRF I/O API libraries, which are included in the WRF model tar file, are also required.

The WPP build requires a number of support libraries (IP, SP, W3). These are provided with the source code and are located in the `WPPV3/lib/` directory. These libraries are for the WPP build only. They should not be confused with the libraries of the same name located in the `hwrf-utilities/libs` directory.

#### 2.10.2 Set Environment Variables

The WPP requires the WRF-NMM I/O API libraries to successfully build. If the WRF-NMM model has not yet been built, it must first be built before compiling WPP.

The WPP build requires linking to the WRF-NMM I/O API libraries. For the build system to find these libraries, one of two conditions must be met. Either the WRF_NMM directory must be called `WRFV3` and it must be located at the same directory level as the WPP directory, or the path variable `WRF_DIR` must be set to the location of the WRFV3 root directory.
In addition to setting the path variable, building WPP for use with HWRF requires setting the environmental flag \texttt{HWRF}. This is the same flag set when building WRF-NMM for HWRF.

To set up the environment for WPP, the environment variables can be set by typing (for csh/tcsh):

\begin{verbatim}
setenv HWRF 1
setenv WRF\_DIR /absolute\_path\_to\_the\_WRF\_directory/WRFV3/
\end{verbatim}

For bash/ksh, the environment variables can be set by typing:

\begin{verbatim}
export HWRF=1
export WRF\_DIR=/absolute\_path\_to\_the\_WRF\_directory/WRFV3/
\end{verbatim}

2.10.3 Configure and Compile

WPP uses a build mechanism similar to that used by the WRF model. Type configure

\texttt{./configure}

to generate the WPP configure file. The configure script will ask for the \texttt{WRF\_DIR} path if the WRF directory is not located where expected and the path has not already been set. You will then be given a list of configuration choices tailored to your computer. For example; for IBM machines there is one option,

\texttt{AIX xlf compiler with xlc (serial)}

while for LINUX operating systems there are three,

\texttt{LINUX i486 i586 i686, PGI compiler (serial)}
\texttt{LINUX i486 i586 i686, Intel compiler (serial)}
\texttt{LINUX i486 i586 i686, gfortran compiler (serial)}

Choose one of the configure options listed. The configuration script will generate the configure file \texttt{configure.wpp}. If necessary, the \texttt{configure.wpp} file can be modified to change the default compile options and paths.

To compile WPP, enter the command:

\begin{verbatim}
./compile |& tee wpp.log
\end{verbatim}

This command should create four WPP libraries in \texttt{lib/} (\texttt{libmpi.a, libsp.a, libip.a,} and \texttt{libw3.a}), and three WPP executables in \texttt{exec/} (\texttt{wrfpost.exe, ndate.exe,} and \texttt{copygb.exe}).
Once again, these libraries are for the WPP only, and should not be used by the other components.

To remove all built files, as well as the `configure.wpp`, type:

```
./clean
```

This is recommended if the compilation failed.

For full details on the operation of WPP, see the WPP chapter of the HWRF user’s guide, and for complete details on building and running the WPP, see the WRF-NMM user guide, which can be downloaded at:

`http://www.dtcenter.org/wrf-nmm/users/docs/overview.php`
Chapter 3: HWRF Preprocessing System

3.1 Introduction

The WRF WPS is a set of three programs whose collective role is to prepare input to real_nmm program for real data simulations. For general information about working with WPS, view the WRF-NMM documentation at http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/V3/users_guide_nmm_chap1-7.pdf

In the operational HWRF, input data from the GFS is processed through the WPS and real_nmm programs, and then submitted to a vortex relocation procedure described in Chapter 4.

3.2 How to Run the HWRF Preprocessing Using Scripts

Four scripts are used to preprocess data for HWRF. hwrfdomain.ksh defines the parent domain grid; geogrid.ksh interpolates static geographical data to the grid defined by hwrfdomain.ksh; ungrib.ksh extracts meteorological fields from GRIB formatted files and writes the fields to intermediate files; and metgrid.ksh horizontally interpolates the meteorological fields extracted by ungrib.ksh to the model domain grid defined by hwrfdomain.ksh. The scripts can be found in /${HOME}/HWRF/src/hwrf-utilities/scripts

3.2.1 hwrfdomain.ksh

Before running hwrfdomain.ksh, edit the script to set the variables:

SID: storm ID
(for example, 07L for the 7th storm in Atlantic Basin)

TCVITALS: the path to the TC vitals files
(for example, /ptmp/HurrTutorial/datasets/Tcvitals)

NAMELIST: wps namelist
(for example, ${HOME}/HWRF/src/hwrf-utilities/parm/hwrf_namelist.wps)

DOMAIN_HOME: base path of the output directory.
(for example, /ptmp/${USER}/HWRF/${SID}/${YYYYMMDDHH}).

Note that the initialization time, ${YYYYMMDDHH}, is extracted from ${NAMELIST}
Then, run the script using the command *hwrfdomain.ksh*

**Overview of script hwrfdomain.ksh:**

1. Get the initialization time from the WPS namelist (*${NAMELIST}*).
2. Enter the working directory, which is *${DOMAIN_HOME}/messages*.
3. A TCvitals record for the storm must be found for the HWRF system to run. Try to locate the TC vitals record for this storm that has 72-h guidance. If it is not available, look for a record without the 72-h guidance.
4. Get the storm center latitude and longitude from the TC vitals record.
5. Compute the reference latitude and longitude for the HWRF domain using the storm center.
6. Test to make sure that the reference longitude is no more than 5 degrees away from storm center longitude.
7. Output the storm center to file *storm.center*.
8. Output the center of the domain to file *domain.center*.

**Output files in directory *${DOMAIN_HOME}/messages***

2. *domain.center*: file that contains domain reference center latitude and longitude.

**Status Check**

If the two output files are found in the directory of *${DOMAIN_HOME}/messages*, the script *hwrfdomain.ksh* has finished successfully.

**HWRF Domain Wizard**

The WRF Domain Wizard now has the capability of setting up the HWRF domain. For more information about the WRF Domain Wizard, view

*http://www.wrfportal.org/DomainWizard.html*
3.2.2 *geogrid.ksh*

Before running *geogrid.ksh*, edit the script to set up the variables:

- **WPS_ROOT**: path to the WPS installation.
  (for example, \${HOME}/HWRF/src/WPSV3)
- **SID**: storm ID
  (for example, 07L for the 7th storm in Atlantic Basin)
- **NAMELIST**: WPS namelist.
  (for example, \${HOME}/HWRF/src/hwrf-utilities/parm/hwrf_namelist.wps)
- **DOMAIN_HOME**: base path of the output directory.
  (for example, 
  /ptmp/\${USER}/HWRF/\${SID}/\${YYYYMMDDHH}. Note that the 
  initialization time \${YYYYMMDDHH}, is extracted from 
  \${NAMELIST}).

Then edit the namelist:

\${HOME}/HWRF/src/hwrf-utilities/parm/hwrf_namelist.wps to define geog_data_path, 
the path to the geo_static dataset

After editing the script and the namelist, run the script using the command 
geogrid.ksh

**Overview of script geogrid.ksh**

1. Get the initialization time from the WPS namelist (\${NAMELIST})
2. Enter the working directory, which is \${DOMAIN_HOME}/geoprd
3. Get the domain center latitude and longitude from file 
   messages/domain.center and insert the center latitude and longitude and the 
   path to the WPS installation into namelis t.wps to replace the place-holders in 
   the that namelist
4. Link to the GEOGRID.TBL file
5. Run geogrid.exe

**Output files in directory in \${DOMAIN_HOME}/geogrid**

1. *geo_nmm.d01.nc*: static geographical data for the parent domain, with a grid 
   spacing of 0.18 degrees.
2. *geo_nmm_nest.l01.nc*: static geographical data that covers the parent domain, 
   with a grid spacing of 0.06 degrees.
Status check
If “Successful completion of program geogrid.exe” is found in the standard output file, 
${DOMAIN_HOME}/geoprd/geogrid.log, the script geogrid.ksh has successfully finished.

3.2.3 ungrib.ksh

Before running ungrib.ksh, edit the script to set up the variables:

- **WPS_ROOT**: path to the WPS installation.
  (for example, 
  `${HOME}/HWRF/src/WPSV3)
- **SOURCE**: type of the GRIB file that will be used to initialize HWRF (use GFS)
- **SID**: storm ID  (for example, 07L for the 7th storm in Atlantic Basin)
- **NAMELIST**: WPS namelist.  
  (for example, 
  `${HOME}/HWRF/src/hwrf-
  utilities/parm/hwrf_namelist.wps)
- **DOMAIN_HOME**: base path of the output directory.  
  (for example, 
  `/ptmp/${USER}/HWRF/${SID}/$YYYYMMDDHH`).
- **SOURCE_PATH**: path to the source (GFS) GRIB date files.  
  (for example, 
  `/ptmp/HurrTutorial/datasets/GFS_gridded/$YYYYMMDDHH`).

Note that the initialization, time ${YYYYMMDDHH}, is extracted from ${NAMELIST}).

After setting up the variables, run the script using the command ungrib.ksh

Overview of script ungrib.ksh
1. Get the initialization time from the WPS namelist ( ${NAMELIST})
2. Enter the working directory, which is ${DOMAIN_HOME}/ungribprd
3. Link the Vtable into the work directory
4. Make links to the input data files
5. Copy the WPS namelist to the working directory
6. Run ungrib.exe

Output files in directory in ${DOMAIN_HOME}/ungribprd
The intermediate files written by ungrib.exe will have names of the form FILE:YYYY-MM-DD_HH (unless the prefix variable in ${NAMELIST} was set to a prefix other than ‘FILE’).
Status check

- If “Successful completion of program ungrig.exe” is found in the standard output file, 
  `${DOMAIN_HOME}/ungribprd/ungrib.log`, the script ungrig.ksh was successfully finished.
- The intermediate files written by ungrig.exe will have names of the form 
  `FILE:YYYY-MM-DD_HH` (unless the prefix variable in `${NAMELIST}` was set to a prefix other than 'FILE').

3.2.4 metgrid.ksh

Before running script metgrid.ksh, edit the script to set up the variables:

- `WPS_ROOT`: path to the WPS installation.
  (for example, `${HOME}/HWRF/src/WPSV3`)
- `SOURCE`: type of the GRIB file that will be used to initialize HWRF (use GFS)
- `SID`: storm ID
  (for example, 07L for the 7th storm in Atlantic Basin)
- `NAMELIST`: WPS namelist
  (for example, `${HOME}/HWRF/src/hwrf-utilities/parm/hwrf_namelist.wps`)
- `DOMAIN_HOME`: base path of the output directory.
  (for example, `/ptmp/${USER}/HWRF/${SID}/${YYYYMMDDHH}`).

