NOAA Technical Memorandum OAR GSD-42

COMMUNITY HWRF USERS GUIDE V3.4A

AUGUST 2012

THE DEVELOPMENTAL TESTBED CENTER

S. Bao
R. Yablonsky
D. Stark
L. Bernardet

Earth System Research Laboratory
Global System Division
Boulder, Colorado
September 2013

noaa NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION / Office of Oceanic and Atmospheric Research

NOAA Technical Memorandum OAR GSD-42

COMMUNITY HWRF USERS GUIDE V3.4A

Shaowu Bao[1]
Richard Yablonsky[2]
Don Stark[3]
Ligia Bernardet[1]

[1] Cooperative Institute for Research in Environmental Sciences (CIRES) and NOAA/ESRL/GSD

[2] University of Rhode Island

[3] National Center for Atmospheric Research

UNITED STATES DEPARTMENT OF COMMERCE

Penny Pritzker
Secretary

NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION

Dr. Kathryn Sullivan
Acting Under Secretary for Oceans
And Atmosphere/acting Administrator

Office of Oceanic and Atmospheric Research

Dr. Robert Detrick
Assistant Administrator

Community HWRF Users Guide V3.4a

August 2012

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 and John Osborn of NOAA/ESRL/GSD for providing edit support for this document and addressing a number of formatting issues.

Table of Contents

Chapter 1: HWRF System Introduction

1.1 HWRF System Overview

The Weather Research and Forecast (WRF) system for hurricane prediction (HWRF) is an operational model implemented at the National Weather Service (NWS)/ National Centers for Environmental Prediction (NCEP) to provide numerical guidance to the National Hurricane Center for the forecasting of tropical cyclones' track, intensity, and structure. HWRF v3.4a and this Users Guide match the operational 2012 implementation of HWRF.

The HWRF model is a primitive equation non-hydrostatic coupled atmosphere-ocean model with the atmospheric component formulated with 42 levels in the vertical. The model uses the Non-hydrostatic Mesoscale Model (NMM) dynamic core of WRF (WRF-NMM) with a parent and two nest domains coded in the WRF framework. The grid projection of the model is rotated latitude-longitude with E-staggering. The parent domain covers roughly 80^o x 80^o on a rotated latitude/longitude E-staggered grid. 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 middle nest domain of about 11° x 10° and the inner nest domain of about 6.0° x 5.5° move 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 middle nest spacing is 0.06° (about 9 km) and the inner nest spacing is 0.02° (about 3 km). The time steps for HWRF are 45, 15, and 5 s, respectively, for the parent, middle nest, and inner nest domains. It is also possible to configure HWRF to use only two domains, a parent grid and a single moving nest with spacing of 27- and 9-km, respectively. Information about this configuration, which was used in operational implementations until 2011, will be provided upon sending a request to wrfhelp@ucar.edu.

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 explicit moist physics. 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 includes 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 removing the GFS vortex and inserting 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 2^{nd} order Smagorinsky-type following (Janjic 1990).

The Community HWRF model can only be used for the two basins for which the national hurricane center is responsible: north Atlantic and northeast Pacific. In both basins, the atmospheric model is coupled with the Princeton Ocean Model (POM) for Tropical Cyclones (POM-TC). The POM was developed at Princeton University. At the University of Rhode Island (URI), the POM was coupled to the GFDL and HWRF models. In the eastern north Pacific, a one-dimensional (column) configuration of the POM-TC is employed, while in the Atlantic basin, POM-TC is run in three dimensions. In both basins the horizontal grid spacing is 1/6° (approximately 18 km). In the Atlantic, the POM-TC is configured with 23 vertical levels, while 16 levels are used in the eastern north Pacific.

The POM-TC is initialized by a diagnostic and prognostic spin up of the ocean circulations using climatological ocean data. For storms located in the western part of the Atlantic basin, the initial conditions are enhanced with real-time sea surface temperature, sea surface height data, and the assimilation of oceanic "features". During the ocean spin up, realistic representations of the structure and positions of the Loop Current, Gulf Stream, and warm- and cold-core eddies are incorporated using a features-based data assimilation technique developed at URI.

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 support to the community by the Developmental Testbed Center (DTC) includes the following three main modules.

- **HWRF atmospheric components**
    - WRF-NMM V3.4a (which has tropical physics schemes and a vortex-following moving nest)
    - WRF Preprocessing System (WPS)
    - Vortex initialization
    - Gridpoint Statistical Interpolation (GSI)

- Post-processing
- 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's Environmental Modeling Center (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.

1.2 HWRF Development and Support

The general WRF code repository is used for the development and support of the HWRF system. The atmospheric model used in 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, documentations and tutorials can be found at *http://www.dtcenter.org/HurrWRF/users* and in the following chapters of this Users Guide. Direct all questions to wrfhelp@ucar.edu.

1.3 HWRF Source Code Directory Structure

The HWRF system source code has the following eight components.

- WRF Atmospheric Model
- WPS
- Unified Post Processor (UPP)
- GSI

- HWRF Utilities
- 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).

- *hwrfv3.4a_utilities.tar.gz*
- *hwrfv3.4a_pomtc.tar.gz*
- *hwrfv3.4a_gfdl_vortextracker.tar.gz*
- *hwrfv3.4a_ncep-coupler.tar.gz*
- *hwrfv3.4a_wrf.tar.gz*
- *hwrfv3.4a_wps.tar.gz*
- *hwrfv3.4a_upp.tar.gz*
- *hwrfv3.4a_gsi.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
- *WPSV3* –WRF Pre-Processor
- *UPP* –Unified Post-Processor
- *GSI* – 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 *${SCRATCH}/HWRF*.

The directory trees for these eight components are listed as follows.

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

| |____	*arch/*	(compile options)
| |____	*clean*	script to clean created files and executables
| |____	*compile*	script to compile the HWRF-Utilities code
| |____	*configure*	script to create the configure.hwrf file for compile
| |____	*exec/*	(executables)
| |____	*libs/*	(libraries including blas, sp, sfcio, bacio, w3 and bufr)
| |____	*makefile*	top level makefile
| |____	*parm/*	(various namelists including those for WPS, WRF, GSI, and UPP;

| | all WRF lookup tables from run subdirectory, some of which are modified for HWRF)
| |____*scripts/* (scripts used to run HWRF system)
| | |____*funcs (shell functions used by the scripts)*
| |____*tools/* (source code for tools to run HWRF system)
| | *Makefile* makefile for tools code
| | |____*grbindex*
| | |____*hwrf_data_remv*
| | |____*hwrf_wrfout_newtime*
| | |____*wgrib*
| |____*vortex_init_2d/* (source code for the 2-domain, non-operational, configuration of HWRF)
| | *Makefile* makefile for vortex_init_2d code
| | |____*hwrf_anl_bogus*
| | |____*hwrf_anl_cs*
| | |____*hwrf_anl_step2*
| | |____*hwrf_create_nest*
| | |____*hwrf_create_trak_fnl*
| | |____*hwrf_create_trak_guess*
| | |____*hwrf_diffwrf_3dvar*
| | |____*hwrf_guess*
| | |____*hwrf_pert_ct*
| | |____*hwrf_set_ijstart*
| | |____*hwrf_split*
| | |____*interpolate*
| | |____*merge_nest*
| |____*vortex_init/* (source code for the 3-domain (operational) configuration of HWRF
| | *Makefile* makefile for vortex_init code
| | |____*hwrf_anl_bogus*
| | |____*hwrf_anl_cs*
| | |____*hwrf_anl_step2*
| | |____*hwrf_create_nest*
| | |____*hwrf_create_trak_fnl*
| | |____*hwrf_create_trak_guess*
| | |____*hwrf_diffwrf_3dvar*
| | |____*hwrf_guess*
| | |____*hwrf_pert_ct*
| | |____*hwrf_set_ijstart*
| | |____*hwrf_split*
| | |____*interpolate*
| | |____*merge_nest*
| |____*wrapper_scripts/* (top-level wrapper scripts to run HWRF system)

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)
	Makefile makefile for the ocean initialization code
	|____*eastatl*
	|____*ext_eastat*
	|____*eastpac*
	|____*united*
	|____*gfdl_find_region*
	|____*getsst*
	|____*sharp_mcs_rf_l2m_rmy5*
	|____*date2day*
	|____*day2date*
|____*ocean_main/*	(source code for the ocean forecast model)
	Makefile makefile for the ocean model code
	|____*ocean_united*
	|____*ocean_eastatl*
	|____*ocean_eastatl_ext*
	|____*ocean_eastpacl*
|____*ocean_parm/*	(namelists for ocean model)
|____*ocean_plot/*	(sample GrADS scripts used to plot ocean output)

