Community HWRF
USERS’ GUIDE
V3.3a

August 2011
updated for code release

The Developmental Testbed Center

Contributors to this Guide:

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

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.

Updates since the April 2011 version of this document:

1) The CO₂ transmission coefficients used in the HWRF radiation scheme now can be calculated at runtime. To do so, set co2tf = 1 in the physics section of namelist_main.input, namelist_ghost.input and namelist_analysis.input.

2) There are two domains that can be used for HWRF ocean component, POM-TC, when the storm is in the Eastern Atlantic basin. When the POM-TC code is compiled, executables for both Eastern Atlantic domains are created. By default, the scripts use the 2011 operational East Atlantic domain, which is the same as the 2010 one. To use an extended East Atlantic domain, set the “use_extended_eastatl” variable to “true” in the scripts wrf-utilities/scripts/wrf.ksh, pomtc/ocean_scripts/kickit03_phase3.sh and pomtc/ocean_scripts/kickit03_phase4.sh. This update does not affect POM-TC’s “united” domain.
Chapter 1: HWRF System Introduction ............................................ 7
  1.1 HWRF System Overview .................................................................................. 7
  1.2 HWRF Source Code Directory Structure ....................................................... 9
  1.3 Input Data ......................................................................................................... 13
  1.4 Production Directory ...................................................................................... 15

Chapter 2: Software Installation ......................................................... 16
  2.1 Introduction ........................................................................................................ 16
  2.2 Obtaining the HWRF Source Code ................................................................. 16
  2.3 Setting up the HWRF System .......................................................................... 17
  2.4 System Requirements, Libraries and Tools ....................................................... 18
    2.4.1 Compilers ..................................................................................................... 18
    2.4.2 netCDF and MPI .......................................................................................... 19
    2.4.3 LAPACK and BLAS .................................................................................... 20
    2.4.4 GRIB ............................................................................................................ 21
  2.5 Included Libraries ............................................................................................. 21
    2.5.1 Component Dependencies .......................................................................... 22
  2.6 Building WRF-NMM ....................................................................................... 22
    2.6.1 Configuring WRF-NMM ............................................................................. 23
    2.6.2 Compiling WRF-NMM ............................................................................... 24
  2.7 Building HWRF-Utilities ................................................................................. 25
    2.7.1 Set Environment Variables ......................................................................... 25
    2.7.2 Configure and Compile .............................................................................. 26
  2.8 Building POM-TC ............................................................................................ 28
    2.8.1 Set Environment Variables ......................................................................... 28
    2.8.2 Configure and Compile .............................................................................. 29
Chapter 3: HWRF Preprocessing System .......................... 39

3.1 Introduction .................................................................................................................. 39
3.2 How to Run the HWRF Preprocessing Using Scripts .................................................. 39
   3.2.1 hwrfdomain.ksh ................................................................................................. 39
   3.2.2 geogrid.ksh ....................................................................................................... 41
   3.2.3 ungrid.ksh .......................................................................................................... 42
   3.2.4 metgrid.ksh .......................................................................................................... 42
3.3 Executables ...................................................................................................................... 43
   3.3.1 geogrid.exe ......................................................................................................... 43
   3.3.2 ungrid.exe ......................................................................................................... 44
   3.3.3 metgrid.exe ......................................................................................................... 44
3.4 Algorithm to Define the HWRF Domain Using the Storm Center Location .... 45
3.5 HWRF Domain Wizard .................................................................................................. 46

Chapter 4: Vortex Initialization .................................................. 47

4.1 Overview ......................................................................................................................... 47
4.2 Domain Grids Used in Vortex Initialization ................................................................. 48
4.3 How to Run the Vortex Initialization Using Scripts ...................................................... 49
   4.3.1 real.ksh .................................................................................................................. 49
   4.3.2 wrf.ksh in analysis mode ..................................................................................... 50
Chapter 5: Ocean Initialization of POM-TC ................. 75

5.1 Introduction ................................................. 75
5.2 Scripts ......................................................... 75
5.3 Procedures in the five "kickit" Scripts .......................... 76
   5.3.1 kickit00_region.sh .................................. 76
   5.3.2 kickit01_sharpn.sh .................................. 76
   5.3.3 kickit02_getsst.sh .................................. 77
   5.3.4 kickit03_phase3.sh .................................. 77

4.3.3 wrf.ksh in ghost mode .................................... 51
4.3.4 relocate_stage1.ksh ...................................... 52
4.3.5 relocate_stage2.ksh ...................................... 53
4.3.6 relocate_stage3.ksh ...................................... 54
4.3.7 run_gsi.ksh .............................................. 56
4.3.8 merge.ksh .................................................. 57

4.4 HWRF Vortex Initialization Executables ........................ 65
   4.4.1 diffwrf_3dvar.exe .................................... 65
   4.4.2 hwrf_create_nest_1x_10m.exe ............................ 65
   4.4.3 hwrf_inter_2to6.exe ................................... 66
   4.4.4 hwrf_pert_ct.exe ...................................... 66
   4.4.5 hwrf_anl_4x_step2.exe ................................ 67
   4.4.6 hwrf_inter_4to2.exe ................................... 67
   4.4.7 hwrf_split.exe ......................................... 68
   4.4.8 hwrf_anl_bogus_10m.exe ................................ 68
   4.4.9 hwrf_create_trak_guess.exe .............................. 69
   4.4.10 hwrf_inter_4to6.exe ................................... 69
   4.4.11 hwrf_anl_cs_10m.exe .................................. 70
   4.4.12 hwrf_merge_nest_4x_10m2.exe ......................... 71
   4.4.13 hwrf_merge_nest_4x_step1.exe ......................... 71
   4.4.14 hwrf_inter_2to1.exe ................................... 72
   4.4.15 hwrf_merge_nest_4x_step2.exe ......................... 72
   4.4.16 hwrf_data_remv.exe .................................. 72
   4.4.17 ssrc.exe ............................................... 73
   4.4.18 gsi.exe .................................................. 73
Chapter 6: How to Run HWRF ........................................ 81

6.1 Introduction .................................................................. 81
6.2 How to Run HWRF Using the Script wrf.ksh .......................... 81
6.3 Overview of the Script .................................................. 83
6.4 Output Files in the Directory .......................................... 85
6.5 Status Check ................................................................ 85
6.6 Executables .................................................................. 85
   6.6.1 wrf.exe ................................................................ 85
   6.6.2 hwrf_wm3c.exe .................................................. 86
   6.6.3 hwrf_ocean_united.exe ......................................... 86
   6.6.4 hwrf_ocean_eastatl.exe ......................................... 87
   6.6.5 hwrf_swcorner_dynamic.exe ................................... 88

Chapter 7: HWRF Post Processor ................................. 88

7.1 Introduction .................................................................. 89
7.2 Script .......................................................................... 89
7.3 Overview of the Script .................................................. 90
7.4 Executables .................................................................. 91
   7.4.1 unipost.exe ....................................................... 91
   7.4.2 copygb.exe ........................................................ 92

Chapter 8: GFDL Vortex Tracker ................................. 93

8.1 Introduction .................................................................. 93
8.2 How to Run the GFDL Vortex Tracker Using Script .................. 93
8.3 Overview of the script ................................................... 94
8.4 Executables .................................................................. 95
   8.4.1 hwrf_gettrk.exe .................................................. 95
8.5 How to Plot the Tracker Output Using ATCF_PLOT .................. 99
Chapter 1: HWRF System Introduction

1.1 HWRF System Overview

The Weather Research and Forecast (WRF) system for hurricane prediction (HWRF) was implemented at the National Weather Service (NWS)/National Centers for Environmental Prediction (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 the Nonhydrostatic Mesoscale Model (NMM) dynamic core of WRF (WRF-NMM) with a parent and nest domain coded in the WRF framework. The grid projection of the model is rotated latitude-longitude with E-staggering. The model domain spans about 80° in the latitudinal direction and extends about 95° in longitudinal direction to the north and about 70° to 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 National Hurricane Center (NHC) 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 spacing is 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 the Geophysical Fluid Dynamics Laboratory (GFDL) hurricane model, which includes a simplified Arakawa-Schubert scheme for cumulus parameterization and a 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 Global Forecast System (GFS) analysis is used to generate initial conditions for the hurricane model. This analysis is modified by the removal of the GFS vortex and the insertion of a vortex extracted from the 6-h forecast of the HWRF model initialized 6-h previously. This vortex is relocated and modified so that the initial storm position, structure and intensity conform to the NHC storm message. When the previous 6-h forecast is not available, a bogus vortex based on theoretical considerations and HWRF climatology is used. The analysis is then further modified using observations and a 3D-VAR data assimilation system. The GFS forecasted fields every 6 hours are used to provide lateral boundary conditions during each forecast.

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 (Janjic 1990).

HWRF can only be used for the two basins for which the national hurricane center is responsible: North Atlantic and North East Pacific. On the North East Pacific, HWRF is run in atmospheric mode only. In the North Atlantic, the atmospheric model is coupled with a version of the Princeton Ocean Model (POM-TC). POM was developed at Princeton University. At the University of Rhode Island (URI), the 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 features-based data assimilation techniques, in which the loop current, warm/cold eddy cores and the Gulf Stream are well represented in the ocean initial conditions. 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 supporting the community through the Developmental Testbed Center (DTC) includes the following three main components:

- HWRF atmospheric components
  - WRF-NMM V3.3 (which has tropical physics schemes and a vortex following moving nest)
  - WRF Preprocessing System (WPS)
  - Vortex Initialization
  - Gridpoint Statistical Interpolation (GSI)
  - Post-processing
  - Geophysical Fluid Dynamics Laboratory (GFDL) vortex tracker
• **HWRF oceanic components**
  - POM-TC model
  - Ocean initialization

• **Atmosphere-Ocean Coupler**

The atmospheric and oceanic components are interactively coupled with a Message Passing Interface (MPI)-based coupler, which was developed at 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 is 9 minutes.

**HWRF development and support**

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

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. Information about obtaining the codes, datasets and documentations can be found at [http://www.dtcenter.org/HurrWRF/users](http://www.dtcenter.org/HurrWRF/users) and in the following sections of this User’s Guide. Direct all questions to wrfhelp@ucar.edu.