Note that the initialization, time `${YYYYMMDDHH}`, is extracted from `{NAMELIST}`).

After setting up the environment variables, run the script using command metgrid.ksh

**Overview of script metgrid.ksh**

1. Get the initialization time from the WPS namelist (`${NAMELIST}`)
2. Enter the working directory, which is `${DOMAIN_HOME}/metgridprd`
3. Link files `METGRID.TBL`, `geo_nmm.d01.nc` and `geo_nmm_nest.l01.nc` to working directory
4. Insert path to `WPS METGRID.TBL` file into namelist.wps
5. Run `metgrid.exe`

**Output in the directory**

1. `${DOMAIN_HOME}/metgridprd`
2. `met_nmm.d01.YYYY-MM-DD_HH:mm:ss.nc` in the case of HWRF domains.
   Here, `YYYY-MM-DD_HH:mm:ss` refers to the date of the interpolated data in each file.
Status Check
If “Successful completion of program metgrid.exe” is found in the standard output file, ${DOMAIN_HOME}/metgridprd/metgrid.log, the script metgrid.ksh was successfully finished.

3.3 Executables

3.3.1 geogrid.exe

FUNCTION:
interpolates static geographical data to the grid.

INPUT:
(geographical static data directory:
(for example, /tmp/HurrTutorial/datasets/fix/geo_static)
GEOGRID.TBL
WPS namelist

OUTPUT:
geo_nmm.d01.nc: static geographical data for the parent domain, with a grid spacing of 0.18 degrees.
geo_nmm_nest.l01.nc: static geographical data that covers the parent domain, with a grid spacing of 0.06 degrees.

USAGE:
${WPS_ROOT}/geogrid.exe

3.3.2 ungrib.exe

FUNCTION:
extracts meteorological fields from GRIB formatted files and writes the fields to intermediate files.

INPUT:
GFS GRIB files
Vtable
WPS namelist
OUTPUT:
The intermediate files written by ungrib.exe will have names of the form
FILE:YYYY-MM-DD_HH (unless the prefix variable was set to a prefix other than
'FILE' in ${NAMELIST}).

USAGE:
${WPS_ROOT}/ungrib.exe

3.3.3 metgrid.exe

FUNCTION:
horizontally interpolates the meteorological fields extracted by ungrib.ksh to the
model parent grid.

INPUT:
METGRID.TBL
geo_nmm.d01.nc
WPS namelist
intermediate files produced by ungrib.exe

OUTPUT:
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.

USAGE:
${WPS_ROOT}/metgrid.exe

3.4 Algorithm to Define HWRF Domain Using the
Storm Center Location

In order to define the domain configuration for HWRF, ref_lat and ref_lon in the
“geogrid” namelist record are calculated according to the location of the observed storm
to be simulated. Script hwrfdomain.ksh reads the TC vitals records and calculate the
storm center location. NHC and JTWC are the two agencies that provide the Tcvital - a
one line text message that contains information on storm name, id, time, location,
intensity and 72-hr forecast position (if available) apart from many other parameters
used to describe the storm.
In the first step, the storm center at the initial time (STORM_LAT and STORM_LON is read from the TC vitals file). If 72-h forecast position is available, LATF72 and LONF72 are also read in. The domain center is treated differently for latitude and longitude.

a) For domain center latitude (CENLA):
   if STORM_LAT < 15.0 then CENLA=15.0
   if STORM_LAT > 25.0 then CENLA=25.0
   if STORM_LAT $\geq$ 35.0 then CENLA=30.0
   if STORM_LAT $\geq$ 40.0 then CENLA=35.0
   if STORM_LAT $\geq$ 45.0 then CENLA=40.0
   if STORM_LAT $\geq$ 50.0 then CENLA=45.0
   if STORM_LAT $\geq$ 55.0 then CENLA=50.0

b) For domain center longitude (CENLO):
   The domain center longitude is the average of storm center (STORM_LON) and the 72-h forecast longitude (LONF72). In the absence of 72-h forecast, 20 degrees are added to STORM_LON to create LONF72.
   CENT = (STORM_LON + LONF72)
   CENTAVG = CENT / 2
   CENLO = -CENTAVG / 10
   To assure that the domain center is separated from storm center by at least 5 degrees, the following procedure is followed:
   SLON=STORM_LON + 5
   if CENLO $\geq$ SLON then CENLO=SLON
   SLON=STORM_LON - 5
   if CENLO $\leq$ SLON then CENLO=SLON

Finally, the values of CENLA and CENLO are written to the namelist.wps as ref_lat and ref_lon.
Chapter 4: Vortex Initialization

4.1 Overview

The initial vortex is often not realistically represented in a high-resolution mesoscale model when the initial conditions are generated from a low-resolution global model, such as GFS. To address this issue, HWRF employs a sophisticated algorithm to initialize the hurricane vortex.

The HWRF vortex initialization process has three stages. First a check is performed to see if a previous 6-hr forecast exists. If so, this is a cycled run, otherwise it is a "cold start". A cycled run will go through all the three stages, while a "cold start" run will go through only stage 2 and 3:

1. Stage 1: Since the operational HWRF forecasts are run in cycles, a previous cycle 6-hr HWRF forecast is separated into environment fields and a storm vortex.

2. Stage 2: The initial condition generated from GFS data using WPS is separated into environment fields and a storm vortex.

3. Stage 3: The storm vortex from the previous 6-hr forecast (for cycled runs) or from a bogus vortex (for cold start runs) is adjusted in its intensity and structure to match the current time observed hurricane center location, intensity and structure information. The new vortex is added to the environment fields obtained from the GFS data to form the new initial condition that is used in HWRF forecast.

When a previous cycle 6-hr forecast vortex is not available (for example, the first cycle of a storm), an axi-symmetric bogus vortex is used and adjusted. The new vortex is added to the GFS environment fields in the observed location to replace the GFS vortex. The merged GFS environment fields and the new vortex are used to initialize both the outer and inner nested domains. The executables, scripts and procedures are described below.

4.2 Domain Grids Used in Vortex Initialization

Four domain grids are used in HWRF vortex initialization process (Fig. 4.1).

1. Model outer domain grid: 216x432 with a resolution of 0.18 deg. Covers about 80x80 degrees.

2. Model inner nest domain grid: 61x100 with a resolution of 0.06 degrees. Covers about 6x6 degrees.
3. 4x domain grid: 420x820 with a resolution of 0.06 degrees. Covers about 40x40 degrees.
This intermediate 4x domain is used to remove the GFS vortex, extract and correct the storm vortex from the previous 6-hr forecast (for cycled runs) or correct the bogus vortex (for cold start runs). The domain is large enough so that the GFS vortex is completely filtered out and a complete storm vortex from the previous 6-hr forecast is extracted, yet small enough to save computing resources. This domain is not used during the model integration.

4. Ghost domain grid: 210x409 with a resolution of 0.06 degrees covers about 20x20 degrees.
The ghost domain is mainly used for GSI data analysis. Although the use of GSI with HWRF is not currently supported by DTC, this domain is retained because it is an integral part of the initialization procedure. This domain is not used during the model integration.

Figure 4.1: The four domains used in HWRF vortex initialization.
4.3 How to Run the Vortex Initialization Using Scripts

The HWRF vortex initialization scripts come in the tarfile HWRF-UTILITIES.tar.gz and, following the procedures outlined in Chapters 1 and 2, will be expanded in the directory ${HOME}/src/hwrf-utilities/scripts

4.3.1 real.ksh

Edit the script to set:

- **WRF_ROOT**: path to WRFV3
- **SID**: storm id
- **NAMELIST**: path to namelist_main.input
- **DOMAIN_HOME**: base path to the output directory

Then run real.ksh. This will read in the output from the WPS executable metgrid.exe and generate wrfinput_d01 and wrfbdy_d01. namelist_main.input is the namelist used by real_nmm.exe and wrf.exe to do the hurricane forecasts (see Fig. 4.2. HWRF initialization procedures before vortex adjustments).

**Overview of script real.ksh:**
Run real_nmm.exe to generate initial and boundary conditions. A high-resolution sea-mask data for the entire outer domain is also generated to be used by the coupler.

**Output files in directory ${DOMAIN_HOME}/realprd:**
- wrfinput_d01 (initial condition)
- wrfbdy_d01 (boundary condition)
- fort.65 (high-resolution sea mask data)

**Status Check**
This step was successfully finished if the user finds “SUCCESS COMPLETE REAL” in files rsl.*

4.3.2 wrf.ksh

Edit the script to set:

- **WRF_ROOT**: path to WRFV3
- **SID**: storm id
- **WRF_MODE**: choose 'analysis' in this step
- **NAMELIST**: path to namelist_analysis.input
- **DOMAIN_HOME**: base path to the output directory
Then run `wrf.ksh`. This will make a one-minute run of `wrf.exe` and generate an analysis output for the inner nest domain. `namelist_analysis.input` is used by `wrf.exe` to do a one-minute run to obtain the initial time “analysis” file which contains the fields on the inner nest domain (see Fig. 4.2).

**Overview of script `wrf.ksh`**

1. Run `hwrf_swcorner_dynamic.exe` to calculate the istart and jstart values for the nest in the `namelist_analysis.input`. In `namelist_analysis.input` the model is set to run for only 1 minute and "analysis" option is set to "false" for both outer and inner domain.
2. Run `wrf.exe` using `namelist_analysis.input`.

Output files in the directory `$DOMAIN_HOME/wrf${WRF_MODE}prd`

"wrfanl_d02*", an "analysis" file for the HWRF inner domain.

**Status Check**

This step was successfully finished if the user finds “ SUCCESS COMPLETE WRF” in files `rsl.*`

### 4.3.3 `wrf.ksh`

Edit the script to set:

```
WRF_ROOT :   path to WRFV3
SID:         storm id
WRF_MODE:    choose 'ghost' in this step
NAMELIST:    path to `namelist_ghost.input`
DOMAIN_HOME : base path to the output directory
```

Then run `wrf.ksh`. This will make a one-minute run of `wrf.exe` and generate an analysis output for a 'ghost' nest domain. The ghost domain has the same center location and resolution as the inner nest domain, has 211x410 grid points and spans roughly 20x20 degrees. It is used only in vortex initialization. `Namelist_ghost.input` is the namelist used by `wrf.exe` to do a one-minute run to obtain the initial time “ghost” file which contains the initialization on the ghost domain (see Fig. 4.2).