3. ncep-coupler (NCEP Coupler)

|____*arch/*	(compile options)
|____*clean*	script to clean created files and executables
|____*compile*	script to compile the coupler code
|____*configure*	script to create the configure.cpl file for compile
|____*cpl_exec/*	(coupler executables)
|____*hwrf_wm3c/*	(source code for an updated 3-way version of the coupler)
|____*makefile*	top level makefile

4. gfdl-vortextracker (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)

| |____ *Makefile* *makefile used to compile WRFV3*
| |____*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- Adanced Research WRF (ARW) dynamic modules, not used in HWRF)
| |____*dyn_exp/* ('toy' dynamic core, not used by HWRF)
| |____*dyn_nmm/* (WRF-NMM dynamic modules, used by HWRF)
| |____*external/* (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 *http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/V3/users_guide_nmm_chap1-7.pdf*

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)
| |____*link_grib.csh* script used by ungrib to link input GRIB files, not used by HWRF
| |____*metgrid/* (source code for metgrid.exe)
| |____*ungrib/* (source code for ungrib.exe)
| |____*test_suite/* (WPS test cases)
| |____*util/* (utility programs for WPSV3)

7. UPP (Unified Post-Processor)

| | |____*arch/* (compile options)
| | |____*bin/* (executables)
| | |____*clean* script to clean created files and executables
| | |____*compile* script to compile the UPP code
| | |____*configure* script to create the configure.upp file for compile
| | |____*include/* (include files)
| | |____*lib/* (libraries)
| | |____*makefile* makefile used to build UPP code
| | |____*parm/* (parameter files, which controls how the UPP is performed, not used by HWRF)
| | |____*scripts/* (sample scripts running UPP, not used by HWRF)
| | |____*src/* (UPP source codes)

8. GSI (Gridpoint Statistical Interpolation)

| | |____*arch/* (compile options)
| | |____*clean* script to clean created files and executables
| | |____*compile* script to compile the GSI code
| | |____*configure* script to create the configure.gsi file for compile
| | |____*include/* (include files)
| | |____*lib/* (libraries)
| | |____*makefile* makefile used to build GSI code
| | |____*run/* (executables)
| | |____*scripts/* (sample scripts running GSI, not used by HWRF)
| | |____*src/* (GSI source codes)
| | |____*util/* (GSI utilities, not used by HWRF)

1.4 Input Data Directory Structure

Users will need the datasets below as input to HWRF components. Test 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. ***abdecks*** **(a-deck and b-deck files)**
3. ***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.20110823* and

hwrf_gfdl_loop_current_wc_ring_rmy5.dat.20110823 contain the loop current and warm-core ring information for August 23 2011.

4. *GFS* (input data from GFS)

4.1 *gridded* (GFS gridded data, for WRF initialization)

GFS_HWRF34a_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.2011082312.pgrbf024* is a GFS 24 hour forecast whose initial time is August 23 12Z 2011.

4.2 *spectral* (GFS spectral data, for ocean initialization)

GFS_HWRF34a_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.2011082312.t12z.sanl* and *gfs.2011082312.t12z.sfcanl* are initial-time GFS spectral data in a GFS forecast whose initial time is August 23 12Z 2011.

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

GFS_HWRF34a_obs has subdirectories ${*YYYYMMDDHH}*, which in turn, contain the following data that will be used in GSI:
gfs.${YYYYMMDDHH}.prepbufr
where *${YYYYMMDDHH}* is the date for the observations.

5. *fix*

5.1 *crtm* (for UPP)

5.2 *gsi* (for GSI)

5.3 *pomtc* (for ocean initialization)

gfdl_ocean_topo_and_mask.eastatl
gfdl_ocean_topo_and_mask.eastatl_extn
gfdl_Hdeepgsu.eastatl
gfdl_gdem.[00-13].ascii
gfdl_initdata.eastatl.[01-12]
gfdl_initdata.gdem.united.[01-12]
gfdl.initdata.united.[01-12]
gfdl_ocean_readu.dat.[01-12]
gfdl_ocean_spinup_gdem3.dat.[01-12]
gfdl_ocean_spinup_gspath.[01-12]

gfdl_ocean_spinup.BAYuf
gfdl_ocean_spinup.FSgsuf
gfdl_ocean_spinup.SGYREuf
gfdl_ocean_topo_and_mask.united
gfdl_pctwat
gfdl_raw_temp_salin.eastpac.[01-12]

1.5 Production Directory Structure

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: *${HWRF_OUTPUT_DIR}/SID/${yyyymmddhh}* (where SID is storm ID, e.g., 09L, and ${yyyymmddhh} is the forecast initial time).
The directory *${HWRF_OUTPUT_DIR}/SID/${yyyymmddhh}* 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. *wrfanalysisprd* (created by *wrf.ksh* run in "analysis" mode)
8. *trkanalysisprd* (created by *track_analysis.ksh*)
9. *relocateprd* (created by vortex initialization scripts)
10. *gsiprd* (created by GSI scripts, if GSI is run)
11. *mergeprd* (created by *merge.ksh*)
12. *oceanprd* (created by ocean initialization scripts)
    - *sharpn* (created by the sharpening program, present only for "united" ocean domain)
    - *getsst* (created by the procedure to extract the SST from GFS)
    - *phase3* (created by the 48-hr spin-up procedure to generate geostrophically-balanced currents)
    - *phase4* (created by the 72-hr spin-up procedure using the wind stress extracted from the NHC hurricane message file)
13. *wrfprd* (created by *wrf.ksh* in "main" mode)
14. *postprd* (created by *run_unipost*)
15. *gvtprd* (created by *tracker.ksh*)

1.6 Scripts for Running HWRF

It is recommended that HWRF v3.4a be run using the shell scripts provided with the HWRF v3.4a release. The scripts are located in two directories. In *hwrf-utilities/scripts*, the users can find low-level scripts that interact with the executables, input/output files, and work directories. These scripts require that several environment variables be set. To make running HWRF easier, a set of high-level wrapper scripts and a list of global variables are provided in the directory *hwrf-utilities/wrapper_scripts*. The users should edit the list of global variables to customize the options for running the HWRF system. The wrapper scripts will read the options specified in the list of global variables and drive the low-level scripts. Usually users will not need to modify the low-level and wrapper scripts, unless instructed otherwise in the following chapters of the Users Guide. The list of the environment variables defined in *hwrf-utilities/wrapper_scripts/global_ vars.ksh* can be found in Appendix.

Some of the executables are parallel code and can only run on the computation nodes. We recommend that users first connect to the computer's remote computation nodes. To do this on Linux machines that run the Oracle grid engine, such as Jet, users can use the *qrsh* command. For example, the command below requests a one-hour connection of 12 cores on the "hfip" nodes using the account "dtc-hurr".

qrsh –A dtc-hurr –pe hfip 12 –l h_rt=1:00:00

The user should seek assistance from the system administrator on how to connect to the computation nodes on the machine used to run HWRF.

After the interactive job starts and the user is connected to the computation nodes, the user should also setup the environment to run parallel jobs. On Linux machines that run the Oracle grid engine, this can be done by running the following commands.

For csh or tcsh:

```
set host=`uname -n`
echo $host
setenv JOB_ID `qstat | grep "$host " | awk '{print $1}'`
setenv MACHINE_FILE /tmp/${JOB_ID}*/machines
setenv NSLOTS `cat $MACHINE_FILE| wc -l`
```

For bash or ksh:

```
host=`uname -n`
echo $host
export JOB_ID=`qstat | grep "$host " | awk '{print $1}'`
fn=`ls /tmp/${JOB_ID}*/machines`
export MACHINE_FILE=$fn
export NSLOTS=`cat $MACHINE_FILE| wc -l`
```

After the environment is set up to run parallel jobs, the user can run the wrapper scripts by typing their names in the terminal.