**1.2 HWRF Source Code Directory Structure**

The HWRF system source code has the following eight components:

• WRF Atmospheric Model
• WRF Preprocessing System (WPS)
• Unified Post Processor (UPP)
• Gridpoint Statistical Interpolation (GSI)
• HWRF Utilities
• Tropical Cyclone Princeton Ocean Model (POM-TC)
• GFDL Vortex Tracker
• NCEP Atmosphere-Ocean Coupler

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

• WRFV3.tar.gz
• WPSV3.tar.gz
• UPPV1.tar.gz
• comGSI_v2.5.tar.gz
• HWRF-UTILITIES.tar.gz
• POMTC.tar.gz
• GFDL-VORTEXTRACKER.tar.gz
• NCEP-COUPLER.tar.gz

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

• WRFV3 – Weather Research and Forecasting model
• WPS – WRF preprocessor
• UPPV1 – Unified Post-Processor
• comGSI_v2.5 – Gridpoint statistical interpolation 3D-VAR data assimilation
• hwrf-utilities – Vortex initialization, utilities, tools, and supplemental libraries
• gfdl-vortextracker – Vortex tracker
• ncep-coupler – Ocean/atmosphere coupler
• pomtc – Tropical cyclone version of POM

For the remainder of this document, it will be assumed that the tar files have been expanded under $HOME/HWRF/src

The directory trees for these eight components are listed below:

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

<table>
<thead>
<tr>
<th></th>
<th>arch/</th>
<th>(compile options)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>clean</td>
<td>script to clean created files and executables</td>
</tr>
<tr>
<td></td>
<td>compile</td>
<td>script to compile the HWRF-Utilities code</td>
</tr>
<tr>
<td></td>
<td>configure</td>
<td>script to create the configure.hwrf file for compile</td>
</tr>
<tr>
<td></td>
<td>exec/</td>
<td>(executables)</td>
</tr>
<tr>
<td></td>
<td>libs/</td>
<td>(libraries including blas, sp, sfcio, bacio, w3 and bufr)</td>
</tr>
<tr>
<td></td>
<td>makefile</td>
<td>top level makefile</td>
</tr>
<tr>
<td></td>
<td>parm/</td>
<td>(WPS namelist; three WRF namelists; GSI namelist; hwrf_eta_micro_lookup.dat; all WRF lookup tables from</td>
</tr>
</tbody>
</table>
run subdirectory, some are modified for HWRF; and UPP namelist)
___scripts/ (scripts used to run HWRF system)
___tools/ (source code for tools to run HWRF system)
   __grbindext
   __hwrfd_data_remy
___vortex_init/ (source code for vortex initialization)
   __hwrfa_anl_bogus
   __hwrfd_create_trak_guess
   __hwrfd_set_ijstart
   __hwrfd_split
   __hwrfd_create_nest
   __hwrfd_anl_cs
   __hwrfd_bin_io
   __hwrfd_pert_ct
   __interpolate
   __merge_nest
   __hwrfd_create_trak_fnl
   __hwrfd_diffwrf_3dvar
   __hwrfd_guess
   __hwrfd_anl_step2

2. pomtc (POM-TC Ocean model)
___arch/ (compile options)
   ___clean script to clean created files and executables
   ___compile script to compile the pomtc code
   ___configure script to create the configure.pom file for compile
   ___makefile top level makefile
___ocean_exec/ (ocean model executables)
___ocean_init/ (source code for generating ocean model initial condition)
     ___eastatl
     ___united
     ___gfdl_find_region
     ___getsst
     ___sharp_mcs_rf_12m_rmy5
     ___date2day
     ___day2date
___ocean_main/ (source code for the ocean forecast model)
     ___ocean_united
     ___ocean_eastatl
___ocean_parm/ (namelists for ocean model)
___ocean_scripts/ (scripts used to generating ocean model initial condition)

3. ncep-coupler (NCEP Coupler)
4. gfdl-vortextacker (GFDL Vortex Tracker)

- arch/ (compile options)
- clean script to clean created files and executables
- compile script to compile the tracker code
- configure script to create the configure.trk file for compile
- makefile top level makefile
- trk_exec/ (GFDL vortex tracker executables)
- trk_plot/ (GFDL vortex tracker plot scripts and data)
- trk_src/ (GFDL vortex tracker source codes)

5. WRFV3 (Atmospheric model)

- Registry/ (WRFV3 Registry files)
- arch/ (compile options)
- chem/ (WRF-Chem, not used in HWRF)
- clean script to clean created files and executables
- compile script to compile the WRF code
- configure script to create the configure.wrf file for compile
- dyn_em/ (WRF-ARW dynamic modules, not used in HWRF)
- dyn_exp/ ('toy' dynamic core, not used by HWRF)
- dyn_nnm/ (WRF-NMM dynamic modules, used by HWRF)
- external/ (external packages including ocean coupler interface)
- frame/ (modules for WRF framework)
- inc/ (include files)
- main/ (WRF main routines, such as wrf.F)
- phys/ (physics modules)
- run/ (run directory, HWRF can be run in other directories)
- share/ (modules for WRF mediation layer and WRF I/O)
- test/ (sub-dirs where one can run specific configuration of WRF)
- tools/ (tools directory)
- var/ (WRF-Var)

See the WRF-NMM User's Guide for more information. The WRF-NMM User’s Guide is available at
6. WPSV3 (WRF Pre-processor)

| arch/ | (compile options) |
| clean | script to clean created files and executables |
| compile | script to compile the WPS code |
| configure | script to create the configure.wps file for compile |
| geogrid/ | (source code for geogrid.exe) |
| ungrid/ | (source code for ungrid.exe) |
| metgrid/ | (source code for metgrid.exe) |
| util/ | (utility programs for WPSV3) |

7. UPPV1 (Unified Post-Processor)

| arch/ | (compile options) |
| bin/ | (UPPV1 executables) |
| clean/ | script to clean created files and executables |
| compile/ | script to compile the UPPV1 code |
| configure/ | script to create the configure.upp file for compile |
| parm/ | (parameter files, which controls how the UPP is performed) |
| lib/ | (UPPV1 libraries) |
| src/ | (UPPV1 source codes) |
| scripts/ | (sample scripts running UPPV1) |

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 (TCVitals data)

2. fix
   - comGSI (for GSI)
   - geo_static (for WPS geogrid.exe)
   - ocean (for ocean initialization)
     gfdl_ocean_topo_and_mask.eastat
     gfdl_Hdeepgsu.eastatl
     gfdl_gdem.[00-13].ascii
     gfdl_ocean_readu.dat.[01-12]
     gfdl_ocean_spinup_gdem3.dat.[01-12]
     gfdl_ocean_spinup_gspath.[01-12]
     gfdl_ocean_spinup.BAYuf
3. **GFS_gridded** (GFS gridded data, for WRF initialization)

`GFS_gridded` has subdirectories `${YYYYMMDDHH}`, which in turn, contain GFS gridded data `gfs.${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 `${YYYYMMDDHH}`, which in turn, contain GFS spectral data `gfs.${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` contain the loop current and warm-core ring information for August 28 2008.

6. **obs** (observational data for GSI in prepBUFR and BUFR formats)

`obs` has subdirectories `${YYYYMMDDHH}`, which in turn, contain the following data that will be used in GSI:

`gfs.${YYYYMMDDHH}.prepbufr`
`gfs.${YYYYMMDDHH}.satang`
`gfs.${YYYYMMDDHH}.abias`
`gfs.${YYYYMMDDHH}.1bhrs4.tm00.bufr_d`
`gfs.${YYYYMMDDHH}.1bmhs.tm00.bufr_d`
`gfs.${YYYYMMDDHH}.airsev.tm00.bufr_d`
`gfs.${YYYYMMDDHH}.geosfv.tm00.bufr_d`
`gfs.${YYYYMMDDHH}.1bamua.tm00.bufr_d`
`gfs.${YYYYMMDDHH}.1bamub.tm00.bufr_d`
`gfs.${YYYYMMDDHH}.1bhrs3.tm00.bufr_d`
${YYYYMMDDHH}$ is the date for the observations.

## 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: ${USER}/HWRF/SID/${yyyymmddhh} (where SID is storm ID, e.g., 07L, and $\{yyyymmddhh\}$ 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 geogrid.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. wrfanalysesprd (created by wrf.ksh run in “analysis” mode)
8. relocatprd (created by vortex initialization scripts)
9. gsiprd (created by GSI scripts, if GSI is run)
10. mergeprd (created by merge.ksh)
11. oceanprd (created by ocean initialization scripts)
    - sharpn (run the sharpening program)
    - getsst (extract the SST from GFS)
    - phase3 (run Phase 3)
    - phase4 (run Phase 4)
12. wrfprd (created by wrf.ksh in “main” mode)
13. postprd (created by run_unipost)
14. gvtprd (created by tracker.ksh)
Chapter 2: Software Installation

2.1 Introduction

The DTC community HWRF system, which is based on the NOAA operational WRF hurricane system (HWRF), has eight components:

- WRF Atmospheric Model
- WRF Preprocessing System (WPS)
- Unified Post Processor (UPP)
- Gridpoint Statistical Interpolation (GSI)
- HWRF Utilities
- Tropical Cyclone Princeton Ocean Model (POM-TC)
- GFDL Vortex Tracker
- NCEP Atmosphere-Ocean Coupler

Each of these components is available from the DTC as community software. The first three of these components are the traditional WRF components: WRF, WPS, and UPP (the replacement for WPP). GSI is a 3D variational data assimilation code used for hurricane initialization. The remaining four components are specific to the hurricane system, and as such are referred to as the hurricane components of the HWRF system.

This chapter discusses how to build the HWRF system. It starts in Section 2.2 by discussing where to find the source code. Section 2.3 covers the preferred directory structure and how to unpack the tar files. Section 2.4 covers the system requirements for building and running the components. Section 2.5 discusses the libraries included in the HWRF-Utilities component. Section 2.6 covers building WRF-NMM for HWRF. The remaining sections are devoted to building each of the remaining components of the HWRF system.

2.2 Obtaining the HWRF Source Code
The HWRF hurricane system has eight components. All of these are available from the HWRF website. While most of these codes are also available from other community websites, the versions needed for HWRF should be acquired from the DTC HWRF website to ensure they are a consistent set. For some components, the code is hosted in the HWRF website itself while, for other components, users will be redirected to external websites.

In any case, all components can be obtained through the HWRF website

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

by selecting the Downloads and HWRF System tabs on the left vertical menu. New users must first register before downloading the source code. Returning users need only provide their registration email address. A successful download produces eight tar files, HWRF-UTILITIES.tar.gz, POMTC.tar.gz, GFDL-VORTEXTRACKER.tar.gz, NCEP-COUPLER.tar.gz, WRFV3.3a.tar.gz, WPSV3.3.TAR, UPPV1.tar.gz, and comGSI_v2.5.tar.gz.

After downloading each of the component codes, the user should check the links to known issues and bug fixes (which ever of the two exists for a particular code), to see if any code updates are required. You now have all the HWRF system components as gzipped tar files. The next section describes how to organize them.

2.3 Setting up the HWRF System

The HWRF run scripts provided by the DTC assume that the HWRF components have a specific directory structure. While it is possible to customize the directory structure, doing so will necessitate editing paths in all of the provided run scripts, The run scripts assume that the component code directories are all located in the directory ${HOME}/HWRF/src. Before unpacking the tar files you just downloaded, create the working directory ${HOME}/HWRF/src and move the tar files into it. You may use the UNIX commands:

```
mkdir -p ${HOME}/HWRF/src
mv *.gz ${HOME}/HWRF/src
cd ${HOME}/HWRF/src
```

The tar files can be unpacked by use of the GNU command, gunzip

```
gunzip *.tar.gz
```

and the tar files extracted by running tar -xvf individually on each of the tar files.

Once unpacked, there should be the eight source directories:
We will now discuss how to build the components.

2.4 System Requirements, Libraries and Tools

In practical terms, the HWRF system consists of a collection of shell scripts, which run a sequence of serial and parallel executables. The source code for these executables is in the form of programs written in FORTRAN, FORTRAN 90, and C. In addition, the parallel executables require some flavor of MPI for the distributed memory parallelism, and the I/O relies on the netCDF I/O libraries. Beyond standard shell scripts, the build system relies on use of the Perl scripting language and GNU Make and date.