**Overview of script `wrf.ksh`**

1. Run `hwrf_swcorner_dynamic.exe` to calculate the istart and jstart values for the nest in the `namelist_ghost.input`. In `namelist_ghost.input`, the model is set to run for only 1 minute and the "analysis" option is set to "false" for both outer and inner domain. Also the "e_we" and "e_sn" options for the nested domain are set to 211 and 410, respectively.
2. Run `wrf.exe` using `namelist_ghost.input`.

46
Output files in the directory 

```
${DOMAIN_HOME}/wrf${WRF_MODE}prd

"ghost_d02*", an "analysis" file for the HWRF ghost domain.
```

**Status Check**

This step was successfully finished if the user finds “SUCCESS COMPLETE WRF” in files `rsl`.

### 4.3.4. *relocate stage1.ksh*

Edit the script to set the environment variables:

```
WRF_ROOT: path to WRFV3
SID: storm id
NAMELIST: path to namelist_analysis.input
DOMAIN_HOME: base path to the output directory
CYCLE_HOME: path to the output directory containing previous 6hr forecast for cycled run
```

Then run `relocate stage1.ksh`. If a previous 6-h forecast exists (a cycled run), the forecast will be interpolated onto a 4x domain and separated into environment fields and storm vortex fields. The storm vortex fields will be adjusted. The 4x domain is about 40x40 deg with the resolution of the inner nest domain. The 4x domain is centered based on the NHC storm message data (see Fig. 4.3).

**Overview of script relocate stage1.ksh**

1. Run `diffwrf_3dvar.exe` to convert existing `wrfout_d01` and `wrfout_d02` into unformatted data files `old_hwrf_d01` and `old_hwrf_d02` respectively.
2. Run `merge_nest_4x_step1.exe` to merge `wrfout_d01` and `wrfout_d02` onto 4x domain and produce a file containing the merged data: `data_4x_hwrf`. The 4x domain has the same resolution as the inner nest domain.
3. Convert the time units in the previous cycle 6-h forecast output track from "minute" to "hour" and copy the track into user's work directory and rename the track file to "hdas_atcfunix".
4. Run `hwrf_create_trak_guess.exe` to produce a guess track (0,3,6,9 hour) for the current forecast.
5. Run `wrf_split.exe` to separate `data_4x_hwrf` into two parts: an environment field (`wrf_env`) and a storm vortex (`storm_pert`). A storm radius data file (`storm_radius`) is also generated.
6. Run `hwrf_pert_ct.exe` to do adjustments to `storm_pert`. The new storm vortex data (`storm_pert_new`) as well as two files containing the storm size information (`storm_size_p`) and the symmetric part of the vortex (`storm_sym`) are generated.
Output files in directory `${DOMAIN_HOME}/relocateprd`:
- `storm_size_p` (storm size information)
- `storm_pert_new` (new storm vortex after adjustments by `hwrf_pert_ct.exe`)
- `storm_sym` (symmetric part of the vortex)
- `storm_radius` (storm radius information)
- `wrf_env` (environment field)

Status Check:
If “failed” is not found in the standard output (stdout) and the files listed above exist, the script `relocate_stage1.ksh` run was successful.

### 4.3.4 `relocate_stage2.ksh`

Edit the script to set:

```
WRF_ROOT:    path to WRFV3  
SID:         storm id      
NAMELIST:    path to `namelist_analysis.input` 
DOMAIN_HOME: base path to the output directory
```

Then run `relocate_stage2.ksh`. This will merge the outer, inner nest and ghost nest domain initial fields onto a 4x domain grid. The merged fields will be separated into environment fields and storm vortex (see Fig. 4.4).

**Overview of script `relocate_stage2.ksh`**

1. Run `diffwrf_3dvar.exe` to convert `wrfinput_d01`, `wrfanl_d02` and `wrfghost_d02` into unformatted files `new_gfs_d01`, `new_gfs_d02` and `new_ght_d02`, respectively.
2. Run `hwrf_create_nest_1x_10m.exe` to rebalance the inner nest domain data. This will generate two data files: `new_gfs_d01` and `new_gfs_d02` which contain the rebalanced outer and inner domain data, respectively.
3. Create `trak.fnl.all_gfs`, a guess track file from `${MOAD_DATAROOT}/messages/storm.center`. For example, for a forecast of hurricane IKE starting 09/09/2008 00Z, the storm ID is 09L, the `storm.center` file contains the following data:
   - 21.7
   - -80.2
   and the guess track file should be in the following form:

   `72HDAS 80909 0 217 802 217 802 217 802 217 802 217 802 0 0 0 0 0 0 0 09L`

   where '72HDAS' is a fixed field, 80909 means 09/09/2008, 217 and 802 are the latitude and longitude (21.7N and 80.2W) multiplied by 10, and 09L is the storm id number.
4. Run `merge_nest_4x_10m2.exe` to merge inner domain (`new_gfs_d02`) and outer domain (`new_gfs_d01`) onto the 4x domain. This will generate the file containing the merged data on the 4x domain (`data_4x_gfs`) and a file containing sea mask and roughness length data (`roughness`).

5. Run `wrf_split.exe` to separate the data `4x_gfs` into an environment data (`gfs_env`) and a storm vortex (`storm_pert_gfs`). A file containing the storm's radius information will be generated too (`storm_radius_gfs`).

Output files in the directory `$DOMAIN_HOME/relocateprd`:
- `gfs_env`: environment fields from GFS data
- `roughness`: sea mask and roughness length from GFS data
- `storm_pert_gfs`: storm vortex from GFS data
- `storm_radius_gfs`: storm radius information from GFS data

**Status Check:**
If “failed” is not found in the standard output (stdout) and the files listed above exist, the script `relocate_stage2.ksh` run was successful.

### 4.3.5 relocate_stage3.ksh

Edit the script to set:

- `WRF_ROOT`: path to WRFV3
- `SID`: storm id
- `NAMELIST`: path to `namelist_analysis.input`
- `DOMAIN_HOME`: base path to the output directory

Then run `relocate_stage3.ksh`. This will create a new storm vortex by adjusting the previous cycle 6 -h forecast vortex (for a cycled run) or a bogus vortex (for a cold start run) to match the observed storm location, intensity and structure. The new storm vortex will be added back to the environment field generated using GFS data. The initial conditions `wrfinput_d01` and `wrfanl_d02` will be updated with the new storm vortex fields (see Fig. 4.5 and Fig. 4.6).

**Overview:**
1. For cold start runs: run `hwrf_anl_bogus_10m.exe` to create a bogus storm and add into the environmental flow on the 4x domain grid. This will generate `new_data_4x`.
2. For cycled runs:
   a. Run `merge_nest_4x_step2.exe` to get the file containing the sea mask and roughness length data from `new_gfs_d01` and `new_gfs_d02` which are generated in stage 2.
b. Run `hwrf_anl_4x_step2.exe` to adjust the storm vortex obtained in stage 1 (`storm_pert_new`) and add the new storm vortex into the environment flow (`gfs_env`) on the 4x domain grid. This will produce a new file (`new_data_4x`) containing the combined environment flow and the adjusted storm vortex.

c. If the combined vortex + environmental flow is less than the observed maximum wind speed, discard the file `new_data_4x` generated in step 2 and run `hwrf_anl_cs_10m.exe` to further adjust the vortex. This will produce a new version of `new_data_4x` which contains the combined environment flow and the adjusted storm vortex.

3. In the following steps the only difference between cold start and cycled runs is that for the storm radius information, the file `storm_radius` is used for cycled runs and `storm_radius_gfs` is used for cold start runs.

4. Run `hwrf_inter_4to6.exe` to interpolate the `new_data_4x` from the 4x domain onto the outer domain grid. This will produce the new `data_merge_d01`.

5. Run `hwrf_inter_4to2.exe` to interpolate the `new_data_4x` from the 4x domain onto the ghost domain grid. This will produce the new `data_merge_2x`.

6. Run `diffwrf_3dvar.exe` to convert the unformatted `data_merge_d01` to netcdf format `wrfinput_d01`.

7. Run `diffwrf_3dvar.exe` to convert the unformatted `data_merge_2x` to netcdf format `wrfghost_d02`.

8. Run `diffwrf_3dvar.exe` to convert the netcdf format `wrfinput_d01`, `wrfanl_d02` and `wrfghost_d02` to unformatted data files `new_hdas_d01`, `new_gfs_d02` and `new_ght_d02`, respectively.

9. Run `hwrf_inter_2to6.exe` to interpolate the files `new_hdas_d01`, `new_gfs_d02` and `new_ght_d02` to the outer domain grid. This will produce the merged data on the outer domain grid (`data_merge_d01`).

10. Run `hwrf_inter_2to1.exe` to interpolate the date in file `new_ght_d02` from the ghost domain grid to the inner nest domain grid. This will produce the merged data on the inner nest grid (`data_merge_d02`).

11. Run `diffwrf_3dvar.exe` to convert the unformatted files `data_merge_d01` and `data_merge_d02` to netcdf format `wrfinput_d01` and `wrfinput_d02`.

12. Rename `wrfinput_d02` to `wrfanl_d02`.

13. `wrfinput_d01` and `wrfanl_d02` are ready to be used by `wrf.exe` to do hurricane forecast.

Output files in the directory `${DOMAIN_HOME}/relocateprd`:

- `wrfinput_d01`: initial condition for the outer domain containing the new vortex
- `wrfanl_d02*`: initial condition for the inner nest domain containing the new vortex

Status Check:
If “failed” is not found in the standard output (stdout) and the files listed above exist, the script `relocate_stage3.ksh` run was successful.
Figure 4.2. HWRF initialization procedures before vortex adjustments.
Figure 4.3. Diagram of HWRF vortex initialization Stage 1 procedures. The color coding is described in Fig. 4.2.
Figure 4.4. Diagram of HWRF vortex initialization Stage 2 procedures. The color coding is described in Fig. 4.2.
Figure 4.5. Diagram of HWRF vortex initialization Stage 3 initial procedures. The color coding is described in Fig. 4.2.
Figure 4.6. Diagram of HWRF vortex initialization Stage 3 final procedures. The color coding is described in Fig. 4.2.
4.4 HWRF Vortex Initialization Executables

4.4.1 diffwrf_3dvar.exe

a) **FUNCTION:**
converts netcdf input to unformatted file (when first argument is "storm_relocate")

**INPUT:**
netcdf format wrfinput_d01 or previous cycle 6-hr forecast

**OUTPUT:**
unformatted data file

**USAGE:**

diffwrf_3dvar.exe storm_relocate(1st arg) wrfout_d01(2nd arg) flnm3(3rd arg) old_hwrf_d01(4th arg)

b) **FUNCTION:**
updates existing netcdf file with new unformatted file (when first argument is "3dvar_update")

**INPUT:**
unformatted file containing new vortex fields.

**OUTPUT:**
updated netcdf input

**USAGE:**

diffwrf_3dvar.exe 3dvar_update wrfinput_d01 data_merge_d01
update wrfinput_d01 with an unformatted file data_merge_d01 which contains new vortex fields.