Parallel code can also be submitted to the computation nodes using a batch system. For the IBM platform that uses the AIX Operational System and the batch system Load Sharing Facility (LSF), the wrapper script *hwrf_wrapper* should be edited to contain the LSF options listed below:

```
#BSUB -P 99999999                  # Project 99999999
#BSUB -a poe                       # select poe
#BSUB -n 202                       # 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 Oracle Grid Engine, previously known as Sun Grid Engine (SGE) batch system, the wrapper script *hwrf_wrapper* should be edited to contain the SGE options listed:

```
#$ -cwd -V                         #Job will execute from current directory and
                                   export variables
#$ -N HWRF                         # Job name
#$ -A 99999999                     # Project Account
#$ -pe hfip 202                    # parallel environment queue and number of
                                    processors
#$ -l h_rt=03:00:00                # Time limit
#$ -o output                       # Output filename
#$ -e output                       # Error filename
```

After the batch system options and environment variables are defined, run the wrapper scripts (for example *hwrf_wrapper*) using the command:

- On IBM with LSF:
  bsub < hwrf_wrapper
- On Linux with SGE:
  qsub hwrf_wrapper

The wrapper script *hwrf_wrapper* will be submitted to the computation nodes and, once it starts, will call the low-level script *wrf.ksh*.

Chapter 2: Software Installation

2.1 Introduction

The DTC community HWRF system, which is based on the NOAA operational WRF hurricane system (HWRF), consists of eight component models.

- 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. GSI is a 3D variational data assimilation code used for data assimilation, and the remaining four components are specific to the hurricane system itself, 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 consists of 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.

All of the HWRF components can be obtained through the HWRF website

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

by selecting the *Download* 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.

- *hwrfv3.4a_utilities.tar.gz*
- *hwrfv3.4a_pomtc.tar.gz*
- *hwrfv3.4a_gfdl-vortextracker.tar.gz*
- *hwrfv3.4a_ncep-coupler.tar.gz*
- *hwrfv3.4a_wrf.tar.gz*
- *hwrfv3.4a_wps.tar.gz*
- *hwrfv3.4a_upp.tar.gz*
- *hwrfv3.4a_gsi.tar.gz*

After downloading each of the component codes, the user should check the links to *known issues* and *bug fixes* 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 are reasonably flexible and with minimal effort can support almost any layout. For simplicity, it is assumed that the HWRF system will be set up in a single flat directory structure. Because of the storage requirements necessary for the complete HWRF system setup, it typically will need to be located on a computer's "scratch" or "temporary" storage space. Therefore before unpacking the tar files you have just downloaded, create a single working directory in that workspace. Then move the tar files into it, and unpack them there.

You may use the UNIX commands:

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

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.