The basic requirements for building and running the HWRF system are the following:

- FORTRAN 90/95 compiler
- C compiler
- MPI v1.2+
- Perl
- netCDF V3.6+ (V3 series only)
- LAPACK and BLAS
- GRIB1/2

Because these tools and libraries are typically the purview of system administrators to install and maintain, they are lumped together here as part of the basic system requirements.

2.4.1 Compilers

The DTC community HWRF system successfully builds and runs on IBM AIX and Linux platforms. Specifically the following compiler/OS combinations are supported:

- IBM with xlf Fortran compiler
- Linux (both 32 and 64-bit) with
  - PGI pgf90
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). Unforeseen build issues may occur when using an older compiler version. As always, the best results come from using the most recent version of compilers.

2.4.2 netCDF and MPI

The HWRF system requires a number of support libraries not included with the source code. Many of these libraries may be part of the compiler installation, and are subsequently referred to as system libraries. For our needs, the most important of these libraries are netCDF and MPI.

An exception to the rule of using the most recent version of code, libraries, and compilers is the netCDF library. The HWRF system I/O requires the most recent V3 series of the library. Version 4 of netCDF diverges significantly from version 3, and is not supported. The preferred version of the library is netCDF V3.6+. The netCDF libraries can be downloaded from the Unidata website:

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 system libraries, such as netCDF, be built with the same FORTRAN compiler, compiler version, and compatible flags, as used to compile the remainder of 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.

Many default Linux installations include a version of netCDF. Typically this version is only compatible with code compiled using gcc. To build the HWRF system, a version of the library must be built using your preferred compiler and with both C and FORTRAN bindings. If you have any doubts about your installation, ask your system administrator.
Building and running the HWRF distributed memory parallel executables 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 use the same word size option flags, as the remainder of the source code.

Installing MPI on a system is typically a job for the system administrator and will not be addressed here. If you are running HWRF on a computer at a large center, check the machines’ documentation before you ask the local system administrator. On Linux systems, you can typically determine whether MPI is available; try running the following UNIX commands:

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

If any of these tests return with *Command Not Found*, there may be a problem with your MPI installation. Contact your system administrator for help if you have any questions.

### 2.4.3 LAPACK and BLAS

The LAPACK and BLAS are open source mathematics libraries for the solution of linear algebra problems. The source code for these libraries is freely available to download from NETLIB at

[http://www.netlib.org/lapack/](http://www.netlib.org/lapack/)  

Most commercial compilers provide their own optimized versions of these routines. These optimized versions of BLAS and LAPACK provide superior performance to the open source versions.

On the IBM machines, the AIX compiler is often, but not always, installed with the Engineering and Scientific Subroutine Libraries or ESSL. In part, the ESSL libraries are highly optimized parallel versions of many of the LAPACK and BLAS routines. The ESSL libraries provide all of the linear algebra library routines needed by the HWRF system.

On Linux systems, HWRF supports both the Intel ifort and PGI pgf90 compilers. The Intel compiler has its own optimized version of the BLAS and LAPACK routines called the Math Kernel Library or MKL. The MKL libraries provide most of the LAPACK and BLAS routines needed by the HWRF system. The PGI compiler typically comes with its own version of the BLAS and LAPACK libraries. Again the PGI version of BLAS and LAPACK contain most of the routines needed by HWRF. For PGI these libraries are loaded automatically. Since the vender versions of the libraries are often incomplete, a
copy of the full BLAS library is provided with the HWRF-Utilities component. The build system will link to this version last.

Should your compiler come without any vendor supplied linear algebra libraries, it will be necessary to download and build your own local version of the BLAS and LAPACK libraries.

2.4.4 GRIB

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 website:

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

2.5 Included Libraries

For convenience in building HWRF-Utilities, the POM-TC, and the GFDL Vortex Tracker components, the HWRF-Utilities component includes a number of libraries in the hwrf-utilities/libs/src/ directory. These libraries are built automatically when the HWRF-Utilities component is built. The included libraries include:

- BACIO
- BLAS
- BUFR
- SFCIO
- SIGIO
- SP
- W3

The other components, WPS, WRF, UPP and GSI, come with their own versions of many of these libraries, but typically they have been customized for that particular component and should not be used by other components.

When the HWRF-Utilities component is compiled, it starts by first building all the included libraries. The vortex initialization code contained in the HWRF-Utilities component requires all of the above libraries except for the SFCIO library. In addition it requires both the BLAS and LAPACK mathematical libraries when the IBM ESSL library is not included with the compiler installation.

The POMTC component requires the SFCIO, SP and W3 libraries. In addition, the local copy of the BLAS library is required when the ESSL library is not included with the compiler installation. This is because the vendor supplied versions of BLAS are typically
incomplete, and the local version supplements the vendor version. Typically this is for any system other than the IBM. The GFDL vortex tracker component requires the BACIO and W3 libraries. The NCEP-Coupler does not require any additional libraries.

2.5.1 Component Dependencies

The eight components of the HWRF system have certain inter-dependencies. Many of the components depend on libraries produced by other components. For example, four of the components, WPS, UPP, GSI and the HWRF-Utilities, require linking to the WRF I/O API libraries to build. Since these I/O libraries are created as part of the WRF build, the WRF component must be built first. Once WRF is built, WPS, UPP, GSI, or the HWRF-Utilities can be built next. Since building the HWRF-Utilities produces the supplemental libraries needed by POM-TC and by the GFDL Vortex Tracker, 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
  - UPP
  - GSI
  - HWRF Utilities
    - POM-TC (BLAS on Linux, sfcio, sp, w3)
    - GFDL vortex tracker (w3 & bacio)
- NCEP Coupler

2.6 Building WRF-NMM

The WRF code has a fairly sophisticated build mechanism. The package attempts to determine the machine where the code is being built, and then presents the user with supported build options on that platform. 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 type of parallelism (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
2.6.1 Configuring WRF-NMM

To correctly configure WRF-NMM for the HWRF system, set the following additional environment variables beyond what WRF typically requires:

In C-Shell use the commands:

```c
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
```

These settings produce a version of WRF-NMM compatible with the HWRF system. Note that setting the environment variable `WRF_NMM_NEST` to 1 does not preclude running with a single domain.

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

```
./configure
```

You will be presented with a list of build choices for your computer. These choices may include multiple compilers and parallelism options.

The choices for the IBM architecture are

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, select option 3 for distributed memory parallelism (dmpar).

For Linux architectures, there are 26 options. Only the first twelve are compatible with the HWRF system. They are:

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, PGI accelerator compiler with gcc (serial)
At this time, only the distributed memory (dmpar) builds are recommended for the HWRF system on Linux environments. Therefore, depending on your choice of compiler and computing hardware, only options 3, 7, or 11 are recommended.

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

### 2.6.2 Compiling WRF-NMM

To build the WRF-NMM component enter the command

```
./compile nmm_real
```

It is generally advisable to save the standard out and error to a log file for reference. In `csh/tcsh` this can be done with the command:

```
./compile nmm_real |& tee build.log
```

which sends the standard out and error to both the file `build.log` and to the screen.

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:

```
./clean
```

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

```
./clean -a
```

A complete clean is strongly recommended if the compilation failed, the Registry has been changed, or 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

Further details on the HWRF atmospheric model, physics options, and running the model can be found in the Running HWRF chapter of the Users’ Guide.

Complete details on building and running the WRF-NMM model are available in the WRF-NMM User’s Guide, which is available from the link


2.7 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 the GFDL Vortex Tracker, include the BACIO, BLAS, BUFR, SIGIO, SFCIO, SP and W3 libraries. In addition to the libraries, this component includes the source for the vortex initialization routines and software tools such as the grbindex.

2.7.1 Set Environment Variables

The HWRF utilities build requires that two path variables NETCDF and WRF_DIR be set to the appropriate paths. The netCDF library path NETCDF is required for building the WRF-NMM component, and its value should be appropriately set if that component compiled successfully. The WRF_DIR path variable should point to the WRF directory compiled in the previous section. You must first build WRF before compiling any of the other components.

If you have followed the directory structure suggested in section 2.3, the WRF_DIR path should be set to ${HOME}/HWRF/src/WRFV3. In csh/tcsh, the variables may be set with:

setenv NETCDF /absolute_path_to_appropriate_netCDF_library/
setenv WRF_DIR ${HOME}/HWRF/src/WRFV3

In bash/ksh, the variables are set with:

export NETCDF=/absolute_path_to_appropriate_netCDF_library/
export WRF_DIR=${HOME}/HWRF/src/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.
2.7.2 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 directory. It concludes by asking the user to choose a configuration supported by current machine architecture.

For Linux three options are available:

1. Linux x86_64, PGI compiler w/LAPACK (dmpar)
2. Linux x86_64, Intel compiler w/MKL (dmpar)
3. Linux x86_64, Intel compiler w/LAPACK (dmpar)

Pick option 2 if your Intel compiler includes the MKL libraries, and option 3 if it does not.

For the IBM, only one choice is available:

1. AIX (dmpar)

If successful, the configure script creates a file called `configure.hwrf` in the *hwrf-utilities* directory. This file contains compilation options, rules, and paths specific to the current machine architecture, and can be edited to change compilation options, if desired.

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 20 executables in the directory `exec/`,

```
diffwrf_3dvar.exe
grbindex.exe
hwrf_anl_4x_step2.exe
hwrf_anl_bogus_10m.exe
hwrf_anl_cs_10m.exe
hwrf_create_nest_1x_10m.exe
hwrf_create_trak_fnl.exe
hwrf_create_trak_guess.exe
hwrf_data_remv.exe
hwrf_guess.exe
hwrf_inter_4to6.exe
hwrf_inter_4to2.exe
hwrf_inter_2to6.exe
hwrf_inter_2to1.exe
hwrf_merge_nest_4x_10m2.exe
hwrf_merge_nest_4x_step1.exe
hwrf_merge_nest_4x_step2.exe
hwrf_pert_ct.exe
hwrf_split.exe
hwrf_swcorner_dynamic.exe
```

and ten libraries in the directory `libs/`

```
libbacio.a - BACIO library
libblas.a - BLAS library
libbufr_i4r4.a - BUFR library built with -i4 -r4 flags
libbufr_i4r8.a - BUFR library built with -i4 -r8 flags
libsfcio_i4r4.a - SFCIO library built with -i4 -r4 flags
libsfcio_i4r8.a - SFCIO library built with -i4 -r8 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 used 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.

The HWRF-Utilities can be compiled to produce the libraries only, by typing

```
./compile library
```
This is useful for users that do not intend to use the entire HWRF system, but just need the libraries to build the tracker.

2.8 Building POM-TC

2.8.1 Set Environment Variables

The Tropical Cyclone version of the 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 has been built successfully. You must first build them before building POM-TC.

Again, assuming the directory structure proposed in section 2.3, for csh/tcsh, the first three library paths can be set with the commands:

```
setenv LIB_W3_PATH ${HOME}/HWRF/src/hwrf-utilities/libs/
setenv LIB_SP_PATH ${HOME}/HWRF/src/hwrf-utilities/libs/
setenv LIB_SFCIO_PATH ${HOME}/HWRF/src/hwrf-utilities/libs/
```

In bash/ksh, the first three library paths can be set with the commands:

```
export LIB_W3_PATH=${HOME}/HWRF/src/hwrf-utilities/libs/
export LIB_SP_PATH=${HOME}/HWRF/src/hwrf-utilities/libs/
export LIB_SFCIO_PATH=${HOME}/HWRF/src/hwrf-utilities/libs/
```

In addition to the three previous libraries, POM-TC requires routines from the linear algebra mathematical library. When building POM-TC in an IBM platform, the build will automatically use the ESSL library. When building POM-TC in a platform without ESSL (such as Linux), the build system uses the BLAS mathematical library provided with the `hwrf-utilities` component. In such a case, the fourth and final path must be set to:

```
setenv LIB_BLAS_PATH ${HOME}/HWRF/src/hwrf-utilities/libs/
```

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

```
export LIB_BLAS_PATH=${HOME}/HWRF/src/hwrf-utilities/libs/
```
2.8.2 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:

1. *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.

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, nine executables are created in `ocean_exec/`:

- `gfdl_date2day.exe`
- `gfdl_day2date.exe`
- `gfdl_find_region.exe`
- `gfdl_getsst.exe`
- `gfdl_ocean_eastatl.exe`
- `gfdl_ocean_united.exe`
- `gfdl_sharp_mcs_rf_l2m_rmy5.exe`
- `hwrf_ocean_eastatl.exe`
- `hwrf_ocean_united.exe`

The executables `hwrf_ocean_united.exe` and `hwrf_ocean_eastatl.exe`, are the ocean model executables used during the coupled atmosphere-ocean model run. The remaining executables are used for the ocean initialization.

### 2.9 Building GFDL Vortex Tracker

#### 2.9.1 Set 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. You must first build the HWRF utilities before building the vortex tracker.

Again, assuming that the directory structure proposed in section 2.3, for csh/tcsh, the library paths to be set are:

```
setenv LIB_W3_PATH ${HOME}/HWRF/src/hwrf-utilities/libs/
setenv LIB_BACIO_PATH ${HOME}/HWRF/src/hwrf-utilities/libs/
```

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

```
export LIB_W3_PATH=${HOME}/HWRF/src/hwrf-utilities/libs/
export LIB_BACIO_PATH=${HOME}/HWRF/src/hwrf-utilities/libs/
```

#### 2.9.2 Configure and Compile

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

1. AIX xlf 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.

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

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

To get help about compilation, type:

```
./compile -h
```

To remove all object files, 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/.

### 2.10 Building NCEP Coupler
2.10.1 Configure and Compile

To configure the NCEP Coupler for compilation, from within the ncep-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:

1. *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.

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_wm3c.exe` in the `cpl_exec/` directory.
2.11 Building WPS

2.11.1 Background

The WRF WPS requires the same build environment as the WRF-NMM model, including the netCDF libraries and MPI libraries. Since the WPS makes direct calls to the WRF I/O API libraries included with the WRF 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. If support for GRIB 2 files is desired, the JASPER library is also needed.

Further details on using the WPS to create HWRF input data can be found in Chapter 3 of the HWRF Users’ 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

2.11.2 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 option 2 for GRIB 2 support with HWRF. 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

Choose option 2 for Intel and option 6 for PGI, for GRIB 2 support with HWRF. 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 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.

### 2.12 Building UPP

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

The UPP requires the same Fortran and C compilers used to build the WRF model. In
addition, UPP requires the netCDF library and the WRF I/O API libraries (the latter is
included with the WRF build).

The UPP build requires a number of support libraries (IP, SP, W3), which are provided
with the source code and are located in the \texttt{UPPV1/lib/} directory. These libraries are for
the UPP build only. They should not be confused with the libraries of the same name
located in the \texttt{hwrf-utilities/}libs\texttt{ directory.}

\subsection*{2.12.1 Set Environment Variables}

The UPP requires the WRF I/O API libraries to be successfully built. These are created
when the WRF model is built. If the WRF model has not yet been compiled, it must first
be built before compiling UPP.

Since the UPP build requires linking to the WRF-NMM I/O API libraries, it must be able
to find the WRF directory. The UPP build looks for WRF in two places. First it looks for
a directory called WRFV3 in the same directory level as the UPP directory. In this case
that would be $\{\text{HOME}\}/HWRF/src$. If it does not find it there, it then checks the path
variable \texttt{WRF\_DIR} for the path to WRF. The path variable \texttt{WRF\_DIR} must therefore be
set to the location of the WRFV3 root directory.

In addition to setting the path variable, building UPP for use with HWRF requires setting
the environment variable \texttt{HWRF}. This is the same variable set when building WRF-
NMM for HWRF.

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

\begin{verbatim}
    setenv HWRF 1
    setenv WRF\_DIR $\{\text{HOME}\}/HWRF/src/WRFV3/
\end{verbatim}

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

\begin{verbatim}
    export HWRF=1
    export WRF\_DIR=$\{\text{HOME}\}/HWRF/src/WRFV3/
\end{verbatim}

\subsection*{2.12.2 Configure and Compile}

UPP uses a build mechanism similar to that used by the WRF model. Type configure
to generate the UPP 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 are two options,

1. \texttt{AIX} (serial)
2. \texttt{AIX} (dmpar)

while for LINUX operating systems there are six,

1. \texttt{Linux x86\_64, PGI compiler} (serial)
2. \texttt{Linux x86\_64, PGI compiler} (dmpar)
3. \texttt{Linux x86\_64, Intel compiler} (serial)
4. \texttt{Linux x86\_64, Intel compiler} (dmpar)
5. \texttt{Linux x86\_64, gfortran compiler} (serial)
6. \texttt{Linux x86\_64, gfortran compiler} (dmpar)

Any of the first four options are compatible with the HWRF system. The configuration script will generate the configure file \texttt{configure.upp}. If necessary, the \texttt{configure.upp} file can be modified to change the default compile options and paths.

To compile UPP, enter the command:

\texttt{./compile} \& tee build.log

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

To remove all built files, as well as the \texttt{configure.upp}, type:

\texttt{./clean}

This is recommended if the compilation failed or if the source code has been changed.

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

\texttt{http://www.dtcenter.org/wrf-nmm/users/docs/overview.php}
2.13 Building GSI

2.13.1 Background

The community GSI requires the same build environment as the WRF-NMM model, including the netCDF, MPI, and LAPACK libraries. In addition, GSI makes direct calls to the WRF I/O API libraries included with the WRF model. Therefore the WRF model must be built prior to building the GSI.

Further details on using the GSI with HWRF can be found in later chapters of this HWRF Users’ Guide.

2.13.2 Configure and Compile

Building GSI for use with HWRF requires setting three environmental variables. The first, HWRF, indicates to turn on the HWRF options in the GSI build. This is the same flag set when building WRF-NMM for HWRF. The second is a path variable pointing to the root of the WRF build directory. The third is the variable LAPACK_PATH, which indicates the location of the LAPACK library on your system.

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

```bash
    setenv HWRF 1
    setenv WRF_DIR ${HOME}/HWRF/src/WRFV3/
```

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

```bash
    export HWRF=1
    export WRF_DIR=${HOME}/HWRF/src/WRFV3/
```

The additional environment variable LAPACK_PATH may be needed on some systems. Typically the environment variable LAPACK_PATH needs only be set on Linux systems without a vendor provided version of LAPACK. IBM systems usually have the ESSL library installed and therefore do not need the LAPACK. Likewise, the PGI compiler often comes with a vendor provided version of LAPACK that links automatically with the compiler. Problems with the vendor supplied LAPACK library are more likely to occur with the Intel compiler. While the Intel compilers typically have the MKL libraries installed, the ifort compiler does not automatically load the library. It is therefore necessary to set the LAPACK_PATH variable to the location of the MKL libraries when using the Intel compiler.
Supposing that the MKL library path is set to the environment variable MKL, then the LAPACK environment for csh/tcsh:

```
setenv LAPACK_PATH $MKL
```

for bash/ksh

```
export LAPACK_PATH=$MKL
```

To build GSI for HWRF, change into the comGSI_v2.5 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 64-bit (dmpar, optimize)
2. AIX 32-bit (dmpar, optimize)

You may choose either option depending on your computer platform. On Linux computers, the listed options are:

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

After selecting the proper option, run the compile script:

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

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