4.4.2 hwrf_create_nest_1x_10m.exe

**FUNCTION:**
rebalances inner nest data
**INPUT:**

$gesfhr(=6)$ is the last digit of the input and output file unit number.

`new_gfs_d02` (fort.46)

`new_gfs_d01` (fort.26)

**OUTPUT:**

`new_data_1x` (fort.56) $\rightarrow$ `new_gfs_d02` - outer domain data interpolated to inner domain

**USAGE:**

`echo $gesfhr | hwrf_create_nest_1x_10m.exe`

### 4.4.3 *hwrf_inter_2to6.exe*

**FUNCTION:**

interpolates from ghost domain to outer domain

**INPUT:**

$gesfhr(=6)$

`new_gfs_d02` (fort.26) - hwrf inner grid data

`new_ght_d02` (fort.36) - data source on ghost grid data

`new_hdas_d01` (fort.46) - outer domain grid data

`storm_radius` (fort.85) - storm radius input from `wrf_split` in either stage 1 (cycled run) or stage 2 (Cold start)

**OUTPUT:**

`data_merge_d01` (fort.56) - interpolated data on outer domain

**USAGE:**

`echo $gesfhr | hwrf_inter_2to6.exe`

### 4.4.4 *hwrf_pert_ct.exe*

**FUNCTION:**

adjusts storm vortex (`storm_pert`)

**INPUT:**

$gesfhr(=6)$

`hdas_atcfunix` (fort.12) - storm track from previous 6 h forecast

`tcvitals.as` (fort.11) - storm center obs

`wrf_env` (fort.26) - environmental flow from previous6 hr forecast (`wrf_split's`
output)
storm_pert (fort.71) - separated 3D vortex field (wrf_split's output)

OUTPUT:
storm_pert_new (fort.58) - adjusted storm perturbation
storm_size_p (fort.14) - storm size information
storm_sym (fort.23) - storm symmetry information
storm_pert_step1_1 (fort.35) - level 2 slice of fields (diagnostic, not used later)

USAGE:
echo $gesfhr | hwrf_pert_ct.exe

4.4.5 hwrf_anl_4x_step2.exe

FUNCTION:
adjusts the storm vortex obtained in stage 1 (storm_pert_new) and adds the new
storm vortex to the environment flow (gfs_env) on the 4x domain grid.

INPUT:
$gesfhr(=6)
storm_size_p (fort.14) - input from stage 1
tcvitals.as (fort.11) - storm center obs
gfs_env (fort.26) - GFS environmental flow
storm_pert_new (fort.71) - adjusted storm perturbation from stage 1
roughness (fort.46) - roughness from merge_nest_4x_step2
storm_sym (fort.23) - symmetric part of storm

OUTPUT:
wrf_env_new (fort.36) - new environmental flow
new_data_4x (fort.56) - adjusted field on 4x domain.

USAGE:
echo $gesfhr | hwrf_anl_4x_step2.exe

4.4.6 hwrf_inter_4to2.exe

FUNCTION:
interpolates from 4x domain onto ghost domain.

INPUT:
$gesfhr (=6)
tcvitals.as (fort.11) - storm center obs
new_data_4x (fort.26) - adjusted storm on 4x domain
new_ght_d02 (fort.36) - ghost inner domain data

OUTPUT:
data_merge_2x (fort.56) - merged data on ghost domain.

USAGE:
echo $gesfhr | hwrf_inter_4to2.exe

4.4.7 **hwrf_split.exe**

FUNCTION:
splits the vortex from the background (environmental) field.

INPUT:
$gesfhr (=6)
$ibgs (=1)
$st_int (the 68-69 characters in the tcvital.as)
tcvitals.as (fort.11) - storm center obs
data_4x_hwrf (fort.26) - merged data, on 4x domain, from inner and outer
domains
trak.fnl.all (fort.30) - storm center guess
old_hwrf_d01 (fort.46) - outer domain data

OUTPUT:
wrf_env (fort.56) - environmental flow
storm_pert (fort.71) - separated 3D vortex field
storm_radius (fort.85) - average of model and observed storm radius
rel_inform.$cdate (fort.52) - diagnostics file (obs-previous 6hr forecast)
vital_syn.$cdate (fort.55) – information for generating bogus if storm not found
in previous 6 hr

USAGE:
echo ${gesfhr} ${ibgs} ${st_int} | hwrf_split.exe

4.4.8 **hwrf_anl_bogus_10m.exe**

FUNCTION:
creates a bogus storm and add into the environmental flow.

INPUT:
$gesfhr(=6)
tcvitals.as (fort.11) – observed storm center
gfs_env (fort.26) - GFS environmental flow
data_4x_gfs (fort.36) - merged GFS inner/outer domain data
storm_pert_gfs (fort.61) - separated GFS 3D vortex field
roughness (fort.46) - roughness info for BL calculation
storm_radius_gfs (fort.85)
hwrf Storm cyn axisy 47 (fort.75) input static vortex data

OUTPUT:
new_data_4x: combined envir flow/bogus field on 4x domain

USAGE:
echo $gesfhr | hwrf_anl_bogus_10m.exe

4.4.9 hwrf_create_trak_guess.exe

FUNCTION:
guesses storm center from previous 6hr forecast position.

INPUT:
$storm_id (storm id)
$sih (model initial hour)
tcvitals.as (fort.11) – observed storm center
hdas_atcfunix (fort.12) - stormid*atcfunix*[combine/parent] from previous cycle 6-hr forecast track file

OUTPUT:
trak.fnl.all (fort.30) - storm center guess (at 0,3,6 9 hrs)

USAGE:
echo $storm_id $ih | hwrf_create_trak_guess.exe

4.4.10 hwrf_inter_4to6.exe

FUNCTION:
interpolate from 4x domain onto outer domain.

INPUT:
$gesfhr
tcvitals.as (fort.11) – observed storm center
new_gfs_d01 (fort.26) - outer domain adjusted GFS data
new_data_4x (fort.36) - adjusted storm
new_gfs_d01 (fort.46) - outer domain adjusted GFS data
storm_radius (fort.85)

OUTPUT:
data_merge_d01 (fort.56) - merged data on outer domain

USAGE:
echo $gesfhr | hwrf_inter_4to6.exe

4.4.11 hwrf_anl_cs_10m.exe

FUNCTION:
further adjusts the storm vortex when combined vortex + environmental flow is less than the observed maximum wind speed

INPUT:
$gesfhr (=6)
tcvitals.as (fort.11) – observed storm center
wrf_env_new (fort.26) - new environmental flow (from hwrf_anl_4x_step2)
storm_sym (fort.23) - symmetric part of storm (from stage 1)
roughness (fort.46) - roughness info for BL calculation (from storm_radius (fort.85) (from stage 1)
hwrf_storm_cyn_axisy_47 (fort.75) input static vortex data

OUTPUT:
test_data (fort.25)
new_data_4x (fort.56) - adjusted field on 4x domain when combined vortex + environmental flow is less than the observed maximum wind speed - replaces previous file

USAGE:
echo $gesfhr | hwrf_anl_cs_10m.exe

4.4.12 hwrf_merge_nest_4x_10m2.exe

FUNCTION:
merges inner and outer domains onto a 4x domain. the 4x domain is about 40x40 deg with the resolution of the inner nest domain. The 4x domain is
centered according to the NHC storm message. It's smaller than the outer
domain but much larger than the inner domain and ghost domain.

INPUT:
$gesfhr(=6)$ $gesfhr$ last digit of the input/output file
$st\_int$ (the 68-69 characters in the tcvital.as)
$ibgs(=1)$ an argument indicating if a cold start (ibgs=1) or a cycled run
(ibgs=0)
tcvitals.as (fort.11) – observed storm center
new_gfs_d02 (fort.46) - inner domain data
new_gfs_d01 (fort.47) - outer domain data
new_gfs_d01 (fort.26) - outer domain data
new_gfs_d02 (fort.36) - inner domain data
trak.fnl.all_gfs (fort.30) - storm track

OUTPUT:
data_4x_gfs (fort.56) - merged data from inner and outer domains
roughness (fort.66) - sea-mask (1=sea, 0=land) and ZNT (roughness length)
merged from (inner+outer OR inner only?) onto the 4x domain.

USAGE:
echo ${gesfhr} ${st\_int} ${ibgs} | hwrf_merge_nest_4x_10m2.exe

4.4.13 hwrf_merge_nest_4x_step1.exe

FUNCTION:
merges outer and inner domain data onto 4x domain.

INPUT:
$gesfhr(=6)$
$st\_int$ (the 68-69 characters in the tcvital.as)
$ibgs(=0)$ an argument indicating if a cold start (ibgs=1) or a cycled run
(ibgs=0)
$CLAT$ latitude of the center of observed storm (used to center 4x domain)
$CLON$ longitude of the center of observed storm(used to center 4x domain)
tcvitals.as (fort.11) observed storm center, location stored in data_4x_hwrf
old_hwrf_d01 (fort.26) outer domain data
old_hwrf_d02 (fort.36) inner domain data

OUTPUT:
data_4x_hwrf (fort.56) 4x data - merged data on 4x domain
**USAGE:**
`echo ${gesfhr} ${st_int} ${ibgs} ${CLAT} ${CLON} | hwrf_merge_nest_4x_step1.exe`

### 4.4.14 hwrf_inter_2to1.exe

**FUNCTION:**
interpolates from ghost domain to inner domain.

**INPUT:**
- `$gesfhr(=6)`
  - `new_ght_d02 (fort.26)` - data on ghost domain
  - `new_gfs_d02 (fort.36)` – data on inner domain

**OUTPUT:**
- `data_merge_d02 (fort.56)` - interpolated data on inner domain

**USAGE:**
`echo ${gesfhr} | hwrf_inter_2to1.exe`

### 4.4.15 hwrf_merge_nest_4x_step2.exe

**FUNCTION:**
merges inner and outer domains onto 4x domain.

**INPUT:**
- `$gesfhr(=6)`
- `$st_int` (tcvital characters 68-69)
- `$ibgs` an argument indicating if a cold start (ibgs=1) or a cycled run (ibgs=0)
- `tcvitals.as (fort.11)` – observed storm center, used for centering 4x domain
- `new_gfs_d02 (fort.46)` - inner domain data from stage 2
- `new_gfs_d01 (fort.47)` - outer domain data from stage 2

**OUTPUT:**
- `roughness (fort.66)` - roughness length and sea mask.

**USAGE:**
`echo ${gesfhr} ${st_int} ${ibgs} | hwrf_merge_nest_4x_step2.exe`
Chapter 5: Ocean Initialization of POM-TC

5.1 Introduction

This document explains how to run the ocean initialization of the Princeton Ocean Model (POM) component of the HWRF Model, available from the DTC. Henceforth, this version of the model will be referred to as POM-TC. Users are also encouraged to read the HWRF Scientific Documentation.