```
tar –xvf  hwrfv3.4a_utilities.tar
tar –xvf  hwrfv3.4a_pomtc.tar
tar –xvf  hwrfv3.4a_gfdl-vortextracker.tar
tar –xvf  hwrfv3.4a_ncep-coupler.tar
```

tar –xvf hwrfv3.4a_wrf.tar
tar –xvf hwrfv3.4a_wps.tar
tar –xvf hwrfv3.4a_upp.tar
tar –xvf hwrfv3.4a_gsi.tar

Once unpacked, there should be the eight source directories.

- *WRFV3* –Weather Research and Forecasting model
- *WPSV3* –WRF Pre-processor
- *UPP* –Unified Post-Processor
- *GSI* – 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

A ninth directory for the output can also be created here as well.

mkdir results

The user should make sure the output directory created here is consistent with the environment variable HWRF_OUTPUT_DIR defined in *hwrf-utilities/wrapper_scripts/global_vars.ksh*.

We will next discuss the system requirement to build the HWRF system.

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/OpenMP for the distributed memory parallelism, and the I/O relies on the netCDF I/O libraries. Beyond the 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 listed below.

- FORTRAN 90+ compiler
- C compiler
- MPI v1.2+
- Perl
- netCDF V3.6+
- 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 with
  - PGI (pgf90+pgcc)
  - Intel (ifort+icc)

HWRF has only been tested on the IBM AIX v5, Linux PGI (v11) and Linux Intel (v11). Unforeseen build issues may occur when using older compiler versions. As always, the best results come from using the most recent version of things.

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/.

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.

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 are listed below.

- 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 the 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 vender-supplied versions of BLAS are typically incomplete, and the local version supplements the vender version. Typically this is for

any system other than 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 in any order. 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 either of these components. The remaining component, the NCEP Coupler, can be built independently of any 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, prompts the user for the desired type of parallelism (such as serial, shared memory, distributed memory, or hybrid), and then 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:

setenv HWRF 1
setenv WRF_NMM_CORE 1
setenv WRF_NMM_NEST 1
setenv WRFIO_NCD_LARGE_FILE_SUPPORT 1

and for IBM AIX builds add:

setenv IBM_REDUCE_BUG_WORKAROUND 1

In Bash Shell use the commands:

export HWRF=1
export WRF_NMM_CORE=1
export WRF_NMM_NEST=1
export WRFIO_NCD_LARGE_FILE_SUPPORT=1

and for IBM AIX builds add:

export IBM_REDUCE_BUG_WORKAROUND=1

These settings produce a version of WRF-NMM compatible with the HWRF system. There is a bug in the IBM MPI implementation. Some MPI processes will get stuck in MPI_Reduce and not return until the PREVIOUS I/O server group finishes writing. When the environment variable IBM_REDUCE_BUG_WORKAROUND is set to 1, a workaround is used that replaces the MPI_Reduce call with many MPI_Send and MPI_Recv calls that perform the sum on the root of the communicator.

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 ${SCRATCH}/HWRF/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 listed below.

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 34 options. 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, 11, 15 or 19 (shown below) are recommended.

3. Linux x86_64, PGI compiler with gcc (dmpar)
7. Linux x86_64, PGI compiler with pgcc, SGI MPT (dmpar)
11. Linux x86_64, PGI accelerator compiler with gcc (dmpar)
15. Linux x86_64 i486 i586 i686, ifort compiler with icc (dmpar)
19. Linux x86_64 i486 i586 i686, ifort compiler with icc, SGI MPT (dmpar)

The configure step for the WRF model 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 the *csh/tcsh* shell this can be done with the command:

./compile nmm_real |& tee build.log

For the *ksh/bash* shell use the command:

./compile nmm_real 2>&1 | tee build.log

In both cases, this sends the standard out and the standard error to both the file *build.log* and to the screen.

The approximate compile time varies according to the system being used. On IBM AIX machines, the compiler optimization significantly slows down the build time and it typically takes at least half an hour to complete. On most Linux systems, the WRF model typically compiles in around ten minutes.

It is important to note that the commands *./compile -h* and *./compile* produce a listing of all of the available compile options, but 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 listed below 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

http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/V3/users_guide_nmm_chap1-7.pdf

2.7 Building HWRF-Utilities

The hwrf-utilities directory consists of an eclectic collection of source code and libraries. The libraries, which are provided in support of the POM-TC and the GFDL Vortex Tracker, include the BACIO, BLAS, BUFR, SIGIO, SFCIO, SP and W3 libraries. In addition to these libraries, this component includes the source code 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 *${SCRATCH}/HWRF/WRFV3*. In csh/tcsh, the variables may be set with the commands:

setenv NETCDF /absolute_path_to_appropriate_netCDF_library/
setenv WRF_DIR ${SCRATCH}/HWRF/WRFV3

For the ksh/bash shells, use:

export NETCDF=/absolute_path_to_appropriate_netCDF_library/
export WRF_DIR=${SCRATCH}/HWRF/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 the code 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 the IBM, only one choice is available.

1. *AIX (dmpar)*

For Linux six options are available.

1. *Linux x86_64, PGI compiler w/LAPACK (dmpar)*
2. *Linux x86_64, PGI compiler w/LAPACK, SGI MPT (dmpar)*
3. *Linux x86_64, Intel compiler w/MKL (dmpar)*
4. *Linux x86_64, Intel compiler w/MKL, SGI MPT (dmpar)*
5. *Linux x86_64, Intel compiler w/LAPACK (dmpar)*
6. *Linux x86_64, Intel compiler w/LAPACK, SGI MPT (dmpar)*

For the PGI compiler, pick options 1 or 2. For Intel builds, pick option 3 or 4 if your compiler includes the MKL libraries, and option 5 or 6 if it does not.

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 build.log

For the *ksh/bash* shell, use the command:

./compile 2>&1 | tee build.log

To remove all object files, type:

./clean

To conduct a complete clean which removes **ALL** build 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 39 executables in the directory exec/.

diffwrf_3dvar.exe*
diffwrf_3dvar_2d.exe*
grbindex.exe*
hwrf_anl_4x_step2.exe*
hwrf_anl_4x_step2_2d.exe*
hwrf_anl_bogus_10m.exe*
hwrf_anl_bogus_10m_2d.exe*
hwrf_anl_cs_10m.exe*
hwrf_anl_cs_10m_2d.exe*
hwrf_create_nest_1x_10m.exe*
hwrf_create_nest_1x_10m_2d.exe*
hwrf_create_trak_fnl.exe*
hwrf_create_trak_fnl_2d.exe*
hwrf_create_trak_guess.exe*
hwrf_create_trak_guess_2d.exe*
hwrf_data_remv.exe*
hwrf_guess.exe*
hwrf_guess_2d.exe*
hwrf_inter_2to1.exe*
hwrf_inter_2to1_2d.exe*
hwrf_inter_2to2.exe*

hwrf_inter_2to6.exe*
hwrf_inter_2to6_2d.exe*
hwrf_inter_4to2.exe*
hwrf_inter_4to2_2d.exe*
hwrf_inter_4to6.exe*
hwrf_inter_4to6_2d.exe*
hwrf_merge_nest_4x_10m2_2d.exe*
hwrf_merge_nest_4x_step12_3n.exe*
hwrf_merge_nest_4x_step1_2d.exe*
hwrf_merge_nest_4x_step2_2d.exe*
hwrf_pert_ct1.exe*
hwrf_pert_ct_2d.exe*
hwrf_split1.exe*
hwrf_split_2d.exe*
hwrf_swcorner_dynamic.exe*
hwrf_swcorner_dynamic_2d.exe*
hwrf_wrfout_newtime.exe*
wgrib.exe*

In addition, it will create 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
libsigio_i4r4.a – SIGIO library built with –i4 –r4 flags
libsp_i4r8.a - SP library built with -i4 -r8 flags
libsp_i4r4.a - SP library built with -i4 -r4 flags
libw3_i4r8.a - W3 library built with -i4 -r8 flags
libw3_i4r4.a - W3 library built with -i4 -r4 flags

These libraries will be 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 only the libraries by typing the command below.

./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, and W3. On platforms that lack the ESSL mathematical libraries, typically anything other than IBM AIX machines, a fourth library (BLAS) is required. All 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 ${SCRATCH}/HWRF/hwrf-utilities/libs/
setenv LIB_SP_PATH ${SCRATCH}/HWRF/hwrf-utilities/libs/
setenv LIB_SFCIO_PATH ${SCRATCH}/HWRF/hwrf-utilities/libs/

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

export LIB_W3_PATH=${SCRATCH}/HWRF/hwrf-utilities/libs/
export LIB_SP_PATH=${SCRATCH}/HWRF/hwrf-utilities/libs/
export LIB_SFCIO_PATH=${SCRATCH}/HWRF/hwrf-utilities/libs/

In addition to the three previous libraries, POM-TC requires linear algebra routines from the BLAS library. When building POM-TC on an IBM platform, the build will automatically use the ESSL library, which includes highly optimized versions of some of the BLAS routines. 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 ${SCRATCH}/HWRF/hwrf-utilities/libs/

For the csh/tcsh shells, and for the ksh/bash shells the path can be set with:

export LIB_BLAS_PATH=${SCRATCH}/HWRF/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, and SFCIO, and for Linux, the BLAS libraries. It concludes by asking the user to choose a configuration supported by current machine architecture.

For the IBM, only one choice is available:

1. AIX (dmpar)

For Linux, the options are:

1. *Linux x86_64, PGI compiler (dmpar)*
2. *Linux x86_64, PGI compiler, SGI MPT (dmpar)*
3. *Linux x86_64, Intel compiler (dmpar)*
4. *Linux x86_64, Intel compiler, SGI MPT (dmpar)*

After selecting the desired compiler option, 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

For the *ksh/bash* shell, use the command:

./compile 2>&1 | 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, thirteen executables are created in *ocean_exec/*.

1 gfdl_date2day.exe
2 gfdl_day2date.exe
3 gfdl_find_region.exe
4 gfdl_getsst.exe
5 gfdl_ocean_eastatl.exe
6 gfdl_ocean_eastpac.exe
7 gfdl_ocean_ext_eastatl.exe
8 gfdl_ocean_united.exe
9 gfdl_sharp_mcs_rf_l2m_rmy5.exe
10 hwrf_ocean_eastatl.exe
11 hwrf_ocean_eastatl_ext.exe
12 hwrf_ocean_eastpac.exe
13 hwrf_ocean_united.exe

The executables *hwrf_ocean_united.exe, hwrf_ocean_eastpac.exe*, *hwrf_ocean_eastatl.exe*, and *hwrf_ocean_eastatl_ext.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 build the HWRF utilities before building the vortex tracker.

Again, assuming that the directory structure is the same as that proposed in Section 2.3 for csh/tcsh, the library paths can be set with:

setenv LIB_W3_PATH ${SCRATCH}/HWRF/hwrf-utilities/libs/
setenv LIB_BACIO_${SCRATCH}/HWRF/hwrf-utilities/libs/

For the ksh/bash shells, the library paths can be set using:

export LIB_W3_PATH=${SCRATCH}/HWRF/hwrf-utilities/libs/
export LIB_BACIO_PATH=${SCRATCH}/HWRF/hwrf-utilities/libs/

2.9.2 Configure and Compile

To configure the vortex tracker for compilation, from within the tracker 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 and BACIO libraries. It concludes by asking the user to choose a configuration supported by current machine architecture.

For the IBM, only one choice is available:

1. AIX (serial)

For Linux, the options are:

1. *Linux x86_64, PGI compiler (serial)*
2. *Linux x86_64, Intel compiler (serial)*
3. *Linux x86_64, Intel compiler super debug (serial)*
4. *Linux x86_64, PGI compiler, SGI MPT (serial)*
5. *Linux x86_64, Intel compiler, SGI MPT (serial)*

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

For the *ksh/bash* shell use the command:

./compile 2>&1 | tee tracker.log

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, three executables are created in the directory *trk_exec/*.

hwrf_gettrk.exe
hwrf_tave.exe
hwrf_vint.exe

2.10 Building the 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 the IBM, only one choice is available:

1. AIX (dmpar)

For Linux, the options are:

1. *Linux x86_64, PGI compiler (dmpar)*
2. *Linux x86_64, Intel compiler (dmpar)*
3. *Linux x86_64, Intel compiler, SGI MPT (dmpar)*

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

For the *ksh/bash* shell, use the command:

./compile 2>&1 | tee coupler.log

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.

Set up the build environment for WPS by setting the *WRF_DIR* environment variable. For csh/tcsh, use:

setenv WRF_DIR ${SCRACH}/HWRF/WRFV3/

For bash/ksh, use:

export WRF_DIR=${SCRATCH}/HWRF/WRFV3/

In order to run the WRF Domain Wizard (http://wrfportal.org/DomainWizard.html), 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

Select the appropriate "dmpar" option for your architecture and compiler choice. If you plan to use GRIB2 data, you will also need to select a build option that supports GRIB2 I/O. This will generate the configure resource file.

On IBM AIX computers the listed options are:

1. *AIX (serial)*
2. *AIX (serial_NO_GRIB2)*
3. *AIX (dmpar)*
4. *AIX (dmpar_NO_GRIB2)*

Choose 3 if you want GRIB 2 support, and 4 if you don't. On Linux computers, there are 30 listed options. The first 20 are the most relevant to HWRF.

1. *Linux x86_64, PGI compiler (serial)*
2. *Linux x86_64, PGI compiler (serial_NO_GRIB2)*
3. *Linux x86_64, PGI compiler (dmpar)*
4. *Linux x86_64, PGI compiler (dmpar_NO_GRIB2)*
5. *Linux x86_64, PGI compiler, SGI MPT (serial)*
6. *Linux x86_64, PGI compiler, SGI MPT (serial_NO_GRIB2)*
7. *Linux x86_64, PGI compiler, SGI MPT (dmpar)*
8. *Linux x86_64, PGI compiler, SGI MPT (dmpar_NO_GRIB2)*
9. *Linux x86_64, IA64 and Opteron (serial)*
10. *Linux x86_64, IA64 and Opteron (serial_NO_GRIB2)*
11. *Linux x86_64, IA64 and Opteron (dmpar)*
12. *Linux x86_64, IA64 and Opteron (dmpar_NO_GRIB2)*
13. *Linux x86_64, Intel compiler (serial)*
14. *Linux x86_64, Intel compiler (serial_NO_GRIB2)*
15. *Linux x86_64, Intel compiler (dmpar)*
16. *Linux x86_64, Intel compiler (dmpar_NO_GRIB2)*
17. *Linux x86_64, Intel compiler, SGI MPT (serial)*
18. *Linux x86_64, Intel compiler, SGI MPT (serial_NO_GRIB2)*
19. *Linux x86_64, Intel compiler, SGI MPT (dmpar)*
20. *Linux x86_64, Intel compiler, SGI MPT (dmpar_NO_GRIB2)*

Select the appropriate "dmpar" option for your choice of compiler. To get help about compilation, type:

./compile -h

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

./compile |& tee wps.log

For the *ksh/bash* shell, use the command:

./compile 2>&1 | 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*; and *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 WRF-NMM User's Guide.

2.12 Building UPP

The NCEP Unified Post-Processor 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 *UPP/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 *hwrf-utilities/libs* directory.

2.12.1 Set Environment Variables

The UPP requires the WRF I/O API libraries to successfully build. 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 uses the ***WRF_DIR*** environment variable to define the path to WRF. The path variable ***WRF_DIR*** must therefore be set to the location of the WRF root directory.

In addition to setting the path variable, building UPP for use with HWRF requires setting the environment variable ***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):

```
setenv HWRF 1
setenv WRF_DIR ${SCRACH}/HWRF/WRFV3/
```

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