```
./clean -a
```

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

Following the compile command, the GSI executable gsi.exe can be found in the run/ directory. If the executable is not found, check the compilation log file to determine what went wrong.

For details on using GSI with HWRF, see the GSI chapter in the HWRF Users Guide. For full details on the operation of GSI, see the DTC community GSI Users’ Guide.
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; ungrrib.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 ungrrib.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 TCVitals 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 TCVitals 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 TCVitals 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**

1. `storm.center`: file that contains the storm center latitude and longitude.
2. `domain.center`: file that contains the 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.
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 `namelist.wps` to replace the place-holders
4. Link to the `GEOGRID.TBL` file
5. Run `geogrid.exe`

**Output files in directory `${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 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. Link the input data files
5. Copy the WPS namelist to the working directory
6. Run ungrig.exe

**Output files in directory** `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 ungrib.exe” is found in the standard output file, `DOMAIN_HOME/ungribprd/ungrib.log`, the script ungrib.ksh was successfully finished.

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`
4. Insert path to WPS `METGRID.TBL` file into `namelist.wps`
5. Run `metgrid.exe`

**Output files in directory `${DOMAIN_HOME}/metgridprd`**
`met_nmm.d01.YYYY-MM-DD_HH:mm:ss.nc`. 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` has successfully finished.

### 3.3 Executables

**3.3.1 geogrid.exe**

**FUNCTION:**
interpolates static geographical data to the parent and nest grids.

**INPUT:**
Files in geographical static data directory:
(for example, /ptmp/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. 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 the 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 observed and predicted location of the storm to be simulated. Script hwrfdomain.ksh reads the TCVitals records and retrieves the storm center location. NHC and JTWC are the two agencies that provide the TCVitals - 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 TCVitals file). If a 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 15.0 ≤ STORM_LAT ≤ 25.0 then CENLA=STORM_LAT
   if 25.0 < STORM_LAT < 35.0 then CENLA=25.0
   if 35.0 ≤ STORM_LAT < 40.0 then CENLA=30.0
   if 40.0 ≤ STORM_LAT < 45.0 then CENLA=35.0
   if 45.0 ≤ STORM_LAT < 50.0 then CENLA=40.0
   if 50.0 ≤ STORM_LAT < 55.0 then CENLA=45.0
   if STORM_LAT ≥ 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 the storm center by at least 5
degrees, the following procedure is followed:
if CENLO > STORM_LON+5 then CENLO= STORM_LON + 5
if CENLO < STORM_LON- 5 then CENLO= STORM_LON - 5
Finally, the values of CENLA and CENLO are written to the namelist.wps as ref_lat and
ref_lon.

3.5 HWRF Domain Wizard

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

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.

Before the vortex initialization scripts are run, the user needs to run real.ksh to produce the initial and boundary conditions. Then wrf.ksh is run twice, once for the analysis domain, and once for the ghost domain (Figure 4.1). The two wrf.ksh runs are both 1-minute runs. The purposes of these two wrf.ksh runs are to obtain the initial fields on the analysis (wrfanl_d02) and ghost domains (wrfghost_d02), respectively. The two files are used in the subsequent vortex initialization and GSI procedures.

The HWRF vortex initialization process has three stages. First a check is performed to see if a previous 6-hr forecast exists and the storm’s maximum wind is larger than 12 ms\(^{-1}\). If so, this is a cycled run; otherwise it is a "cold start". For a cold start run, an axi-symmetric bogus vortex is used and adjusted. A cycled run will go through all the three stages, while a "cold start" run will go through stages 2 and 3 only:

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 and cycled 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.

4. After the 3 stages, the new vortex is created. Next, if the storm is deep, GSI (a 3D-VAR data assimilation system) is used to assimilate observational upper-level, surface and satellite radiance data into the HWRF atmospheric model’s initial condition background fields. The new vortex is added to the GFS environment fields from the GSI analysis in the observed location to replace the GFS vortex. When a storm is not deep, GSI is not run, and the new vortex is added to the original GFS environment fields in the observed location to replace the GFS vortex. The merged GFS environment fields and the new vortex are then 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 (Figure 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 and cycled 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. This domain is not used during the model integration. GSI is run twice, once for the parent domain, and once for the ghost domain.
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`

Note the executables called in scripts `real.ksh`, `wrf.ksh`, `run_gsi.ksh` are parallel codes, and if they need to be submitted with a batch system, the users should be responsible for understanding the batch system commands for the machine and infrastructure where the HWRF system is run. For the batch system commands for IBM/AIX (LSF) and Intel/Linux (SGE) systems, please see Chapter 6.2

4.3.1 `real.ksh`

Edit the script to set:
\textbf{WRF_ROOT:} path to WRFV3 \\
\textbf{SID:} storm ID \\
\textbf{NAMELIST:} WRF namelist \texttt{namelist\_main.input} \\
\textbf{DOMAIN\_HOME:} base path of the output directory. (for example, \\
/ptmp/$\{\text{USER}\}/HWRF/$\{\text{SID}\}/$\{\text{YYYYMMDDHH}\}). \\

Then run \texttt{real.ksh}. This will read in the output from the WPS executable \texttt{metgrid.exe} and generate \texttt{wrfinput\_d01} and \texttt{wrfbdy\_d01}. \texttt{namelist\_main.input} is the namelist used by \texttt{real\_nmm.exe} and \texttt{wrf.exe} to do the hurricane forecasts (see Figure 4.2. HWRF initialization procedures before vortex adjustments).

\textbf{Overview of script real.ksh:} \\
Run \texttt{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.

Note: to run \texttt{real.ksh} successfully, users should set the computer’s stacksize to be equal to or larger than 2GB. To do this:

In bash shell, use the command \texttt{ulimit –s 204800} \\
In C-shell, use the command \texttt{limit stacksize 2048m}

\textbf{Output files in directory $\{\text{DOMAIN\_HOME}\}/realprd:} \\
\texttt{wrfinput\_d01} (initial condition) \\
\texttt{wrfbdy\_d01} (boundary condition) \\
\texttt{fort.65} (high-resolution sea mask data)

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

\subsection*{4.3.2 \texttt{wrf.ksh} in analysis mode}

Edit the script to set:

\textbf{WRF\_ROOT:} path to WRFV3 \\
\textbf{SID:} storm ID \\
\textbf{WRF\_MODE:} choose 'analysis' in this step \\
\textbf{NAMELIST:} WRF namelist \texttt{namelist\_analysis.input} \\
\textbf{DOMAIN\_HOME:} base path of the output directory. (for example, \\
/ptmp/$\{\text{USER}\}/HWRF/$\{\text{SID}\}/$\{\text{YYYYMMDDHH}\}). \\
\textbf{WRF\_CORES:} number of processors \\
\textbf{DATA\_DIR:} path to fix ocean input data (not used when WRF\_MODE=analysis)
Edit the namelist $\{\text{NAMELIST}\}$ to set the starting and ending times of the ghost run. Note this is a one-minute run when $\{\text{WRF\_MODE}\}$=analysis.

Then run \textit{wrf.ksh}. This will make a one-minute run of \textit{wrf.exe} and generate an analysis output for the inner nest domain. \textit{namelist\_analysis.input} is used by \textit{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).

\textbf{Overview of script \textit{wrf.ksh}}

1. Run \textit{hwrf\_swcorner\_dynamic.exe} to calculate the istart and jstart values for the nest in the \textit{namelist\_analysis.input}. In \textit{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 \textit{wrf.exe} using \textit{namelist\_analysis.input}.

\textbf{Output files in the directory $\{\text{DOMAIN\_HOME}/wrf\{\text{WRF\_MODE}\}\text{prd}\}}$

"wrfanl\_d02\*", an "analysis" file for the HWRF inner domain.

\textbf{Status Check}

This step was successfully finished if the user finds “SUCCESS COMPLETE WRF” in files \textit{rsl.*}.

\subsection*{4.3.3 \textit{wrf.ksh} in ghost mode}

Edit the script to set:

\begin{itemize}
  \item \textit{WRF\_ROOT}: path to WRFV3
  \item \textit{SID}: storm ID
  \item \textit{WRF\_MODE}: choose 'ghost' in this step
  \item \textit{NAMELIST}: WRF namelist \textit{namelist\_ghost.input}
  \item \textit{DOMAIN\_HOME}: base path of the output directory.
    (for example, /ptmp/$\{\text{USER}\}/HWRF/$\{\text{SID}\}$/YYYYMMDDHH).
  \item \textit{WRF\_CORES}: number of processors
  \item \textit{DATA\_DIR}: path to fix ocean input data (not used when WRF\_MODE=ghost)
\end{itemize}

Edit the namelist $\{\text{NAMELIST}\}$ to set the starting and ending times of the ghost run. Note this is a one-minute run when $\{\text{WRF\_MODE}\}$=ghost.

Then run \textit{wrf.ksh}. This will make a one-minute run of \textit{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. File \textit{namelist\_ghost.input} is the namelist.
used by \textit{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).

\textbf{Overview of script \textit{wrf.ksh}}

1. Run \textit{hwrf_swcorner\_dynamic.exe} to calculate the istart and jstart values for the nest in the \textit{namelist\_ghost.input}. In \textit{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 \textit{wrf.exe} using \textit{namelist\_ghost.input}. 

Output files in the directory $\textit{DOMAIN\_HOME}$/wrf$\textit{WRF\_MODE}$/prd "ghost\_d02\*", an "analysis" file for the HWRF ghost domain.

\textbf{Status Check}

This step was successfully finished if the user finds “SUCCESS COMPLETE WRF” in files \textit{rsl.*}.

\textbf{4.3.4. relocate\_stage1.ksh}

Edit the script to set the environment variables:

\begin{itemize}
  \item \textit{WRF\_ROOT}: path to WRFV3
  \item \textit{SID}: storm ID
  \item \textit{NAMELIST}: WRF namelist \textit{namelist\_analysis.input}
  \item \textit{DOMAIN\_HOME}: base path of the output directory.
      (for example, 
      \texttt{/ptmp/$\{USER\}/HWRF/$\{SID\}/$\{YYYYMMDDHH\}}).
  \item \textit{CYCLE\_HOME}: path to the output directory containing previous 6-h forecast for a cycled run
\end{itemize}

Then run \textit{relocate\_stage1.ksh}. If a previous 6-h forecast exists (a cycled run), the forecast will be interpolated onto the 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 Figure 4.3). Note in the script \textit{relocate\_stage1.ksh}, a gnu version of the command “\texttt{date}” is used.

\textbf{Overview of script \textit{relocate\_stage1.ksh}}

1. Run \textit{diffwrf\_3dvar.exe} to convert existing \textit{wrfout\_d01} and \textit{wrfout\_d02} into unformatted data files \textit{old\_hwrf\_d01} and \textit{old\_hwrf\_d02} respectively.

2. Run \textit{merge\_nest\_4x\_step1.exe} to merge \textit{wrfout\_d01} and \textit{wrfout\_d02} onto 4X domain and produce a file containing the merged data: \textit{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 the 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.

### 4.3.5 relocate_stage2.ksh

Edit the script to set:

- `WRF_ROOT`: path to WRFV3
- `SID`: storm ID
- `NAMELIST`: WRF namelist `namelist_analysis.input`
- `DOMAIN_HOME`: base path of the output directory.
  (for example, `/ptmp/${USER}/HWRF/${SID}/${YYYYMMDDHH}`).

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.fnall_gfs`, a guess track file from

   `~/DOMAIN_HOME/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 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 environment data (`gfs_env`) and 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.

### 4.3.6 relocate_stage3.ksh

Edit the script to set:

- `WRF_ROOT`: path to WRFV3
- `SID`: storm ID
- `NAMELIST`: WRF namelist `namelist_analysis.input` (for example, `/ptmp/$USER/HWRF/$SID/$YYYYMMDDHH`).
- `DOMAIN_HOME`: base path of 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 or a cycled run) to match the observed storm location, intensity and structure. (see Figure 4.5 and Figure 4.6).

**Overview of script relocate_stage3.ksh:**

1. For cold start runs (previous cycle 6-h forecast does not exist or the observed storm’s maximum wind is less than 12 ms$^{-1}$): 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 (previous cycle 6-h forecast exists and the storm’s maximum wind is larger than 12 ms$^{-1}$):
   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 maximum wind speed of the combined vortex + environmental flow is less than the observed one, 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 the netCDF file `wrfinput_d01`.

7. Run `diffwrf_3dvar.exe` to convert the unformatted `data_merge_2x` to the netCDF file `wrfghost_d02`.

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

**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.
4.3.7 run_gsi.ksh

Edit the script to set:

GSIPROC: Number of processors used for GSI analysis
ARCH: System architecture. Supported configurations:
IBM_LSF, LINUX_Intel, LINUX_Intel_PBS,
LINUX_PGI, LINUX_PGI_PBS, DARWIN_PGI
(Note: HWRF currently do not support the configuration of
DARWIN_PGI, although community GSI does.)
DOMAIN: WRF domain (set to 1 for parent domain and 2 for ghost domain)
SID: ID of the storm being processed, used to define path names
NAMELIST: path to WRF namelist namelist_analysis.input
GSI_EXE: path and name of the gsi executable
HWRF_UTILITIES_ROOT: path to HWRF-Utilities
DATA_REMV: path and name of the hwrf_data_remv executable.
SSRC: path and name of the ssrc executable
FIX_ROOT: path of fix files
CYCLE_DATA: previous cycle data directory
WORK_ROOT: working directory, where GSI runs
OBS_ROOT: path of the directory where the observations files are located
PREPBUFR: prepBUFR conventional observation files
BK_FILE: path and name of background file
bk_core: which WRF core is used as background (NMM or ARW, for
HWRF, choose NMM)
bkcv_option: which background error covariance and parameter is used
(GLOBAL or NAM, for HWRF, choose NAM)

Then run the script run_gsi.ksh.

Note in HWRF, the GSI script run_gsi.ksh needs to run two times, one for the parent
domain (set DOMAIN=1), and the other for the ghost domain (set DOMAIN=2).

Overview of script run_gsi.ksh

1. Check if all the environment variables are set
2. Check the machine’s architecture to decide the run command
3. Check the endianness
4. Create a working directory
5. Reverse the endianness of data if using NCEP prepBUFR data (big_endian) on
   machines with little_endian (e.g. Linux). Copy prepBUFR data to work directory
6. Remove observational data near storm center
7. Copy fixed data to working directory
8. Create a namelist for GSI analysis
9. Run the executable gsi.exe
Output files in directory `\$\{DOMAIN_HOME\}/gsiprd:
stdout:
Standard text output file. It is the file most often used to check GSI analysis processes as it contains basic and important information about the analyses.

`wrf_inout`:
Analysis results. The format is same as the input background file.

**Status Check:**

If you see “PROGRAM GSI ANL HAS ENDED” in the file `stdout`, the script `run_gsi.ksh` has run successfully.

For more information on checking GSI output, refer to the GSI User’s Guide (http://www.dtcenter.org/com-GSI/users/docs/index.php)

### 4.3.8 `merge.ksh`

Edit the script to set:

- `WRF_ROOT`: path to WRFV3
- `SID`: storm ID
- `NAMELIST`: WRF namelist `namelist_analysis.input`
- `DOMAIN_HOME`: base path of the output directory.
  
  (for example, `/ptmp/$\{USER\}/HWRF/$\{SID\}/$\{YYYYMMDDHH\}`).

Then run `merge.ksh`. If the GSI analysis was run, this will update the HWRF initial conditions using the GSI analysis. If the GSI analysis was not run, the initial conditions from the vortex relocation stage 3 are updated.

**Overview of script `merge.ksh`:**

1. Check if the GSI analysis was run. If it was, copy the file `gsiprd/wrfinput_d01/wrf_inout` and rename it to `wrfinput_d01`; copy the file `gsiprd/wrfinput_d02/wrf_inout` and rename it to `wrfinput_d02`. If the GSI analysis was not run, copy the files `wrfinput_d01` and `wrfghost_d02` from directory `\$\{DOMAIN_HOME\}/relocateprd`.
2. Run `diffwrf_3dvar.exe` to convert the netCDF format `wrfinput_d01` and `wrfghost_d02` to unformatted data files `new_hdas_d01`, and `new_ght_d02`, respectively.
3. 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`).
4. Run `hwrft_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`).
5. Run `diffwrf_3dvar.exe` to convert the unformatted files `data_merge_d01` and `data_merge_d02` to netCDF format `wrfinput_d01` and `wrfinput_d02`.
6. Rename `wrfinput_d02` to `wrfanl_d02`.
7. `wrfinput_d01` and `wrfanl_d02` are ready to be used by `wrf.exe` to do the hurricane forecast.

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

- `wrfinput_d01`: initial condition for the outer domain containing the new vortex
- `wrfanl_d02_${YYYY-MM-DD}_00:00:00`: initial condition for the inner nest domain containing the new vortex. `${YYYY-MM-DD}` is the model run’s initial time.

---

**Figure 4.2. HWRF initialization procedures before vortex adjustments.**

- `met_nmm*`: meteorological input (parent)
- `wrfinput_d01`: initial condition for the outer domain
- `wrfbdy_d01`: boundary condition for the outer domain
- `wrfexe (1 min)`: executable file for 1-minute model run
- `ghost_d02`: ghost domain file
- `wrfanl_d02`: final condition for the inner nest domain

Color Coding for File Formats:
- **Ascii**
- **FORTRAN Binary**
- **GRIB**
- **BUFR/PrepBUFR**
- **NetCDF**

---

58
Figure 4.3. Diagram of HWRF vortex initialization Stage 1 procedures. The color coding is described in Figure 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 Figure 4.2.
Figure 4.6. Diagram of HWRF vortex initialization Stage 3 final procedures. The color coding is described in Figure 4.2.
Figure 4.7. Diagram of HWRF vortex initialization GSI procedures. The color coding is described in Figure 4.2.
Figure 4.8. Diagram of HWRF vortex initialization merge procedures. The color coding is described in Figure 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) fnm3(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 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) `mv -> 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 previous 6-h 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
hdas_atcfunix (fort.12) - Input track file from previous 6-h forecast
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.$cdate (fort.52) - diagnostics file (obs-previous 6-h forecast)
vital_syn.$cdate (fort.55) – information for generating bogus if storm not found in previous 6-h forecast

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

4.4.8 hwrf_anl_bogus_10m.exe
**FUNCTION:**
creates a bogus storm and adds it to the environmental flow.

**INPUT:**
$gesfrh(=6)$
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 boundary layer 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 $gesfrh | hwrf_anl_bogus_10m.exe`

### 4.4.9 hwrf_create_trak_guess.exe

**FUNCTION:**
guesses storm center from previous 6-h forecast position.

**INPUT:**
$storm\_id$ (storm ID)
$ih$ (model initial hour)
tcvitals.as (fort.11) – observed storm center
hdas_atcfunix (fort.12) – track file from previous cycle 6-h forecast

**OUTPUT:**
trak.fn1.all (fort.30) - storm center guess (at 0,3,6 9 h)

**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 boundary layer calculation
  (from hwrf_merge_nest_4x_step2.exe)
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.

**INPUT:**

$gesfhr(=6)$ $gesfhr$ last digit of the input/output file
$st_{int}$ (the 68-69 characters in the tcvital.as)
$ibgs(=1)$ 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.fn1.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 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)
$ibgs (=0)$ 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
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 (TCVitals characters 68-69)
$ibgs 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

4.4.16 hwrf_data_remv.exe
**FUNCTION:**
Utility program used to remove the observational data near storm center

**INPUT:**
fort.21 (prepbufr.ALL), the prepBUFR data before the observations are removed near the storm center
RLATC: storm center latitude
RLONC: storm center longitude
RRADC: radius within which data will be removed

**OUTPUT:**
fort.51: prepbufr, the prepBUFR data after the observations are removed near the storm center

**USAGE:**
./hwrf_data_remv.exe

### 4.4.17 ssrc.exe

**FUNCTION:**
Utility program to reverse the endianness when running GSI on little_endian machines (e.g. Linux) using big_endian prepBUFR data (e.g. those generated on IBM machines)

**INPUT:**
${PREPBUFR}$: the NCEP (big-endian) prepBUFR data

**OUTPUT:**
prepbufr: the little-endian prepBUFR data

**USAGE:**
./ssrc.exe

### 4.4.18 gsi.exe

**FUNCTION:**
GSI 3D-VAR data assimilation analysis main program

**INPUT:**
gsiparm.anl: gsi namelist, created by modifying
${HWRF_UTILITIES_ROOT}/parm/gsi_namelist.input
wrf_inout: background file, copied from ${BK_FILE}
prepbufr: conventional observation prepBUFR data, linked to ${PREPBUFR}
satellite radiance data, copied from \$\{OBS\_ROOT\}:

- amsuabufr
- amsubbufr
- hirs3bufr
- hirs4bufr
- mhsbufr
- airsbufr
- gsnd1bufr

fix files, from \$\{FIX\_ROOT\}, which is specified in run\_gsi.ksh

**OUTPUT:**

wrf\_inout: analysis results if GSI completes successfully. The format is same as the background file.

**USAGE:**

On IBM/AIX machines: `mpirun.lsf /gsi.exe < gsiparm.anl`

On Intel/Linux: `mpiexec –np 1 /gsi.exe < gsiparm.anl`
Chapter 5: Ocean Initialization of POM-TC

5.1 Introduction

This document explains how to run the initialization of the 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 six scripts, five of which are “kickit” scripts. The sixth script, “gfdl_pre_ocean_sortvit.sh,” is used by a “kickit” script and should not be changed.

The five “kickit” scripts are designed to be run in order from kickit00 through kickit04. If you are only running the default tutorial case, 2008082800, on Bluefire, then no changes should be needed to these scripts either, and you can easily run the entire ocean initialization as follows:

[prompt]$ ./kickit00_region.sh
[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 five “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
```
Note in addition to the scripts listed above, in the directory of