5.2 Scripts

The scripts can be found in the directory

${HOME}/HWRF/src/pomtc/ocean_scripts

You will see five scripts, four of which are “kickit” scripts. The fifth script, “gfdl_pre_sortvit.sh,” is used internally by the fourth “kickit” script and should not be changed.

The four “kickit” scripts are designed to be run in order from kickit01 through kickit04. If you are only running the default tutorial case, 2008082800, then no changes should be needed to these scripts either, and you can easily run the entire ocean initialization as follows:

[prompt]$ ./kickit01_sharpn.sh
[prompt]$ ./kickit02_getsst.sh
[prompt]$ ./kickit03_phase3.sh
[prompt]$ ./kickit04_phase4.sh

If you want to run a case other than the default tutorial case, edit the four “kickit” scripts, and then run the ocean initialization as described above. If running on Bluefire, the only two lines in each “kickit” script that need to be edited to run a different case are as follows:

stormid=07L   # e.g. SID = 07L
start_date=2008082800  # e.g. YYYYMMDDHH = 2008082800

If you are running on a machine other than Bluefire, you may need to edit the following lines in each “kickit” script as well, depending on the paths to your input datasets, source code, and output work directories, respectively:

data_d=/ptmp/HurrTutorial/datasets
sorc_d=${HOME}/HWRF/src/pomtc
work_d=/ptmp/${USER}/HWRF/${stormid}/${start_date}/oceanprd
5.3 Procedures in the four “kickit” Scripts

5.3.1 kickit01_sharpn.sh

1. Set the storm ID.
2. Set the starting date for POM-TC as YYYYMMDDHH.
3. Set which ocean climatology to use (default is GDEM).
4. Set the ocean region (always use “united” with sharpening).
5. Define the directories.
6. Create the work directory if it does not already exist.
7. Create the sharpening directory, overwriting old attempts.
8. Slice up the storm ID and starting date.
9. Use some pieces of the starting date to select the two climatology months.
10. Create parameter “input_sharp” depending on the chosen climatology.
11. Create symbolic links for the input files.
12. Run the sharpening code.
14. End the sharpening script by returning to the “ocean_script” directory.

5.3.2 kickit02_getsst.sh

1. Set the storm ID.
2. Set the starting date for POM-TC as YYYYMMDDHH.
3. Define the directories.
4. Create the work directory if it does not already exist.
5. Create the sst/mask/lonlat directory, overwriting old attempts.
6. Slice up the starting date, using HH to define the model cycle.
7. Create symbolic links for the GFS spectral input files.
8. Increase ulimit –s to prevent a segmentation fault.
9. Run the getsst code.
11. End the getsst script by returning to the “ocean_script” directory.

5.3.3 kickit03_phase3.sh

1. Set the storm ID.
2. Set the starting date for POM-TC as YYYYMMDDHH.
3. Set the ocean region (always use “united” with sharpening).
4. Define the directories.
5. Create the work directory if it does not already exist.
6. Create the phase3 directory, overwriting old attempts.
7. Slice up the starting date, using HH to define the model cycle.
8. Modify the phase3 parameter file by including the starting date.
9. Create symbolic links for all input files.
10. Run the POM-TC code for phase3.
12. End the phase3 script by returning to the “ocean_script” directory.

5.3.4 kickit04_phase4.sh

1. Set the storm ID.
2. Set the starting date for POM-TC as YYYYMMDDHH.
3. Set the ocean region (always use “united” with sharpening).
4. Define the directories.
5. Create the work directory if it does not already exist.
6. Create the phase4 directory, overwriting old attempts.
7. Slice up the starting date, using HH to define the model cycle.
8. Use programs gfdl_date2day and gfdl_day2date to manipulate the starting date, which is necessary if choosing to back up 3 days (as in the operational HWRF). Then, use this new starting date (start_date2) as the start of phase4.
9. The purpose of backing up 3 days is to end phase4 at the coupled starting date.
10. Run operational script gfdl_pre_sortvit.sh to extract track information for the specified storm from the yearly NHC hurricane message file (i.e. tcvitals).
11. Then, append a line of zeros to the end of the track file.
12. Modify the phase4 parameter file by including the starting date, as well as the track and phase3 restart files.
13. Create symbolic links for all input files.
15. Rename the phase4 restart file in preparation for coupled HWRF/POM-TC run.
16. End the phase4 script by returning to the “ocean_script” directory.

5.4 Executables

5.4.1 gfdl_date2day.exe

**FUNCTION:**
convert a date integer in YYMMDDHH format to Julian day in year YY

**INPUT:**
date2day.inp

**OUTPUT:**
5.4.2 *gfdl_day2date.exe*

**FUNCTION:**
Convert year and Julian day input to date integer in YYMMDDHH Format

**INPUT(S):**
day2date.inp

**OUTPUT(S):**
fort.61 (day2date.dat)

**USAGE:**
${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_day2date.exe < day2date.inp

5.4.3 *gfdl_getsst.exe*

**FUNCTION:**
Extract SST, land-sea mask, and lon/lat data from the GFS spectral files

**INPUT:**
fort11 (gfs.$\{start_date\}.t$\{cyc\}z.sfcanl)
fort.11 (gfs.$\{start_date\}.t$\{cyc\}z.sfcanl)
fort.12 (gfs.$\{start_date\}.t$\{cyc\}z.sanl)

**OUTPUT:**
fort.23 (lonlat.gfs)
fort.74 (sst.gfs.dat)
fort.77 (mask.gfs.dat)
getsst.out

**USAGE:**
${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_getsst.exe >> getsst.out

5.4.4 *gfdl_sharp_mcs_rf_l2m_rmy5.exe*

**FUNCTION:**
run the sharpening program, which takes the T/S climatology, horizontally-
interpolates it onto the POM-TC grid for the United region domain, assimilates a land/sea mask and bathymetry, and employs the diagnostic, feature-based modeling procedure described in the HWRF Scientific Documentation.

**INPUT:**

`input_sharp`

- `fort.66 (gfdl_ocean_topo_and_mask.${region})`
- `fort.8 (gfdl_gdem.${mm}.ascii)`
- `fort.90 (gfdl_gdem.${mm}.ascii)`
- `fort.24 (gfdl_ocean_readu.dat.${mm})`
- `fort.82 (gfdl_ocean_spinup_gdem3.dat.${mm})`
- `fort.50 (gfdl_ocean_spinup_gspath.${mm})`
- `fort.55 (gfdl_ocean_spinup.BAYuf)`
- `fort.65 (gfdl_ocean_spinup.FSgsuf)`
- `fort.75 (gfdl_ocean_spinup.SGYREuf)`
- `fort.91 (mmdd.dat)`
- `fort.31 (hwrf_gfdl_loop_current_rmy5.dat.${yyyymmdd})`
- `fort.32 (hwrf_gfdl_loop_current_wc_ring_rmy5.dat.${yyyymmdd})`

**OUTPUT:**

- `fort.13 (gfdl_initdata.${region}.${mm})`
- `sharpn.out`

**USAGE:**

```
${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_sharp_mcs_rf_l2m_rmy5.exe < input_sharp > sharpn.out
```

5.4.5 **gfdl_ocean_united.exe**

**FUNCTION:**

run POM-TC ocean phase 1 or phase 2 (also known historically as ocean phase 3 and phase 4, respectively, as in the model code).

**INPUT:**

- `fort.10 (parameters.inp)`
- `fort.15 (nullfile if phase 1; track if phase 2)`
- `fort.21 (sst.gfs.dat)`
- `fort.22 (mask.gfs.dat)`
- `fort.23 (lonlat.gfs)`
- `fort.13 (gfdl_initdata.${region}.${mm})`
- `fort.66 (gfdl_ocean_topo_and_mask.${region})`
- `fort.14 (not used if phase 1; RST.phase3.${region} if phase 2)`

**OUTPUT:**

`RST.yymmddhh (RST.phase3.${region})`
if phase 1; RST.final
if phase 2)
phase3.out if phase 1;
phase4.out if phase 2

**USAGE:**
Phase 1: ${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_ocean_${region}.exe > phase3.out
Phase 2: ${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_ocean_${region}.exe > phase4.out
Chapter 6: How to Run HWRF

6.1 Introduction

HWRF is an atmosphere-ocean coupled forecast system, which includes an atmospheric component (WRF-NMM), an ocean component (POM-TC) and the NCEP Coupler. Therefore HWRF is a Multiple Program Multiple Data (MPMD) system which consists of three executables: WRF, POM-TC and Coupler. After the ocean and vortex initializations are successfully completed, the coupled HWRF system run can be submitted. The commands issued for the model run depend on the computer platform.

6.2 How to Run HWRF Using the Script wrf.ksh

This section describes how to use the script ${HOME}/src/hwrf-utilities/scripts/wrf.ksh to run the coupled HWRF forecast on two types of platforms: the IBM/AIX and Linux machines. The user should be responsible to understand the batch system used on the platform where the HWRF system will be run, if that system is not covered in this document.