```
export HWRF=1
export WRF_DIR=${SCRATCH}/HWRF/WRFV3/
```

2.12.2 Configure and Compile

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

```
./configure
```

to generate the UPP configure file. The configure script will complain if the ***WRF_DIR*** path has not 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. *AIX with IBM Make ($DEBUG ignored) (serial)*
2. *AIX with IBM Make ($DEBUG ignored) (dmpar)*
3. *AIX with GNU Make (serial)*
4. *AIX with GNU Make (dmpar)*

Any of the “dmpar” options (2 or 4) are compatible with the HWRF system. For the LINUX operating systems, there are eight options. We show the six relevant options here.

1. *Linux x86_64, PGI compiler (serial)*
2. *Linux x86_64, PGI compiler (dmpar)*
3. *Linux x86_64, Intel compiler (serial)*
4. *Linux x86_64, Intel compiler (dmpar)*
5. *Linux x86_64, Intel compiler, SGI MPT (serial)*
6. *Linux x86_64, Intel compiler, SGI MPT (dmpar)*

Any of the "dmpar" options (2,4, or 6) are compatible with the HWRF system. The configuration script will generate the configure file *configure.upp*. If necessary, the *configure.upp* file can be modified to change the default compile options and paths.

To compile UPP, enter the command (csh/tsch):

./compile |& tee build.log

For the *ksh/bash* shell, use the command:

./compile 2>&1 | tee build.log

This command should create eight UPP libraries in *lib/* (libCRTM.a, libbacio.a, libip.a, libmersenne.a, libsfcio.a, libsigio.a, libsp.a, and libw3.a), and three UPP executables in *bin/* (*unipost.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 *configure.upp*, type:

./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 Users Guide, and for complete details on building and running the UPP, see the WRF-NMM User's Guide, which can be downloaded at:

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):

setenv HWRF 1
setenv WRF_DIR ${SCRATCH}/HWRF/WRFV3

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

export HWRF=1
export WRF_DIR=${SCRATCH}/HWRF/WRFV3

The additional environment variable *LAPACK_PATH* may be needed on some systems. Typically, the environment variable *LAPACK_PATH* needs only to be set on Linux systems without a vender 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 vender-provided version of LAPACK that links automatically with the compiler. Problems with the vender-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 is:

setenv LAPACK_PATH *$MKL*

for bash/ksh it is:

export LAPACK_PATH=*$MKL*

To build GSI for HWRF, change into the GSI directory and issue the configure command:

./configure

Choose one of the configure options listed.

For example, on IBM computers the listed options are as follows.

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 as follows.

1. *Linux x86_64, PGI compilers (pgf90 & pgcc) (dmpar,optimize)*
2. *Linux x86_64, PGI compilers (pgf90 & pgcc), SGI MPT (dmpar,optimize)*
3. *Linux x86_64, PGI compilers (pgf90 & gcc) (dmpar,optimize)*
4. *Linux x86_64, Intel compiler, EMC OPTIONS (dmpar,optimize)*
5. *Linux x86_64, Intel compiler (ifort & icc) (dmpar,optimize)*
6. *Linux x86_64, Intel compiler (ifort & icc), SGI MPT (dmpar,optimize)*
7. *Linux x86_64, Intel/gnu compiler (ifort & gcc) (dmpar,optimize)*

Select the appropriate "dmpar" option for your platform and compiler.

After selecting the proper option, run the compile script (csh/tcsh):

./compile |& tee build.log

For the *ksh/bash* shell, use the command:

./compile 2>&1 | 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. Another executable, *ssrc.exe,* which is used to convert a binary data file's endianness, can be found in the *util/test* directory.

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 User's Guide.

http://www.dtcenter.org/com-GSI/users/docs/index.php

Chapter 3: HWRF Preprocessing System

3.1 Introduction

The WRF WPS is a set of three programs whose collective role is to prepare input to real_nmm program for real data simulations. For general information about working with WPS, view the WRF-NMM documentation at

http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/V3/users_guide_nmm_chap1-7.pdf

In the Community 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 wrapper scripts are used to preprocess data for HWRF: *hwrfdomain_wrapper, geogrid_wrapper, ungrib_wrapper and metgrid_wrapper*. These wrapper scripts drive the four corresponding low-level scripts, *hwrfdomain.ksh, geogrid.ksh, ungrib.ksh and metgrid.ksh*, respectively. Script *hwrfdomain.ksh* defines the location of the parent domain; *geogrid.ksh* interpolates static geographical data to the three HWRF domains; *ungrib.ksh* extracts meteorological fields from GRIB-formatted files and writes the fields to intermediate files, and *metgrid.ksh* horizontally interpolates the meteorological fields extracted by *ungrib.ksh* to the HWRF parent domain. The wrapper scripts can be found in *${SCRATCH}/HWRF/hwrf-utilities/wrapper_scripts*

Before running the wrappers, users need to edit file *global_vars.ksh* and make sure that the list of global variables is correctly customized for the desired HWRF run. Note if the user wants a wrapper script to output detailed debug information in the standard output (stdout), he can define an environment variable, DEBUG, in the wrapper script by adding the following statement:

export DEBUG=1

3.2.1 *hwrfdomain_wrapper*

Before running *hwrfdomain_wrapper,* check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix):

HWRF_SCRIPTS
SID
START_TIME
TCVITALS
DOMAIN_DATA

Then run the wrapper script in *hwrf-utilities/wrapper_scripts* using the command: *hwrfdomain_wrapper,* which, in turn, will run the low-level script *hwrf-utilities/scripts/hwrfdomain.ksh.*

Overview of script *hwrfdomain.ksh:*

1. Initialize the function library and check to see if all the environment variables are set.
2. Create a working directory, which is *${DOMAIN_DATA}/messages*.
3. Create the *tcvital* and *tcvitals.as* files.
4. Get the storm center latitude and longitude from the TCVitals record.
5. Compute the reference latitude and longitude for the HWRF parent domain using the storm center.
6. Test to make sure that the reference longitude is no more than 5 degrees away from the storm center longitude.
7. Output the storm center to file storm.center.
8. Output the center of the parent 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.
3. *tcvital*: file that contains the storm's tcvital record.
4. *tcvital.as*: file that contains the storm's tcvital record.

Status Check

If the four output files are found in the directory of *${DOMAIN_DATA}/*messages, the wrapper script *hwrfdomain_wrapper* and the low-level script *hwrfdomain.ksh* have finished successfully.

3.2.2 *geogrid_wrapper*

Before running *geogrid_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
WPS_ROOT
DOMAIN_DATA
GEOGRID_CORES
FCST_LENGTH
GEOG_DATA_PATH
HWRF_UTILITIES_ROOT
ATMOS_DOMAINS
IO_FMT

First use the *qrsh* command to connect to the computer's remote computation nodes (see Section 1.6). Note the number of processors the user should connect to is defined in *global_vars.ksh* as *GEOGRID_CORES.*

Then run the wrapper script in *hwrf-utilities/wrapper_scripts* using the command: *geogrid_wrapper,* which, in turn, will run the low-level script *hwrf-utilities/scripts/geogrid.ksh*.

Overview of script *geogrid.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the *geogrid.exe* executable exists.
2. Create the working directory *${DOMAIN_DATA}/geoprd.*
3. Create the namelist and copy the geogrid table file.
4. Run *geogrid.exe* to generate the geographical data.

Output files in directory *${DOMAIN_DATA}/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.
3. geo_nmm_nest.l02.nc: static geographical data that covers the parent domain, with a grid spacing of 0.02 degrees. Status check

If "Successful completion of program geogrid.exe" is found in the standard output files, *${DOMAIN_DATA}/geoprd/geogrid.log.**, the wrapper script *geogrid_wrapper* and the low-level script *geogrid.ksh* have successfully finished.