${HOME}/HWRF/src/pomtc/ocean_plot,

you can find two scripts, *pom-tc-united-grads.sh* and *pom-tc-eastatl-grads.sh*, that allow you to plot ocean output.

### 5.3 Procedures in the five “kickit” Scripts

#### 5.3.1 kickit00_region.sh

1. Set the storm ID.
2. Set the starting date for POM-TC as YYYYMMDDHH.
3. Define the directories.
4. Set up paths to certain shell commands.
5. Create the work directory if it does not already exist.
6. Slice up the starting date.
7. Use a piece of the storm ID to determine the ocean basin.
8. Continue with steps 9-15 below if region is Atlantic; otherwise, run uncoupled.
9. Run operational script *gfdl_pre_ocean_sortvit.sh* to extract track information for the specified storm from the yearly NHC hurricane message file (i.e. TCVitals). Then, append a line of zeros to the end of this complete track file.
10. Use an embedded Perl script to remove all cycles after the current one from the complete track file and save as a shortened track file.
11. Use the shortened track file for subsequent steps unless it is empty.
12. Extract various storm statistics from the track file and use those statistics to generate a 72-hour projected track that assumes storm direction and speed remain constant. Save this projected track in a “shortstats” file.
13. Run the find region code, which selects the ocean region based on the projected track points in the “shortstats” file.
15. End the region selection script by returning to the “ocean_script” directory.

#### 5.3.2 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. If ocean region is not “united,” skip steps 5-15 below.
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. If Loop Current files are missing, use climate LC penetration instead.
10. Use some pieces of the starting date to select the two climatology months.
11. Create parameter “input_sharp” depending on the chosen climatology.
12. Create symbolic links for the input files.
13. Run the sharpening code.
15. End the sharpening script by returning to the “ocean_script” directory.

5.3.3 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.

5.3.4 kickit03_phase3.sh

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

5.3.5 kickit04_phase4.sh

1. Set the storm ID.
2. Set the starting date for POM-TC as YYYYMMDDHH.
3. Define the directories.
4. Set the ocean region.
5. Create the work directory if it does not already exist.
6. Create the phase4 directory, overwriting old attempts.
7. If track file created in kickit00_region.sh is less than three lines, skip steps 8-14 below and use the phase 3 restart file to initialize the coupled HWRF run.
8. Slice up the starting date, using HH to define the model cycle.
9. Use the GNU date command 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 phase 4. The purpose of backing up 3 days is to end phase 4 at the coupled model starting date.
10. Copy track file to the phase4 directory, and modify the phase 4 parameter file by including the starting date, as well as the track and phase 3 restart files.
11. Create symbolic links for all input files.
12. Run the POM-TC code for phase 4.
13. Rename the phase 4 restart file in preparation for coupled HWRF run.
14. End the phase 4 script by returning to the “ocean_script” directory.

5.4 Executables

5.4.1 gfdl_find_region.exe

FUNCTION:
Determine which POM-TC region to use based on the current and projected storm track given in the “shortstats” file.

INPUT:
shortstats

OUTPUT:
fort.61 (ocean_region_info.txt)

USAGE:
${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_find_region.exe < shortstats

5.4.2 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.sfcasnl)
fort.11 (gfs.$\{start_date\}.t$\{cyc\}.z.sfcasnl)
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

5.4.3 **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.${mmm2}.ascii)
fort.24 (gfdl_ocean_readu.dat.${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 (mmdat.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

5.4.4 **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) in the United region.

**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.united.$\{mm\})
fort.66 (gfdl_ocean_topo_and_mask.united)
fort.14 (not used if phase 1; RST.phase3.united if phase 2)

OUTPUT:
(1) RST.phase3.united if phase 1; RST.final if phase 2
(2) phase3.out if phase 1; phase4.out if phase 2

USAGE:
Phase 1: ${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_ocean_united.exe > phase3.out
Phase 2: ${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_ocean_united.exe > phase4.out

5.4.5 gfdl_ocean_eastatl.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) in the East Atlantic region.