For the IBM platform which uses the AIX Operation System and the batch system Load Sharing Facility (LSF), the script wrf.ksh should contains the LSF options listed below that the users will need to edit:

```
#BSUB -P 99999999 # Project 99999999
#BSUB -a poe # select poe
#BSUB -n 32 # number of total (MPI) tasks
#BSUB -R "span[ptile=32]" # run a max of 16 tasks per node
#BSUB -J hwrf # job name
#BSUB -o hwrf.%J.out # output filename
#BSUB -e hwrf.%J.out # error filename
#BSUB -W 2:30 # wallclock time
#BSUB -q debug # queue
#BSUB -K # Don't return prompt until the job is finished
```

For a Linux platform which uses the Sun Grid Engine (SGE) batch system, the script wrf.ksh should contains the SGE options listed below that the users will need to edit:

```
#$ -cwd -V # directories and export variables
#$ -N HWRF # Job name
#$ -A 99999999 # Project Account
```
After setting up the batch system options, edit the script to set up the environment variables:

\[
\begin{align*}
WRF\_ROOT: & \quad \text{path to the WRF code.} \\
& \quad (\text{for example,}\ \${HOME}/HWRF/src/WRFV3) \\
SID: & \quad \text{storm ID} \\
& \quad (\text{for example,}\ 07L\ \text{for the seventh storm in Atlantic Basin}) \\
WRF\_MODE: & \quad \text{Mode for the WRFV3 run. Must be one of: ‘analysis’, ‘ghost’, or} \\
& \quad \text{‘main’. For the coupled forecast, choose ‘main’} \\
NAMELIST: & \quad \text{Path to the WRFV3 namelist} \\
& \quad (\text{for example,}\ \${HOME}/HWRF/src/hwrf-utilities/parm/namelist\_main.input) \\
DOMAIN\_HOME: & \quad \text{Path to the top directory of the forecast output.} \\
& \quad (\text{for example,}\ /ptmp/\${USER}/HWRF/\${SID}/$\{YYYY}\$\{MM}\$\{DD}\$\{HH}\})
\end{align*}
\]

Also edit the script to specify the path to the file \textit{gfdl\_ocean\_topo\_and\_mask.united} and link it to \texttt{fort.66}:

\texttt{ln -s path/gfdl\_ocean\_topo\_and\_mask.united fort.66}

Note that \texttt{wrf.ksh} is run three times for each forecast, two of them are 1-minute runs used in vortex initialization (see Chapter 4), and the third one is for the actual coupled forecast. The \texttt{WRF\_MODE} used for these three \texttt{wrf.ksh} runs are “analysis”, “ghost” and “main”, respectively. The three \texttt{wrf.ksh} runs use different namelists too, which are specified by \texttt{namelist\_${WRF\_MODE}\_input}.

After the batch system options and environment variables are defined, run the script: \texttt{wrf.ksh} using the command:

- On IBM with LSF: \texttt{bsub < wrf.ksh}
- On Linux with SGE: \texttt{qsub wrf.ksh}

### 6.2 Overview of the Script

1. Extract the initialization time components from $\{NAMELIST\}$
2. Compute forecast length that will be used in the namelist for the coupler.
3. Define the working directory as `${DOMAIN_HOME}/wrfprd` and enter the working directory.

4. Link the required input files to the work directory:
   - geogrid static files: `geo_nmm.d01.nc` and `geo_nmm_nest.l01.nc`
   - wrfbdy file: `wrfbdy_d01`
   - wrfinput file: `wrfinput_d01`
   - wrfanl_d02 file: `wrfanl_d02_${YYYY}-${MM}-${DD}_${HH}:00:00`
   - `fort.65`
   - WRF static files

5. Run `hwrf_swcorner_dynamic.exe` to calculate the istart and jstart values for the nest in `namelist_main.input`. Update the istart and jstart in `namelist_main.input`.

6. Construct the namelist `PARAMETERS.inp` for the ocean model.

7. Link the input files for the ocean model:
   - `gfdl_ocean_topo_and_mask.united`
   - `gfdl_initdata.united.${MM}, ${MM}` is the month for the forecast storm
   - `RST.final`
   - `sst.gfs.dat`
   - `mask.gfs.dat`
   - `lonlat.gfs`
   - `track`

8. Construct the namelist for the coupler.

9. Submit the coupled run.
   - On IBM with LSF:
     Use the command `mpirun.lsf`
     ```
     mpirun.lsf -cmdfile cmdfile
     ```
     where `cmdfile` is a file containing the list of the executables. For example, the `cmdfile` below indicates that the coupled run will be submitted to 90 processors, one for the coupler (`hwrf_pwmc.exe`), one for the ocean model (`hwrf_ocean_united.exe`) and 88 for `wrf.exe`:
     ```
     hwrf_pwmc.exe
     hwrf_ocean_united.exe
     wrf.exe
     wrf.exe
     .... (totally 88 wrf.exe)
     wrf.exe
     wrf.exe
     ```
   - On Linux with SGE:
     Use the command `mpirun`
     ```
     mpirun
     ```
     For example, the following command will run the coupled model using 90 processors, one for the coupler (`hwrf_pwmc.exe`), one for the ocean model (`hwrf_ocean_united.exe`) and 88 for `wrf.exe`
6.3 Output Files in the Directory

$\{DOMAIN\_HOME\}/wrfprd A successful run of wrf.ksh will produce output files with the following naming convention:

\texttt{wrfout\_d01\_yyyy-mm-dd\_hh:mm:ss}

For example, the first WRF output files for a run started at 0000 UTC, 28\textsuperscript{rd} August 2008 would be:

\texttt{wrfout\_d01\_2008-08-28\_00:00:00} and \texttt{wrfout\_d02\_2008-08-28\_00:00:00}

The ocean model will produce diagnostic output files with the following naming convention: \texttt{GRADS.ymmddh}

For example, the first POM-TC output file for a run started at 0000 UTC, 28\textsuperscript{rd} August 2008 would be: \texttt{GRADS.08082800}

6.4 Status Check

To check whether the run was successful, look for “SUCCESS COMPLETE WRF” at the end of the log file (e.g., \texttt{rsl.out.0000}).

6.5 Executables

6.5.1 \texttt{wrf.exe}

\textbf{FUNCTION:}

atmosphere component of HWRF

\textbf{INPUT:}

geogrid static files: \texttt{geo\_nmm\_d01\_nc} and \texttt{geo\_nmm\_nest\_l01\_nc}
\texttt{wrfbdy} file: \texttt{wrfbdy\_d01}
\texttt{wrfinput} file: \texttt{wrfinput\_d01}
\texttt{wrfanl\_d02} file: \texttt{wrfanl\_d02\_$\{YYYY\}_-$\{MM\}_-$\{DD\}_-$\{HH\}\_00:00}
\texttt{fort.65} and gravity wave drag files
WRF static files
\texttt{namelist\_main.input}
OUTPUT:
A successful run of wrf.ksh will produce output files with the following naming convention:

\textit{wrfout\_d01\_yyyy-mm-dd\_hh:mm:ss}

For example, the first WRF output file for a run started at 0000 UTC, 28rd August 2008 would be:

\textit{wrfout\_d01\_2008-08-28\_00:00:00}

and

\textit{wrfout\_d02\_2008-08-28\_00:00:00}

USAGE:
For a coupled HWRF forecast, wrf.exe must be submitted with the coupler and the ocean model (see Section 6.2).

6.5.2 \texttt{hwrf\_pwmc.exe}

FUNCTION:
coupler that links the atmospheric component \texttt{wrf.exe} and oceanic component POM-TC

INPUT:
coupler namelist: \texttt{cpl.nml}

OUTPUT:
None

USAGE:
For a coupled HWRF forecast, the coupler \texttt{hwrf\_pwmc.exe} must be submitted to the computers with the atmosphere model \texttt{wrf.exe} and the ocean model \texttt{hwrf\_ocean\_united.exe} (see Section 6.2).

6.5.3 \texttt{hwrf\_ocean\_united.exe}

FUNCTION:
 oceanic model for HWRF.

INPUT:
\texttt{gfdl\_ocean\_topo\_and\_mask\_united}
\texttt{gfdl\_initdata\_united.$\{MM\}$, $\{MM\}$ is the month for the forecast storm}
\texttt{RST.final}
The ocean model will produce output files with the following naming convention: GRADS.yymmddhh
For example, the first POM-TC output file for a run started at 0000 UTC, 28rd August 2008 would be:
GRADS.08082800

For a coupled HWRF forecast, the ocean model hwrf_ocean_united.exe must be submitted to the computers with the atmosphere model wrf.exe and the coupler hwrf_pwmc.exe (see Section 6.2).

6.5.4 hwrf_swcorner_dynamic.exe

FUNCTION:

hwrf_swcorner_dynamic calculate the lower-left corner of the nest as (i_parent_start, j_parent_start) location in the nest’s parent domain. i_parent_start and j_parent_start are calculated at run-time.

INPUT:

storm center location: storm.center
domain center location: domain.center
fort.12: namelist_main.input

OUTPUT:

set_nest, which contains the i_parent_start and j_parent_start. for example the following set_nest file specifies that the nest domain lower-left corner location is at (98,208) on parent domain grid:

istart=00098
jstart=00208

USAGE:

${HOME}/HWRF/src/hwrf-utilities/exec/hwrf_swcorner_dy
Chapter 7: HWRF Post Processor

7.1 Introduction

The NCEP WRF Postprocessor was designed to de-stagger HWRF parent and nest domains output, compute diagnostic variables and interpolate from their native grids to National Weather Service (NWS) standard levels (pressure, height, etc.) and standard output grids (latitude/longitude, AWIPS, Lambert Conformal, polar-stereographic, etc.), in NWS and World Meteorological Organization (WMO) GRIB format. This package also combines the parent and nest domains output to one combined domain grid.

Information on how to acquire and build the WPP code is available in Chapter 2.

7.2 Script

The WPP scripts are distributed in the tar file WPPV3.tar.gz and, following the procedure outlined in Chapter 2, will be expanded in the directory of 

\$HOME/HWRF/src/hwrf-utilities/scripts:

- run_wrfpost
- run_wrfpostandgempak
- run_wrfpostandgrads

All the three scripts can be used to run wrfpost to post-process HWRF output. run_wrfpostandgempak and run_wrfpostandgrads will, in additional to postprocessing HWRF output, also generate plots from the post-processed HWRF output using GEMPAK and GrADS, respectively. In this user’s guide, we will explain how to use run_wrfpost to run wrfpost. Detailed information on run_wrfpostandgempak and run_wrfpostandgrads are available in WRF-NMM User’s Guide.

Edit the script run_wrfpost to set the following environment variables:

- **WPP_ROOT**: Path to the WPPV3 code  
  (for example, \$HOME/HWRF/src/WPPV3)
- **SID**: Storm ID (for example, 07L for GUSTAV 2008)
- **NAMELIST**: Namelist used in the HWRF coupled forecast  
  (for example \$HOME/HWRF/src/hwrf-utilities/parm/namelist_main.input)
- **DOMAIN_HOME**: Base path to the output directory

Run the script by the command: run_wrfpost
7.3 Overview of the Script

1. Get the start time, end time and history output interval from `namelist_main.input`, and calculate the forecast length and the number of output files to post process.

2. Compute boundaries and copygb grid parameters for final GRIB files, based on the storm location information in `$DOMAIN_HOME/messages/storm.center`.