3.2.3 *ungrib_wrapper*

Before running *ungrib_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
WPS_ROOT
START_TIME
FCST_LENGTH
HWRF_UTILITIES_ROOT
DOMAIN_DATA
GFS_GRIDDED_DIR

Then run the wrapper script in *hwrf-utilities/wrapper_scripts* using the command: *ungrib_wrapper,* which, in turn, will run the low-level script *hwrf-utilities/scripts/ungrib.ksh*.

Overview of script *ungrib.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the *ungrib.exe* executable exists.
2. Create and enter a work directory *${DOMAIN_DATA}/ungribprd.*
3. Create the namelist used by *ungrib.exe.*
4. Copy the ungrib table.
5. Link the grib files.
6. Run *ungrib*.

Output files in directory *${DOMAIN_DATA}/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.wps* 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_DATA}/ungribprd/ungrib.log*, the wrapper script *ungrib_wrapper* and the low-level script *ungrib.ksh* have successfully finished.

3.2.4 *metgrid_wrapper*

Before running *metgrid_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
WPS_ROOT
DOMAIN_DATA
METGRID_CORES
START_TIME
FCST_LENGTH
GEOG_DATA_PATH
HWRF_UTILITIES_ROOT
IO_FMT

Next use the *qrsh* command to connect to the computer's remote computation nodes (see Section 1.6). Note the number of processors the user should connect to is defined in *global_vars.ksh* as *METGRID_CORES.*

Then run the wrapper script in *hwrf-utilities/wrapper_scripts* using the command: *metgrid_wrapper,* which, in turn, will run the low-level script *hwrf-utilities/scripts/metgrid.ksh*.

Overview of script *metgrid.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the *metgrid.exe* executable exists.
2. Create and enter a work directory *${DOMAIN_DATA}/metgridprd.*
3. Create the namelist used by *metgrid.exe.*
4. Copy the metgrid table.
5. Copy in the geogrid output files.
6. Copy in the ungrib output files.
7. Run *metgrid.exe.*

Output files in directory *${DOMAIN_DATA}/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_DATA}/metgridprd/metgrid.log*, the wrapper script *metgrid_wrapper* and the low-level script *metgrid.ksh* have successfully finished.

3.1 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,*${GEOG_DATA_PATH}*)
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.
geo_nmm_nest.l02.nc: static geographical data that covers the parent domain, with a grid spacing of 0.02 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 WPS 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-h 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 has the capability of setting up the HWRF domain, and running gogrid, ungrib and metgrid. For more information about the WRF Domain Wizard, see

http://www.wrfportal.org/DomainWizard.html

Chapter 4: Vortex Initialization

4.1 Overview

The initial vortex is often not realistically represented in a high-resolution mesoscale model when the initial conditions are generated from a low-resolution global model, such as GFS. To address this issue, HWRF employs a sophisticated algorithm to initialize the hurricane vortex. The HWRF initialization processes are illustrated in Figure 4.1.

Before the vortex initialization scripts are run, the user needs to run the wrapper script *real_wrapper* to produce the preliminary initial and boundary conditions for the parent domain. These preliminary initial conditions are then further modified by the vortex initialization and data assimilation procedures. To do that, WRF is run twice, once using the same domains employed for the coupled model forecast (this is referred to as the analysis run and uses the wrapper script *wrfanalysis wrapper*), and once for the ghost domains (using the wrapper script *wrfghost wrapper*) (see the HWRF domains in Figure 4.2). The two WRF *runs* are both 90-second runs. The purpose of these two WRF runs is to obtain the initial fields on the analysis (*wrfanl d02, wrfanl d03*) and ghost domains (*wrfghost d02, wrfghost d03*), respectively. These files are used in the subsequent vortex initialization and GSI procedures. The ghost domains are only used for initialization and are not employed in the actual forecast. The wrapper script *track_analysis_wrapper* is used to run the post-processor and GFDL vortex tracker on the WRF analysis output to identify the center of the storm in the GFS analysis, which is used later in the vortex relocation 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 according to the TCVitals. 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. This step is run only for cycled cases.

2. Stage 2: The preliminary initial condition generated from the GFS data using WPS, real, and the WRF ghost and analysis runs 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 when the forecast storm is weaker than the observed one) 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.

After the 3 stages, the new vortex is created. Next, GSI (a 3D-VAR data assimilation system) is used to assimilate observational upper-level and surface conventional observations into the HWRF atmospheric model's initial condition background fields. Satellite radiance data are not used in the 2012 HWRF forecasts. GSI is run independently in the parent domain and in the inner nest of the WRF ghost run. After the two GSI runs, wrapper script *merge_wrapper* is used to generate the final initial conditions for the coupled HWRF forecast.

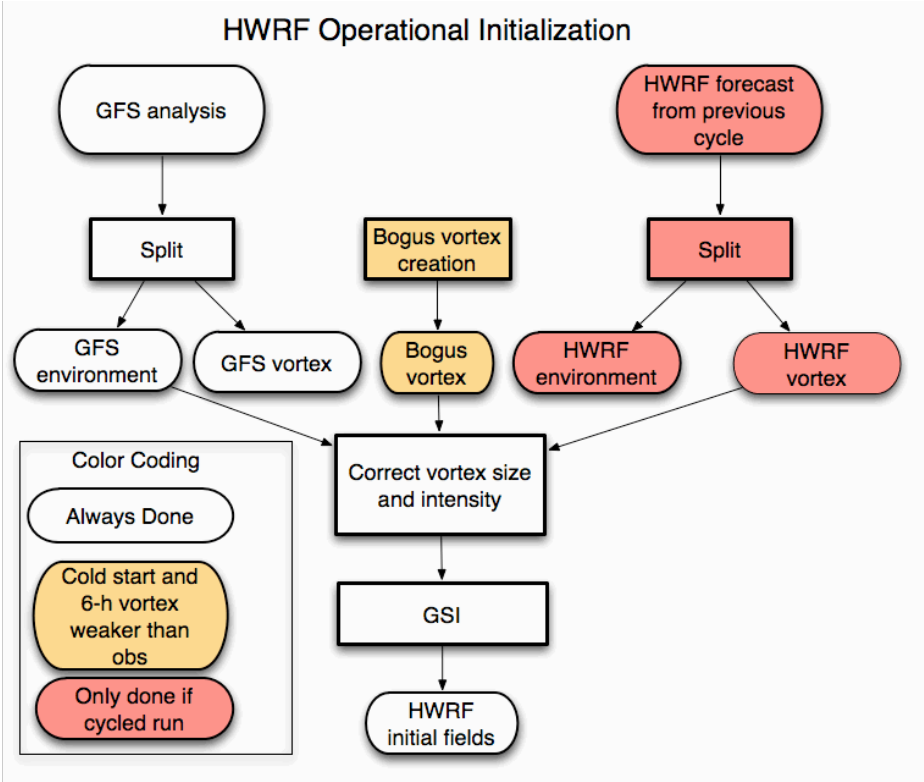

Figure 4.1. HWRF operational initialization flowchart.

4.2 Domains Used in HWRF

The following domain grids are used in HWRF vortex initialization process (Figure 4.2 and Table 4.1).

	D01	**D02**	**D03**
Grid spacing (deg)	0.18	0.06	0.02
Coupled Forecast	216x432 - 80^ox80^o	88x170 - 11^ox10^o	154x272 - 6.0^ox5.5^o
Analysis run	216x432 - 80^ox80^o	88x170 - 11^ox10^o	154x272 - 6.0^ox5.5^o
Ghost run	216x432 - 80^ox80^o	211x410 - 24^ox24^o	529x988 - 20^ox20^o
3X domain			748x1504 - 30^ox30^o

Table 4.1. Resolution and number of grid points for the HWRF atmospheric grids.

Notes

1. Although during the WRF ghost runs output data is generated on domains one and two (see Section 4.3.3), these files are not used later.
2. The ghost domain three is mainly used for GSI data analysis. This domain is not used during the model integration. GSI is run twice, once for the outer domain, and once for the ghost domain three.
3. The 3X domain is used to remove the GFS vortex, then extract and correct the storm vortex from the previous 6-hr forecast. 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 and only exists inside the vortex initialization code.
4. The ocean model grid placement depends on the position of the observed and 72-hr NHC forecast of the observed storm. As an example, in Figure 4.2 the cyan box shows the "united" ocean model domain grid, which is used in the forecast of hurricane Irene initialized at 12 UTC on 23 August 2011. The "united" ocean model grid covers the area 10N-47.5N and 98.5W-60W.

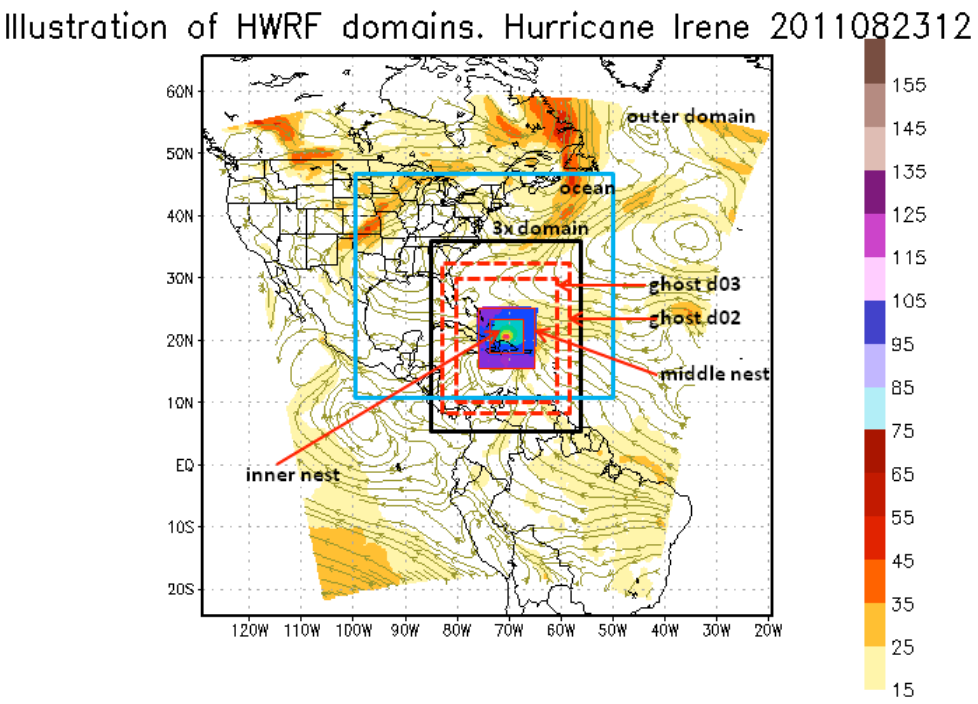

Figure 4.2. The domains used in HWRF.

4.3 How to Run the Vortex Initialization Using Scripts

The HWRF vortex initialization scripts come in the tarfile *hwrfv3.4a utilities.tar* and, following the procedures outlined in Chapters 1 and 2, will be expanded in the directories *${SCRATCH}/HWRF/hwrf-utilities/wrapper_scripts* and *${SCRATCH}/HWRF/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 are 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 Section 1.6.

4.3.1 *real_wrapper*

Before running *real_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
WRF_ROOT
HWRF_UTILITIES_ROOT
REAL_CORES
FCST_LENGTH
GFS_GRIDDED_DIR
DOMAIN_DATA

Next use the *qrsh* command to connect to the computer's remote computation nodes (see Section 1.6). Note the number of processors the user should connect to is defined in *global_vars.ksh* as *REAL_CORES.*