INPUT(S):
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.12 (gfdl_initdata.gdem.united.$\{mm\})
fort.13 (gfdl_initdata.eastatl.$\{mm\})
fort.66 (gfdl_ocean_topo_and_mask.eastatl)
fort.14 (not used if phase 1; RST.phase3.eastatl if phase 2)

OUTPUT(S):
(1) RST.phase3.eastatl if phase 1; RST.final if phase 2
(2) phase3.out if phase 1; phase4.out if phase 2

USAGE:
Phase 1: ${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_ocean_eastatl.exe > phase3.out
Phase 2: ${HOME}/HWRF/src/pomtc/ocean_exec/gfdl_ocean_eastatl.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 atmosphere 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 is responsible for understanding 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 Operational System and the batch system Load Sharing Facility (LSF), the script `wrf.ksh` should contain 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
```
#$ -pe wcomp 32  # parallel environment queue and number of processors
#$ -l h_rt=03:00:00 # Time limit
#$ -o output  # Output filename
#$ -e output  # Error filename

After setting up the batch system options, edit the script to set up the environment variables:

**WRF_ROOT:** path to the WRF code.
(for example, ${HOME}/HWRF/src/WRFV3)

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

**WRF_MODE:** Mode for the WRFV3 run. Must be one of: ‘analysis’, ‘ghost’, or ‘main’. For the coupled forecast, choose ‘main’

**NAMELIST:** Path to the WRFV3 namelist
(for example, ${HOME}/HWRF/src/hwrf-utilities/parm/namelist_main.input)

**DOMAIN_HOME:** Path to the top directory of the forecast output.
(for example, /ptmp/${USER}/HWRF/${SID}/${YYYY}${MM}${DD}${HH})

Also edit the script to specify the path to the file *gfdl_ocean_topo_and_mask.united* (for the Atlantic “united” domain) or *gfdl_ocean_topo_and_mask.eastatl* (for the east Atlantic domain) and link it to fort.66:

```
ln -s path/gfdl_ocean_topo_and_mask.united     fort.66 (for the United domain)
ln -s path/gfdl_ocean_topo_and_mask.eastatl    fort.66 (for the east Atlantic domain)
```

For the east Atlantic domain, edit the script to specify the path to file *gfdl_initdata.eastatl.${MM}* and link it to fort.13, where ${MM} is the month for the forecast storm:

```
ln -s path/gfdl_initdata.eastatl.${MM}         fort.13
```

also, for the east Atlantic domain, edit the script to specify the path to file *gfdl_initdata.gdem.united.${MM}* and link it to fort.12, where ${MM} is the month for the forecast storm:

```
ln -s path/gfdl_initdata.gdem.united.${MM}    fort.12
```

Note for the United domain, fort.13 is generated by the ocean initialization code in the directory of *oceanprd* (see Chapter 5), so the users do not need to manually specify its path.
Note that `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 the actual coupled forecast. The `WRF_MODE` used for these three `wrf.ksh` runs are “analysis”, “ghost” and “main”, respectively. The three `wrf.ksh` runs use different namelists too, which are specified by `namelist_${WRF_MODE}.input`

After the batch system options and environment variables are defined, run the script `wrf.ksh` using the command:

- **On IBM with LSF:**
  
  ```bash
  bsub < wrf.ksh
  ```
- **On Linux with SGE:**
  
  ```bash
  qsub wrf.ksh
  ```

### 6.3 Overview of the Script

1. Extract the initialization time components from `$NAMELIST`
2. Compute the forecast length that will be used in the coupler namelist.
3. Define the working directory as `${DOMAIN_HOME}/wrfprd` and enter the working directory.
4. Link the required input files:
   - 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 ocean model namelist `PARAMETERS.inp`.
7. Link the input files for the ocean model:
   - ocean topography and mask files:
     ```
     gfdl_ocean_topo_and_mask.united (for the united domain)
     gfdl_ocean_topo_and_mask.eastatl (for the east Atlantic domain)
     ```
   - ocean initial data:
     ```
     gfdl_initdata.united.${MM} (for the west Atlantic domain)
     gfdl_initdata.eastatl.${MM} (for the east Atlantic domain)
     ```
   - `RST.final`
   - `sst.gfs.dat`
8. Construct the coupler namelist `cpl.nml`.
9. Submit the coupled run.

- **On IBM with LSF:**
  Use the command `mpirun.lsf`  
  ```bash
  mpirun.lsf -cmdfile cmdfile
  ```
  where `cmdfile` is a file containing the list of executables. For example, the `cmdfile` file below indicates that the coupled run will be submitted to 90 processors, one for the coupler (`hwrf_wm3c.exe`), one for the United domain ocean model and 88 for `wrf.exe`:

  ```
  hwrf_wm3c.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`
  For example, the following command will run the coupled model using 90 processors, one for the coupler (`hwrf_wm3c.exe`), one for the United domain ocean model and 88 for `wrf.exe`:

  ```
  /usr/local/esrl/bin/mpirun -np 1 ./hwrf_wm3c.exe : -np 1 ./hwrf_ocean_united.exe : -np 88 ./wrf.exe
  ```

Note that in the examples listed above, for the east Atlantic domain, the ocean model `hwrf_ocean_eastatl.exe` is used.

10. The script `wrf.ksh` has the capability of uncoupled HWRF (atmosphere standalone) runs.
- **On IBM with LSF:**
  Use the command `mpirun.lsf`
  ```bash
  mpirun.lsf ${WRF_ROOT}/main/wrf.exe
  ```
  `wrf.exe` will be submitted using the number of processors specified by the LSF options

- **On Linux with SGE:**
  Use the command `mpirun`
For example, the following command will run the uncoupled model using 88 processors for `wrf.exe`

```
/usr/local/esrl/bin/mpirun -np 88 ./wrf.exe
```

### 6.4 Output Files in the Directory

Output files in directory `${DOMAIN_HOME}/wrfprd`

A successful run of `wrf.ksh` will produce output files with the following naming convention: `wrfout_d01_yyyy-mm-dd_hh:mm:ss` and `wrfout_d02_yyyy-mm-dd_hh:mm:ss`

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

`wrfout_d01_2008-08-28_00:00:00` and `wrfout_d02_2008-08-28_00:00:00`

The ocean model will produce diagnostic output files with the following naming convention: `GRADS.yymmddhh`

For example, the first POM-TC output file for a run started at 0000 UTC, 28th August 2008 would be: `GRADS.08082800`

### 6.5 Status Check

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

### 6.6 Executables

#### 6.6.1 `wrf.exe`

**FUNCTION:**

atmosphere component of HWRF

**INPUT:**

- 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 and gravity wave drag file gwd_surface
WRF static files
namelist_main.input

OUTPUT:
A successful run of wrf.exe will produce output files with the following naming convention:
wrfout_d01_yyyy-mm-dd_hh:mm:ss
For example, the first WRF output files for a run started at 0000 UTC, 28th August 2008 would be:
wrfout_d01_2008-08-28_00:00:00
and
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).

6.6.2 hwrf_wm3c.exe

FUNCTION:
coupler that links the atmospheric component wrf.exe and oceanic component hwrf_ocean_united.exe or hwrf_ocean_eastatl.exe

INPUT:
coupler namelist: cpl.nml

OUTPUT:
None

USAGE:
For a coupled HWRF forecast, the coupler hwrf_wm3c.exe must be submitted to the computers with the atmosphere model wrf.exe and the ocean model hwrf_ocean_united.exe or hwrf_ocean_eastatl.exe (see Section 6.2).

6.6.3 hwrf_ocean_united.exe

FUNCTION:
oceanic model for HWRF, for the United domain
INPUT:

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

Note the ocean’s initial state of temperature and salinity for the United domain (gfdl_initdata.united.${MM}) comes from the ocean initialization with a sharpening process.

OUTPUT:

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, 28th August 2008 would be: GRADS.08082800

USAGE:

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_wm3c.exe (see Section 6.2).

6.6.4 hwrf_ocean_eastatl.exe

FUNCTION:

Oceanic model for HWRF, for the east Atlantic domain

INPUT:


gfdl_ocean_topo_and_mask.eastatl

gfdl_initdata.eastatl.${MM}, ${MM} is the month for the forecast storm

gfdl_initdata.gdem.united.${MM}, ${MM} is the month for the forecast storm

RST.final

sst.gfs.dat

mask.gfs.dat

lonlat.gfs

track
Note the ocean’s initial state of temperature and salinity for East Atlantic basin (gfdl_initdata.eastatl.${MM}) comes from fixed data based on climatology.

**OUTPUT:**
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, 28 August 2008 would be: 
GRADS.08082800 

**USAGE:**
For a coupled HWRF forecast, the ocean model hwrf_ocean_eastatl.exe must be submitted to the computers with the atmosphere model wrf.exe and the coupler hwrf_wm3c.exe (see Section 6.2).

### 6.6.5 hwrf_swcorner_dynamic.exe

**FUNCTION:**
Calculates the lower-left corner of the nest as (i_parent_start, j_parent_start).

**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:

```plaintext
i_start=00098
j_start=00208
```

**USAGE:**

```
${HOME}/HWRF/src/hwrf-utilities/exec/hwrf_swcorner_dynamic.exe
```
Chapter 7: HWRF Post Processor

7.1 Introduction

The NCEP UPP was designed to de-stagger HWRF parent and nest domain output, compute diagnostic variables and interpolate from their native grids to NWS standard levels (pressure, height, etc.) and standard output grids (latitude/longitude, AWIPS, Lambert Conformal, polar- stereographic, etc.), in NWS and 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 UPP code is available in Chapter 2.

7.2 Script

The UPP script used by HWRF, run_unipost, is distributed in the tar file HWRF-UTILITIES.tar.gz and, following the procedure outlined in Chapter 2, will be expanded in the directory of ${HOME}/HWRF/src/hwrf-utilities/scripts

Edit the script run_unipost to set the following environment variables:

- **UPP_ROOT**: Path to the UPPV1 code
  (for example, ${HOME}/HWRF/src/UPPV1)
- **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_unipost

Note that other UPP scripts are distributed in the UPP release tar file UPPV1.tar.gz but they do not perform all the processes required for HWRF.

A script named run_grads is provided that allows the users to run GrADS to plot UPP outputs. The users can find the script run_grads in the directory of

${HOME}/HWRF/src/hwrf-utilities/scripts.
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 ${HOME}/HWRF/src/hwrf-utilities/parm/hwrf_eta_micro_lookup.dat.
   • Link the control file ${HOME}/HWRF/src/hwrf-utilities/parm/hwrf_cntrl.hurcn to wrf_control.parm in the working directory.