3. For each forecast time:
   - Link the microphysical table `~/HWRF/src/hwrf-utilities/parm/hwrf_eta_micro_lookup.dat` and control file `~/HWRF/src/hwrf-utilities/parm/hwrf_cntrl.hurcn` to `wrf_control.parm` in the working directory.
   - In `~/HWRF/src/hwrf-utilities/parm/hwrf_cntrl.hurcn` the user can specify the variables that will be post processed (for more information see WRF-NMM documentation), and if changes in the post-processed variables are desired, the control file `hwrf_cntrl.hurcn` needs to be altered. For HWRF, the following variables, which are required by the vortex tracker (see Chapter 4), should be post processed:
     - absolute vorticity at 10 m, 850 mb and 700 mb
     - MSLP
     - geopotential height at 850 and 700 mb
     - wind speed at 10 m, 850 mb, 700 mb and 500 mb.
   - Set up how many domains will be post-processed. For HWRF, the parent and nest domains (d01 and d02) are post-processed. For each domain:
     - Create namelist itag that will be read in by `wrfpost.exe` from stdin (unit 5). This namelist contains 4 lines:
       - Name of the WRF output file to be posted.
       - Format of WRF model output (netcdf for binary; choose netcdf for HWRF).
       - Forecast valid time (not model start time) in WRF format.
     - Model name (NMM or NCAR; choose NMM for HWRF)
       - Run wrfpost and check for errors. The execution command in the distributed scripts is for a single processor:
         `wrfpost.exe < itag > outpost`
       - Set up a copygb grid, `$hr_grid`, to interpolate the native wrfpost.exe output GRIB file to (see full description under Run copygb below). For both parent and nest domains the grid spacing in `$hr_grid` are set to 0.1 degree.
       - copygb is a program to horizontally interpolate a GRIB file to a user-specified grid. Run copygb to interpolate the `wrfpost.exe` output GRIB file to the copygb grid `$hr_grid` and check for errors.
copygb.exe -xg"${hr_grid}" WRFPRS_${domain}.${fhr}
wrprs_${domain}.${fhr}
where $\{hr\_grid\}$ refers to the output grid to which the native forecast is being interpolated, $WRFPRS_{domain}.{fhr}$ is the native Wrfpost.exe output GRIB file and $wrprs_{domain}.{fhr}$ is the copygb output GRIB file in which the forecast output is interpolated to the grid $\{hr\_grid\}$.

- Run $\{HOME\}/$HWRF/src/hwrf-utilities/exec/hwrf_combine.exe to get the combined domain output. The combined domain output has a grid spacing of 0.1 degree.
  - link the parent domain output to fort.401 (input)
  - link the nest domain output to fort.402 (input)
  - link the combined domain output to fort.403 (output)
  - use wgrib to get the number of records in the output files and write the number to a file named date_file
  - run hwrf_combine.exe:
    $\{HOME\}/$HWRF/src/hwrf-utilities/exec/hwrf_combine.exe < date_file
    the resulting $wrprs.{fhr}$ is the combined domain output in GRIB format at current forecast time.

4. Output files in the working directory $\{DOMAIN\_HOME\}/postprd:
   - wrfprs_d01.$\{fhr\}$ HWRF parent domain GRIB format output
   - wrfprs_d02.$\{fhr\}$ HWRF nest domain GRIB format output
   - wrfprs.$\{fhr\}$: HWRF combined domain GRIB format output

5. Status check: If “End of Output Job” is found in the standard output (stdout), the HWRF WPP script is finished successfully.

7.4 Executables

7.4.1 wrfpost.exe

**FUNCTION:**
converts HWRF native output (wrfout_d01 or wrfout_d02) to GRIB format.

**INPUT:**
Ferrier’s microphysics table
wrfpost control file $wrf\_cntrl.parm$
HWRF native output (wrfout_d01 or wrfout_d02)
namelist $itag$
OUTPUT:
HWRF output in GRIB format $WRFPRS_{${domain}}.${fhr}$

USAGE:
${HOME}/HWRF/src/WPPV3/exec/wrfpost.exe < itag

7.4.2  copygb.exe

FUNCTION:
interpolates a GRIB file to a user-specified grid.

INPUT:
user-specified grid (${hr_grid})
wrfpost.exe output ($WRFPRS_{${domain}}.${fhr}$)

OUTPUT:
GRIB file on the grid of ${hr_grid}: wrfprs_${domain}.${fhr}

USAGE:
${HOME}/HWRF/src/WPPV3/exec/copygb.exe -xg"${hr_grid}" $WRFPRS_{${domain}}.${fhr} wrfprs_${domain}.${fhr}

7.4.3  hwrf_combine.exe

FUNCTION:
combines HWRF parent and nest domain outputs into one combined domain.

INPUT:
$wrfprs_d01.${fhr}$ HWRF parent domain GRIB format output
$wrfprs_d02.${fhr}$ HWRF nest domain GRIB format output
date_file: ascii file containing the number of records in $wrfprs_d01.${fhr}$ and $wrfprs_d02.${fhr}$

OUTPUT:
$wrfprs.${fhr}$: the HWRF GRIB format output on the combined domain.
**USAGE:**

```
${HOME}/HWRF/src/hwrf-utilities/exec/hwrf_combine.exe <date_file
```
Chapter 8: GFDL Vortex Tracker

8.1 Introduction

The GFDL vortex tracker is a program that ingests model forecasts in GRIB format, objectively analyzes the data to provide an estimate of the vortex center position (latitude and longitude), and tracks the storm for the duration of the forecast. Additionally, it reports additional metrics of the forecast storm, such as intensity (maximum 10-m winds and the minimum mean sea level pressure - MSLP) and structure (wind radii for 34, 50 and 64 knot thresholds in each quadrant of each storm) at each output time. The GFDL vortex tracker requires the forecast grids to be on a cylindrical equidistant, latitude-longitude (lat/lon) grid. For HWRF, WPP is used to process the raw model output and create the GRIB files for the tracker.

The vortex tracker creates two output files containing the vortex position, intensity and structure information: one in Automated Tropical Cyclone Forecast (ATCF) format and another in a modified ATCF format.

The GFDL vortex tracker tracks the hurricane vortex center positions by searching for the average of the maximum or minimum of several parameters in the vicinity of an input first guess position of the targeted vortex. The primary tracking parameters are relative vorticity at 10 m, 850 mb and 700 mb, MSLP, and geopotential height at 850 and 700 mb. Secondarily, wind speed at 10 m, and 850 mb and 700 mb are used. Winds at 500 mb are used, together with other parameters, for advecting the storm and creating a first guess position for all times beyond initialization. Many parameters are used in order to provide more accurate position estimates for weaker storms, which often have poorly defined structures/centers.

Besides the forecast file in GRIB format, the vortex tracker also ingests a GRIB index file which is generated by running the program `grbindex`. The GRIB utility `wgrib` is also used for preparing data for the tracker. Both grbindex are wgrib were developed by NCEP; `grbindex` is distributed by DTC as part of the hwrf-utilities and `wgrib` can be obtained from http://www.cpc.noaa.gov/products/wesley/wgrib.html.

8.2 How to Run the GFDL Vortex Tracker Using Script

Find the GFDL vortex tracker script tracker.ksh in directory

```
$HOME/HWRF/src/hwrf-utilities/scripts
```

To run the vortex tracker:

1. Edit the script to set up paths to executable wgrib.
2. Edit the script to set up the following environmental variables:

```
DOMAIN_HOME: top level directory of HWRF output
  (for example, /ptmp/user/HWRF/09L/2008090900)
TRACKER: directory of the tracker executable
  (for example, ${HOME}/HWRF/src/gfdl vortextracker/trk_exec)
GRBINDEX: the executable gribindex.
SID: storm ID.
  (for example, 09L .ATCFNAME=HTU:)
```

Run the tracker script with the command: `tracker.ksh`

### 8.3 Overview of the script

The steps performed by script tracker.ksh are listed below:

- Obtain the model initialization date from `hwrf-utilities/parm/namelist_main.input`
- Extract, from the WPP output in `${DOMAIN_HOME}/postprd`, the fields needed by tracker for each output time and concatenate them together to produce one GRIB file. The extracted fields include:
  - winds at 10 m, 500 mb, 700 mb, and 850 mb
  - absolute vorticity at 10 m, 700 mb and 850 mb
  - geopotential height at 700 and 850 mb
  - MSLP
- Run `grbindex` to get a GRIB index file for the GRIB file generated in 1.
- Get the output times from the GRIB file generated in 1.
- Create a file, `fcst_minutes`, which contains the forecast output times the tracker will process
- Generate the namelist for the vortex tracker
- Link the input files (see GFDL vortex tracker software input description)
- Run the tracker executable `hwrf_gettrk.exe`
- Output will be generated in `${DOMAIN_HOME}/gvtprd`

### 8.4 Executables

#### 8.4.1 `hwrf_gettrk.exe`

**INPUT:**
- `fort.11`: GRIB file containing the HWRF forecast
- `fort.12`: TC vitals file containing the first guess location of the forecast vortex
For example, the following TC vitals file (this should be a 1-line file without line break) provides a first guess location for Hurricane Ike of 21.7 N and 80.2 W.

```
NHC 09L IKE 20080909 0000 217N 0802W 295 057 0967 1008 0371 36 028 0278 0278 0185 0278 D 0167 0167 0056 0167 72 255N 890W 0056 0056 -999 0056
```

`fort.14`: TC vitals file used for tropical cyclone-genesis tracking. This file is not used by HWRF. File `fort.14`, which can be blank, should exist in the directory where the tracker is run otherwise the tracker will stop.