Then run the wrapper script *hwrf-utilities/wrapper_scripts/real_wrapper* by typing the name of the script in the terminal, which, in turn, will run the low-level script *hwrf-utilities/scripts/real.ksh*. This will read in the output from the WPS executable *metgrid.exe* and generate *wrfinput_d01*, *wrfbdy_d01* and *fort.65*.

Overview of script *real.ksh*:

1. Initialize the function library and check to see if all the environment variables are set and the executables *real_nmm.exe, wgrib* and *hwrf_swcorner_dynamic.exe* exist.
2. Create and enter the work directory *realprd.*
3. Link input and fix files.
4. Run *hwrf_swcorner_dynamic.exe* to calculate the nest domain location.
5. Generate the namelist.
6. Run *real_nmm.exe* to generate initial and boundary conditions. A high-resolution sea-mask data file (*fort.65*) for the entire outer domain is also generated. It is later used by the coupler.

Note: to run *real.ksh* successfully, users should set the computer's stacksize to be equal to or larger than 2 GB. To do this:

In bash shell, use the command *ulimit –s 204800.*
In C-shell, use the command *limit stacksize 2048m.*

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

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

4.3.2 *wrfanalysis_wrapper*

Before running *wrfanalysis_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
WRF_ROOT
HWRF_UTILITIES_ROOT
START_TIME
ATMOS_DOMAINS
FCST_LENGTH
FCST_INTERVAL
DOMAIN_DATA

Note that in the wrapper script *wrfanalysis_wrapper*, the following two variables are defined.

WRF_MODE = analysis

WRF_CORES =4

WRF_MODE should not be altered but WRF_CORES can be customized to set the number of processors for the WRF analysis run.

Next, use the *qrsh* command to connect to the computer's remote computation nodes (see 1.6). Note the number of processors the user should connect to is defined by *WRF_CORES.*

Then run the wrapper script *hwrf-utilities/wrapper_scripts/wrfanalysis_wrapper*, which, in turn, will run the low-level script *hwrf-utilities/scripts/wrf.ksh*. This will make a 90 second run of *wrf.exe* and generate an analysis output for the middle and inner nest domains.

Overview of script *wrf.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the *wrf.exe* and *hwrf_swcorner_dynamic.exe* executables exist.
2. Create and enter the work directory *wrfanalysisprd.*
3. Link the fix data, initial condition and boundary conditions.
4. Run *hwrf_swcorner_dynamic.exe* to calculate the istart and jstart values for the middle nest domain grid and create a namelist.input file.
5. Run *wrf.exe*.

Output files in the directory *${DOMAIN_DATA}/wrfanalysisprd*

"*wrfanl_d02**" and "*wrfanl_d03**" are two "analysis" files for the HWRF middle and inner nest domains.

Status Check

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

4.3.3 *wrfghost_wrapper*

Before running *wrfghost_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
WRF_ROOT
HWRF_UTILITIES_ROOT
START_TIME
ATMOS_DOMAINS
FCST_LENGTH
FCST_INTERVAL
DOMAIN_DATA

Note that in the wrapper script *wrfghost_wrapper*, the following two variables are defined.

WRF_MODE = ghost

WRF_CORES =12

WRF_MODE should not be altered but WRF_CORES can be customized to set the number of processors for the WRF ghost run.

Next use the *qrsh* command to connect to the computer's remote computation nodes (see Section 1.6). Note the number of processors the user should connect to is defined as *WRF_CORES.*

After the connection between the user and the computation nodes established, the user can run the wrapper script *hwrf-utilities/wrapper_scripts/wrfghost_wrapper* by typing the name of the wrapper script, which, in turn, will run the low-level script *hwrf-utilities/scripts/wrf.ksh*.

This will make a 90 second run of *wrf.exe* and generate output for the middle and inner "ghost" domains (see Figure 4.2). These "ghost" domains are used only in GSI data assimilation procedures (see Section 4.3.9).

Overview of script *wrf.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the *wrf.exe* and *hwrf_swcorner_dynamic.exe* executables exist.
2. Create and enter the work directory *wrfghostprd.*
3. Link the fix data, initial condition, and boundary conditions.
4. Run *hwrf swcorner dynamic.exe* to calculate the istart and jstart values for the middle "ghost" domain grid and create a namelist.input file.
5. Run *wrf.exe.*

Output files in the directory *${DOMAIN_DATA}/wrfghostprd*

"*ghost d02**" and "*ghost d03**", two "analysis" files for the HWRF middle and inner "ghost" domains.

Status Check

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

4.3.4 *track_analysis_wrapper*

Before running *track_analysis_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
UPP_ROOT
HWRF_UTILITIES_ROOT
TRACKER_ROOT
DOMAIN_DATA
START_TIME
SID

Then run *track_analysis_wrapper* by typing the name of the wrapper script, which, in turn, will run the low-level script *track_analysis.ksh*. This script will run *unipost.exe* and *copygb.exe* to interpolate the WRF analysis run (Section 4.3.2) *wrfout_d01* horizontally to a regular lat/lon grid and vertically to isobaric levels, and to output a file in GRIB format. Then the GFDL vortex tracker is run to identify the center of the storm (see Figure 4.4).

Overview of script *track_analysis.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.

2. Create the work directory, enter it and copy the unipost fix files and the *wrfout_d01* file from the WRF analysis 90 s run.
3. Run *hwrf wrfout newtime.exe* to change the time stamp in this *wrfout d01* to t=0. This is needed because the GFDL vortex tracker requires tracking from the beginning of the forecast.
4. Run *unipost.exe* to post-process the wrfout_d01 file.
5. Run *copygb.exe* to horizontally interpolate the unipost.exe output to a regular lat/lon grid.
6. Run the GFDL vortex tracker.

Output files in directory ${DOMAIN_DATA}/trkanalysisprd
gfs-anl-fix.atcfunix (storm center at initial time in the WRF analysis run output)

Status Check:
If "failed" is not found in the standard output (*stdout*) and file *gfs-anl-fix.atcfunix* exists, the wrapper script *track_analysis_wrapper* and the low-level script *track_analysis.ksh* runs were successful.

4.3.5 *relocate1_wrapper*

Before running *relocate1_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix):

HWRF_SCRIPTS
HWRF_UTILITIES_ROOT
DOMAIN_DATA
CYCLE_DATA
START_TIME
IO_FMT
FCST_INTERVAL
ATCFNAME
SID

Then run the wrapper script *relocate1 wrapper* by typing its name on the terminal, which, in turn, will run the low-level script *relocate stage1 3d.ksh*. If a previous 6-hr forecast exists and the observed storm maximum wind speed is greater than 12 ms^{-1} (a cycled run), the previous forecast will be interpolated onto the 3X domain and separated into environment fields and storm vortex fields (Figure 4.5). The storm vortex fields will be adjusted. The 3X domain is about 30x30 degrees with the resolution of the inner nest domain and is centered based on the NHC storm message data (see Figure 4.2). Be aware that the 3X ($30x30^{o}$) domain is sometimes referred to in the source code or in executable names as the 4X domain. This is a legacy from the 2011 configuration of HWRF in which this domain had $40x40^{o}$dimensions.

Note in the script *relocate_stage1.ksh*, the gnu version of the command "*date*" is used.

Overview of script *relocate_stage1_3d.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.
2. Create the work directory, enter it and copy fixed files and namelist.
3. Check to see if the previous cycle forecast exists and the storm intensity is greater than 12 ms^{-1}; if not, exit.
4. Run *diffwrf_3dvar.exe* to convert the previous cycle forecast output *wrfout_d01*, *wrfout_d02* and *wrfout_d03* into unformatted data files *old_hwrf_d01, old_hwrf_d02* and *old_hwrf_d03* respectively.
5. Run *merge_nest_4x_step12_3n.exe* to merge *wrfout_d01*, *wrfout_d02* and *wrfout_d03* onto 3X domain and produce a file containing the merged data: *data_4x_hwrf*.
6. Run *hwrf_create_trak_guess.exe* to produce a guess track (0,3,6,9 hour) for the current forecast using previous cycle forecast track.