File ${HOME}/HWRF/src/hwrf-utilities/parm/hwrf_cntrl.hurcn is used to 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 GFDL Vortex Tracker (see Chapter 8), 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:
   o Create namelist itag that will be read in by unipost.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 or binary; choose netCDF for HWRF).
     ▪ Forecast valid time (not model start time) in WRF format.
     ▪ Model name (NMM or NCAR; choose NMM for HWRF).
   o Run unipost and check for errors. The execution command in the distributed scripts is for a single processor:
     unipost.exe < itag > outpost
   o Set up a copygb grid, ${hr_grid}, to interpolate the native unipost.exe output GRIB file to (see full description under Run copygb below). For both the parent and nest domains, the grid spacing in {hr_grid} is set to 0.1 degree.
   o Copygb can be used to horizontally interpolate a GRIB file to a user-specified grid. Run copygb to interpolate the unipost.exe output GRIB file
to the copygb grid \$\{hr\_grid\} and check for errors.

\[\text{copygb.exe -xg} \$\{hr\_grid\} \"\text{WRFPRS}_\$\{domain\}_\$\{fhr\}\ \\
\text{wrfprs}_\$\{domain\}_\$\{fhr\}\]

where \$\{hr\_grid\} refers to the output grid to which the native forecast is being interpolated, \text{WRFPRS}_\$\{domain\}_\$\{fhr\} is the native unipost.exe output GRIB file and \text{wrfprs}_\$\{domain\}_\$\{fhr\} is the copygb output GRIB file in which the forecast output is interpolated to the grid \$\{hr\_grid\}.

- Run copygb again to get the combined domain output. The combined domain output has a grid spacing of 0.1 degree.

\textit{Output files} in the working directory \$\{DOMAIN\_HOME\}/postprd:

- \textit{wrfprs\_d01}.$\{fhr}$: HWRF parent domain in GRIB format
- \textit{wrfprs\_d02}.$\{fhr}$: HWRF nest domain, interpolated onto the parent domain, in GRIB format
- \textit{wrfprs}.$\{fhr}$: HWRF combined domain in GRIB format output

\textit{Status check:}

If “End of Output Job” is found in the standard output (stdout), the HWRF UPP script has finished successfully.

### 7.4 Executables

#### 7.4.1 \texttt{unipost.exe}

**FUNCTION:**
de-staggers the HWRF native output (wrfout\_d01 or wrfout\_d02), interpolates it vertically to pressure levels, computes derived variables, and outputs in GRIB format.

**INPUT:**
Table \$\{HOME\}/HWRF/src/hwrf-utilities/parm/hwrf_eta_micro_lookup.dat
unipost control file \texttt{wrf\_cntrl.parm}
HWRF native output (wrfout\_d01 or wrfout\_d02)
namelist itag

**OUTPUT:**
HWRF output in GRIB format \texttt{WRFPRS}_\$\{domain\}_\$\{fhr\}
**USAGE:**

```
$HOME/HWRF/src/UPPV1/bin/unipost.exe < itag
```

### 7.4.2 `copygb.exe`

**FUNCTIONS:**

- a) interpolates a GRIB file to a user-specified grid
- b) combines two GRIB files

**INPUT:**

- user-specified grid (`$hr_grid`)
- unipost.exe output (`WRFPRS_${domain}.${fhr}`)

**OUTPUT:**

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

**USAGE:**

- a) `$HOME/HWRF/src/UPPV1/bin/copygb.exe -xg"${hr_grid}" WRFPRS_${domain}.${fhr} wrfp${domain}.${fhr}`

- b) When a “-M” option is used, and the argument following it is a GRIB file, the GRIB file will be interpreted as a merge file. This option can be used to combine two GRIB files.

For example, the following command will combine `wrfprs_d01.${fhr}` and `wrfprs_d02.${fhr}` to `wrfprs.${fhr}`, whose grid is specified by `$hr_grid`.

```
$HOME/HWRF/src/UPPV1/bin/copygb.exe -g"${hr_grid}" -xM wrfp${domain}.${fhr} wrfp${domain}.${fhr}
```
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, UPP 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 grbindx. The GRIB utility wgrib is also used for preparing data for the tracker. Both grbindx and wgrib were developed by NCEP; grbindx is distributed by the DTC as part of the hwrf-utilities and wgrib can be obtained from http://www.epc.noaa.gov/products/wesley/wgrib.html.

This version of the tracker contains a preliminary capability of tracking cyclogenesis and identifying cyclone thermodynamic phases. These features are untested and not supported by the DTC yet.

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-
To run the vortex tracker,

first edit the script to set up the following environmental variables:

**UPP_ROOT**
- directory of the Unified post-processing package
  - (for example, `${HOME}/HWRF/src/UPPV1`)

**DOMAIN_HOME:**
- top level directory of HWRF output
  - (for example, `/ptmp/user/HWRF/09L/2008090900`)

**NAMELIST**
- path to the WRFV3 namlist
  - (for example, `${HOME}/HWRF/src/hwrf.utilities/parm/namelist_main.input`)

**SID:**
- storm ID (for example, 09L)

**ATCFNAME**
- character model ID that will appear in the ATCF output
  - (for example, HTUT)

**WGRIB**
- path to the utility wgrib
  - (for example, `/usr/local/bin/wgrib`)

**GNUDATE**
- path to the GNU version of date command
  - (for example, `/bin/date`)

After the environmental variables are properly defined, run the tracker script with the command: *tracker.ksh*

### 8.3 Overview of the Script

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

1. Obtain the model initialization date from the namelist
   *hwrf-utilities/parm/namelist_main.input*
2. Extract, from the UPP output in ${DOMAIN_HOME}/postprd, the fields needed by tracker for each output time and concatenate them together to produce one GRIB file. The extracted fields include:
   - winds at 10 m, 500 mb, 700 mb, and 850 mb
   - absolute vorticity at 10 m, 700 mb and 850 mb
   - geopotential height at 700 and 850 mb
   - MSLP
3. Run *grbindx* to get a GRIB index file for the GRIB file generated in 2
4. Get the output times from the GRIB file generated in 2
5. Create a file, *fcst_minutes*, which contains the forecast output times the tracker will process
6. Generate the namelist for the vortex tracker
7. Link the input files (see GFDL vortex tracker software input description)
8. Run the tracker executable *hwrf_gettrak.exe*
9. Output will be generated in `${DOMAIN_HOME}/gvtparams`
8.4 Executables

8.4.1 *hwrf-gettrak.exe*

**INPUT:**

*fort.11*: GRIB file containing the postprocessed HWRF forecast

*fort.12*: TCVitals file containing the first guess location of the forecast vortex

For example, the following TCVitals 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*: TCVitals file used for tropical cyclonegenesis 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.

```
1 0
2 180
3 360
4 540
```

Note the format of the records in this file is a 4-digit integer showing the number of the forecast lead time, followed by 1 blank space, followed by a 5-digit integer showing the forecast lead time in minutes.

*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>------------</td>
<td>--------------------------------------------------</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. The Model ID numbers for other models are listed below: (1) GFS, (2) MRF, (3) UKMET, (4) ECMWF, (5) NAM, (6) UKMET, (7) NCEP, (8) GDAS, (9) NCEP Ensemble, (10) ECMWF Ensemble, (11) ECMWF Ensemble, (12) SREF Ensemble, (13) NCEP Ensemble, (14) NCEP Ensemble, (15) CMC, (16) CMC Ensemble, (17) HWRF Ensemble, (18) HWRF-DAS (HDAS), (19) HWRF-DAS (HDAS), (20) Ensemble RELOCATION (21) UKMET hi-res (NHC)</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>inp%file_seq</td>
<td>'onebig' or 'multi', this specifies if the tracker will process one big input file or multiple files for each individual lead times</td>
</tr>
<tr>
<td>fnameinfo%gmod name</td>
<td>Defines the model name in the input files, e.g., 'hwrf'. Only when inp%file_seq='multi'</td>
</tr>
<tr>
<td>fnameinfo%runde scr</td>
<td>Describe the model runs in the input files, e.g., 'combined'. Only when inp%file_seq='multi'</td>
</tr>
<tr>
<td>fnameinfo%atcfde scr</td>
<td>Describe the storm information in the input files, e.g., 'gustav07l'. Only when inp%file_seq='multi'</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 yyyyymmddhh date that will be used in output text track files</td>
</tr>
<tr>
<td>trkrinfo%westbd</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%eastbd</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>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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 TCVitals. 'tegen' and 'midlat' run in genesis mode and will look for new storms in addition to tracking from TCVitals. 'tegen' will look for all parameters at the various vertical levels, while 'midlat' will only look for mslp. Presently the only support option is 'tracker'. For HWRF, choose 'tracker'.</td>
</tr>
<tr>
<td>trkrinfo%mslpthresh</td>
<td>Threshold for the minimum MSLP gradient (units mb/km) that must be met in order to continue tracking.</td>
</tr>
<tr>
<td>trkrinfo%v850thresh</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>trkrinfo%gridtype</td>
<td>'global' or 'regional', this defines the type of domain grid. For HWRF or other limited area models, choose 'regional'.</td>
</tr>
<tr>
<td>trkrinfo%contint</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_vit</td>
<td>This is only set to 'y' if the tracker is running in genesis mode, and it tells the tracker to write out a &quot;TCVitals&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. Choose 'n' for HWRF</td>
</tr>
</tbody>
</table>
phasescheme | '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'

structflag | 'y' or 'n', tells the program whether or not to determine the cyclone thermodynamic structure. Choose 'n' for HWRF.

**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 in 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, 0025, 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, 0030, 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 and “EP” Northeast Pacific
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 hours).
Column 7-8: vortex center position (latitude and longitude multiplied by 10).
Column 9: vortex maximum 10-m wind (in kt).
Column 10: vortex minimum MSLP (in hpa).
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 quadrant.
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`.
- Run `gribmap` on the GrADS ctl file `plottrak.ctl`. `gribmap` is a GrADS utility that maps what is in the ctl file with the binary data that it finds inside the 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 `plottrak.ctl` and the newly created `plottrak.ix` map file need to be in the directory where the script below is run.

99
Edit the atcfplot.sh to set the following paths:
1. gradsv2: path to the GrADS executable (for example, /contrib/grads/bin/gradsc).
2. GADDIR: path to the directory containing the supplemental font and map files for GrADS (for example, /contrib/grads/lib).
3. scrdir: path to the working directory (for example, /glade/home/${USER}/HWRF/src/gfdl-vortextracker/trk_plot/plottrak).
4. plotdir: path to the directory where the plot files will be created (for example, /glade/home/${USER}/HWRF/src/gfdl-vortextracker/trk_plot/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 script ndate.ksh
4. nhour: path to the script nhour.ksh

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 script ndate.ksh
4. nhour: path to the script nhour.ksh

The users need to insert or append their vortex tracker output, fort.64, into the file a${Basin}${SID}${YYYY}.dat.

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 $rundir (${SID} is the storm ID) for storms in year ${YYYY} in the ${Basin} basin.

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.

The default tracker namelist is set to use the ATCF model name “HTUT”. If the user changes this name in the tracker namelist, the ATCF_PLOT GUI will not recognize the new name. In this case, the user needs to replace an unused atcfname with the new atcfname. The atcfnames in the GUI can be found by searching in function “modnames” in file atcfplot.gs. Note all three instances of the unused atcfname need
to be replaced in *atcfplot.gs*.

For example, if “USER” was employed as the atcfname in the users’ GFDL Vortex Tracker output *fort.64*, file *atcfplot.gs* needs to be modified to have the ATCF_PLOT program GUI interface show a button for the atcfname “USER”. To do that, open file *atcfplot.gs*, go to function “modnames”, find an atcfname that will not be used, for example “HTUT”, and manually replace the string “HTUT” with “USER”. Also make the