`fort.15`: Forecast lead times (in minutes) the tracker will process.
For example, the following file specifies that the tracker will process the GRIB output for lead times 0, 180, 360 and 540 minutes

```
0 180 360 540
```

`fort.31`: a GRIB index file generated by the program of grbindex

**NAMELIST:**

<table>
<thead>
<tr>
<th>inp%bcc</th>
<th>First 2 digits of the year for the initial time of the forecast (e.g., the &quot;20&quot; in &quot;2009&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>inp%byy</td>
<td>Last 2 digits of the year for the initial time of the forecast (e.g., the &quot;09&quot; in &quot;2009&quot;)</td>
</tr>
<tr>
<td>inp%bmm</td>
<td>2-digit month (01, 02, etc) for the initial time of the forecast</td>
</tr>
<tr>
<td>inp%bdd</td>
<td>2-digit day for the initial time of the forecast</td>
</tr>
<tr>
<td>inp%bhh</td>
<td>2-digit hour for the initial time of the forecast</td>
</tr>
<tr>
<td>inp%model</td>
<td>Model ID number as defined by the user in the script. This is used in subroutine getdata to define what the GRIB IDs are for surface wind levels. Create a unique number in the script for your model and make sure you have the corresponding IDs set up for it in subroutine getdata. For HWRF use 17.</td>
</tr>
<tr>
<td>inp%lt_units</td>
<td>'hours' or 'minutes', this defines the lead time units used by the PDS in your GRIB header</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>atcfnum</td>
<td>Obsolete; can be set to any integer</td>
</tr>
<tr>
<td>atcfname</td>
<td>Character model ID that will appear in the ATCF output (e.g., GFSO, HWRF, AHW, etc)</td>
</tr>
<tr>
<td>atcfymdh</td>
<td>10-digit yyyymmddhh date that will be used in output text track files</td>
</tr>
<tr>
<td>trkrinfo%westb</td>
<td>For genesis runs, the western boundary for searching for new storms. Does not need to match the boundaries of your grid, it can be smaller than your grid.</td>
</tr>
<tr>
<td>trkrinfo%eastb</td>
<td>For genesis runs, the eastern boundary for searching for new storms. Does not need to match the boundaries of your grid, it can be smaller than your grid.</td>
</tr>
<tr>
<td>trkrinfo%northbd</td>
<td>For genesis runs, the northern boundary for searching for new storms. Does not need to match the boundaries of your grid, it can be smaller than your grid.</td>
</tr>
<tr>
<td>trkrinfo%southbd</td>
<td>For genesis runs, the southern boundary for searching for new storms. Does not need to match the boundaries of your grid, it can be smaller than your grid.</td>
</tr>
<tr>
<td>trkrinfo%type</td>
<td>trkrinfo%type defines the type of tracking to do. A 'tracker' run functions as the standard TC tracker and tracks only storms from the TC vitals. 'tgen' and 'midlat' run in genesis mode and will look for new storms in addition to tracking from TC vitals. 'tgen' will look for all parameters at the various vertical levels, while 'midlat' will only look for mslp. For HWRF, choose 'tracker'</td>
</tr>
<tr>
<td>trkrinfo%mslpt</td>
<td>Threshold for the minimum MSLP gradient (units are in mb/km) that must be met in order to continue tracking</td>
</tr>
<tr>
<td>trkrinfo%v850t</td>
<td>Threshold for the minimum azimuthally-average 850 mb cyclonic tangential wind speed (m/s) that must be exceeded in order to keep tracking</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>trkrinfo%gridtype</td>
<td>'global' or 'regional', this defines the type of domain grid. For HWRF choose 'regional'</td>
</tr>
<tr>
<td>trkrinfo%contin</td>
<td>This specifies the interval (in Pa) used by subroutine check_closed_contour to check for a closed contour in the mslp field when running in genesis mode. Note that check_closed_contour is also called from the routine that checks for a warm core, but the contour interval is hard-wired in the executable as 1.0 degree K for that usage.</td>
</tr>
<tr>
<td>trkrinfo%out_v</td>
<td>This is only set to 'y' if you are running in genesis mode, and it tells the tracker to write out a &quot;tc vitals&quot; record for any storms that it finds at tau = 00h in a forecast. For HWRF, choose ‘n’.</td>
</tr>
<tr>
<td>phaseflag</td>
<td>'y' or 'n', tells the program whether or not to determine the cyclone thermodynamic phase</td>
</tr>
<tr>
<td></td>
<td>Choose 'n' for HWRF</td>
</tr>
<tr>
<td>phasescheme</td>
<td>'cps', 'vtt', 'both', tells the program which scheme to use for checking the cyclone phase. 'cps' is Hart's cyclone phase space, 'vtt' is a simple 300-500 mb warm core check based on Vitart, and 'both' tells the program to use both schemes. Not used if phaseflag='n'</td>
</tr>
<tr>
<td>structflag</td>
<td>'y' or 'n', tells the program whether or not to determine the cyclone thermodynamic structure. Choose 'n' for HWRF.</td>
</tr>
</tbody>
</table>
OUTPUT:

Two files are output: one in ATCF format (fort.64) and one in a modified ATCF format (fort.69). The differences between the two files are 1) ATCF format only supports forecast lead time n hours, while modified format supports lead time in fraction of hours, and 2) the ATCF format file contains records with lead times every 6 h, while the modified format contains records with the lead times listed in input file fort.15.

A sample of the vortex tracker output in modified ATCF format is listed below:

```
AL, 09, 2008090900, 03, HWRF, 00000, 215N,  802W,  68,  968, XX,  34, NEQ, 0081, 0053, 0048, 0065,  0,  0, 23
AL, 09, 2008090900, 03, HWRF, 00000, 215N,  802W,  68,  968, XX,  50, NEQ, 0032, 0026, 0026, 0041,  0,  0, 23
AL, 09, 2008090900, 03, HWRF, 00000, 215N,  802W,  68,  968, XX,  64, NEQ, 0000, 0000, 0000,  0,  0, 23
AL, 09, 2008090900, 03, HWRF, 00300, 218N,  810W,  75,  970, XX,  34, NEQ, 0185, 0124, 0051, 0111,  0,  0, 26
AL, 09, 2008090900, 03, HWRF, 00300, 218N,  810W,  75,  970, XX,  50, NEQ, 0044, 0045, 0023, 0027,  0,  0, 26
AL, 09, 2008090900, 03, HWRF, 00300, 218N,  810W,  75,  970, XX,  64, NEQ, 0034, 0000, 0000,  0,  0, 26
AL, 09, 2008090900, 03, HWRF, 00600, 219N,  816W,  70,  970, XX,  34, NEQ, 0193, 0118, 0068, 0129,  0,  0, 27
AL, 09, 2008090900, 03, HWRF, 00600, 219N,  816W,  70,  970, XX,  50, NEQ, 0045, 0045, 0035, 0045,  0,  0, 27
AL, 09, 2008090900, 03, HWRF, 00600, 219N,  816W,  70,  970, XX,  64, NEQ, 0034, 0027, 0000, 0000,  0,  0, 27
AL, 09, 2008090900, 03, HWRF, 00900, 220N,  821W,  66,  973, XX,  34, NEQ, 0198, 0120, 0072, 0133,  0,  0, 30
AL, 09, 2008090900, 03, HWRF, 00900, 220N,  821W,  66,  973, XX,  50, NEQ, 0051, 0052, 0027, 0039,  0,  0, 30
AL, 09, 2008090900, 03, HWRF, 00900, 220N,  821W,  66,  973, XX,  64, NEQ, 0031, 0000, 0000, 0030,  0,  0, 30
```

Column 1: basin name. "AL" represents Atlantic.
Column 2: ATCF storm ID number. Ike was the 9th storm in the Atlantic Basin in 2008.
Column 3: model starting time.
Column 4: constant and 03 simply indicates that this record contains model forecast data.
Column 5: model name.
Column 6: forecast lead time in hours multiplied by 100 (e.g., 00900 represents 9.00 h).
Column 7-8: vortex center position (latitude and longitude multiplied by 10).
Column 9: vortex maximum 10-m wind.
Column 10: vortex minimum MSLP.
Column 11: placeholder for character strings that indicate whether the storm is a
depression, tropical storm, hurricane, subtropical storm etc.

Column 12: thresholds wind speed in knots The first in those six columns is an identifier that indicates whether this record contains radii for the 34-, 50- or 64-knot wind thresholds.

Column 13: “NEQ” indicates that the four radii values that follow will begin in the northeast quadrant and progress clockwise.

Column 14-16: wind radii (in nm) for the threshold winds in each quadrants.

Column 17-18: not used.

Column 19: radius of maximum winds, in n mi.

**USAGE:**

```
hwrf_gettrk.exe < namelist
```

### 8.5 How to Plot the Tracker Output Using ATCF_PLOT

*atcf_plot* is a set of GrADS scripts that can be used to plot hurricane track files in ATCF format.

*atcf_plot* can be found in the directory: `${HOME}/HWRF/src/gfdl-vortextracker/trk_plot`

To use *atcf_plot* to plot the storm’s track:

- Enter the directory `${HOME}/HWRF/src/gfdl-vortextracker/trk_plot`
- Build the two executables ndate.x in the directory ndate.cd/, and nhour in the directory nhour.fd (see Chapter 2).
- Run gribmap on the GrADS ctl file plottrak.ctl. gribmap is a GrADS utility that maps what is in your ctl file with the binary data that it finds inside your actual GRIB data file. It creates a map (plottrak.ix) that points to the locations where the requested binary data starts for the different variables and levels.

Create the map file by using the command:

```
gribmap -v -i plottrak.ctl
```

You should see one line in the output that has "MATCH" in the string. Both the plottrack.ctl and the newly created plottrak.ix map file need to be in the same directory where you are running the script below.

- Edit the atcfplot.sh to set the following paths:
  1. *gradsv2*: path to the GrADS executable (for example, 
     `/automounts/local_mmm/grads-1.9b4/bin/gradsc`).
  2. *GADDIR*: path to the directory containing the supplemental font and map files in for GrADS (for example, `/automounts/local_mmm/grads-1.9b4/lib`).
  3. *scrdir*: path to the working directory (for example,
4. `plotdir`: path to the directory where the plot files will be created (for example, `/data1/class08/BASIC/Test/plottrak/tracks/`).
   - Edit `atcfplot.gs` to define the following paths:
     1. `rundir`: same as `scrdir` in `atcfplot.sh`.
     2. `netdir`: same as `plotdir` in `atcfplot.sh`.
   - Edit `get_mods.sh` to define the following paths:
     1. `rundir`: same as `scrdir` in `atcfplot.sh`.
     2. `netdir`: same as `plotdir` in `atcfplot.sh`.
     3. `ndate`: path to the executable `ndate.x`.
     4. `nhour`: path to the executable `nhour`.
   - Edit `get_verif.sh` to define the following paths:
     1. `rundir`: same as `scrdir` in `atcfplot.sh`.
     2. `netdir`: same as `plotdir` in `atcfplot.sh`.
     3. `ndate`: path to the executable `ndate.x`.
     4. `nhour`: path to the executable `nhour`.
   - After setting up the paths to the correct locations in your system, run the script using the command:

   ```
   atcfplot.sh ${YYYY} ${Basin}
   ```

   This will start a GUI window and read in ATCF format track files `a${Basin}${SID}${YYYY}.dat` in the `rundir` directory. The user needs to insert the gfdl-vortex tracker output into the files `a${Basin}${SID}${YYYY}.dat`.

   For example, the user can use the command “atcfplot.sh 2008 al” to plot the track files `aal${SID}2008.dat` in the `rundir` directory.

   When the GUI window appears, from the drop down menu, select a storm, start date, and a model name (“atcfname” in the gfdl vortex tracker namelist), then click the “Plot” button to plot the track. The plots can be exported to image files by using the “Main” and then “Print” menu options.

   If the atcfname in an ATCF format track file cannot be found on the GUI window, the user can use his atcfname to replace an unused one (search in function “modnames”) in the file `atcfplot.gs`. Note all three instances of the unused atcfname need to be replaced in `atcfplot.gs`. 