7. Run *wrf_split1.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.
8. *Run hwrf_pert_ct1.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_DATA}/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 wrapper script *relocate1_wrapper* and the low-level script *relocate_stage1_3d.ksh* runs were successful.

4.3.6 *relocate2_wrapper*

Before running *relocate2_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
HWRF_UTILITIES_ROOT
DOMAIN_DATA
START_TIME
IO_FMT

FCST_INTERVAL
START_TIME
GFS_GRIDDED_DIR
SID

Then run the wrapper script *relocate2_wrapper* by typing its name in the terminal. This will merge the outer nest, inner nest, and ghost nest domain initial fields onto a 3X domain grid. The merged fields will be separated into environment fields and storm vortex (see Figure 4.6).

Overview of script *relocate_stage2.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.
2. Enter the work directory and copy needed fix files and namelist.
3. Run *diffwrf_3dvar.exe* to convert *wrfinput_d01*, *wrfanl_d02*, *wrfanl_d03* and *wrfghost_d02* (copied from *wrfghost_d03*) into binary files *new_gfs_d01, new_gfs_d02*, *new_gfs_d03* and *new_ght_d02*, respectively.
4. Run *hwrf_create_nest_1x_10m.exe* to rebalance the inner nest domain data. This will generate the data file *new_gfs_d01* that contains the rebalanced outer and inner domain data.
5. Run hwrf_create_trak_fnl.exe to create *trak.fnl.all_gfs*, a guess track file from *atcfunix*.
   For example, for a forecast of hurricane Irene starting at 12 UTC on 8/23/2011, the storm ID is 09L, file *atcfunix* shows the following storm information:

   AL, 09, 2011082312, 03, HWRF, 000, 207N, 706W, 54, 991, XX, 34, NEQ, 0168, 0138, 0060, 0158, 0, 0, 60;
   AL, 09, 2011082312, 03, HWRF, 000, 207N, 706W, 54, 991, XX, 50, NEQ, 0081, 0000, 0000, 0052, 0, 0, 60;

   and the guess track file should be in the following form:

   72HDAS11082312 207 706 207 706 207 706 207 706 0 0 0 0 0 0 09L

   where ' 72HDAS' is a fixed field, 11082312 means 08/23/2011 12 UTC, 207 and 706 are the latitude and longitude multiplied by 10 (20.7N and 70.6W), and 09L is the storm ID.
6. Run *hwrf_merge_nest_4x_step12_3n.exe* to merge inner domain (*new_gfs_d03*) , middle domain (*new_gfs_d02*), and outer domain (*new_gfs_d01*) onto the 3X domain. This will generate the file containing the merged data on the 3X domain (*data_4x_gfs*) and a file containing sea mask and roughness length data (*roughness2*).
   Run *hwrf_split1.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
roughness2:	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 wrapper script *relocate2_wrapper* and the low-level script *relocate_stage2.ksh* runs were successful.

4.3.7 *relocate3_wrapper*

Before running *relocate3_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
HWRF_UTILITIES_ROOT
DOMAIN_DATA
USE_GSI
FCST_INTERVAL
ATMOS_DOMAINS
IO_FMT

Then run the wrapper script *relocate3_wrapper* by typing its name in a terminal, which in turn will run the low-level script *relocate_stage3.ksh*. This will create a new storm vortex by adjusting the previous cycle 6-hr 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 Figures 4.7 and 4.8).

Overview of script *relocate_stage3.ksh*:

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.
2. Enter the work directory.
3. For cold start runs (previous cycle 6-hr 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 3X domain grid. This will generate *new_data_4x*.
4. For cycled runs (previous cycle 6-hr forecast exists and the storm's maximum wind is larger than or equal to 12 ms^{-1}),
   a. *Run hwrf_anl_4x_step2.exe* to adjust the storm vortex obtained in stage 1 (*storm_pert_new*) and add the new storm vortex to the environment flow (*gfs_env*) on the 3X domain grid. This will produce a new file (*new_data_4x*) containing the combined environment flow and the adjusted storm vortex.

b. 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* that contains the combined environment flow and the adjusted storm vortex.

5. Run *hwrf_inter_4to6.exe* to interpolate the *new_data_4x* from the 3X domain onto the outer domain grid. This will produce the new *data_merge_d01*. In this step, 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.
6. Run *hwrf_inter_4to2.exe* to interpolate the *new_data_4x* from the 3X domain onto the ghost domain grid. This will produce the new *data_merge_2x*.
7. Run *diffwrf_3dvar.exe* to convert the unformatted *data_merge_d01* to the netCDF file *wrfinput_d01*.
8. Run *diffwrf_3dvar.exe* to convert the unformatted *data_merge_2x* to the netCDF file *wrfghost_d02*.
9. Decide if GSI will be run based on environmental variable set by the user. By default, GSI I is run for all storms as in the 2012 operational implementation.

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

wrfinput_d01: Adjusted parent domain fields that contains both the vortex and the environment
wrfghost_d02: Adjusted ghost domain fields that contains both the vortex and the environment

Status Check:
If "failed" is not found in the standard output (*stdout*) and the files listed above exist, the script *relocate3_wrapper* run was successful.

4.3.8 *gsi_wrapper*

Before running *gsi_wrapper,* check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPT
GSI_ROOT
HWRF_UTILITIES_ROOT
DOMAIN_DATA
PREPBUFR
OBS_ROOT
START_TIME
FIX_ROOT
CYCLE_DATA
BK_DIR
IO_FORMAT

GSI_CORES

Note that in the wrapper script *gsi wrapper*, an additional environment variable, DOMAIN, is defined. The wrapper has a loop and DOMAIN assumes the values wrfinput and wrfghost so that GSI can be run for both domains.

Next use the *qrsh* command to connect to the computer's remote computation nodes (see Section 1.6). Note the number of processors the user should connect to is defined as *GSI_CORES.*

Then run the wrapper script *gsi_wrapper,* which in turn will run the low-level script *run gsi.ksh* twice, one for the parent domain, the other for the ghost inner nest (Figure 4.9).

Overview of script *run_gsi.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.
2. Create and enter the work directory.
3. Check the endianness.
4. Reverse the endianness of data if using NCEP prepBUFR data (big_endian) on machines with little_endian (e.g. Linux).
5. Remove the observational data near the storm center.
6. Copy the fixed data and background analysis to the working directory.
7. Create a namelist for GSI analysis.
8. 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 the 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/users_guide/GSIUserGuide_v3.1.pdf*).

4.3.9 *merge_wrapper*

Before running *merge_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
HWRF_UTILITIES_ROOT
DOMAIN_DATA
ATMOS_DOMAINS
IO_FMT
FCST_INTERVAL
START_TIME

Then run the wrapper script *merge_wrapper* by typing its name in a terminal, which in turn will run the low-level script *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 are created from the vortex relocation stage 3.

Overview of script *merge.ksh*:
Initialize the function library and check to see if all the environment variables are set and the executables exist.

1. Create and enter the work directory.
2. Copy the input analysis files. If GSI was run, the input files for the parent and ghost domains will be from the GSI output in the directory *gsiprd,* otherwise they will be from the relocation stage 3 output in the directory *relocateprd*. The input analysis file for the 3rd domain will be from the directory *relocateprd*.
3. Copy the namelist.
4. Run *diffwrf_3dvar.exe* to convert the netCDF format *wrfinput_d01, wrfinput_d02*, *wrfinput_d03*, and *wrfghost_d02* to unformatted data files *new_hdas_d01*, *new_gfs_d02*, *new_gfs_d03*, and *new_ght_d02*, respectively.
5. 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*).
6. Run *hwrf_inter_2to1.exe* to interpolate the data in file *new_ght_d02* and *new_gfs_d03* to the inner nest domain grid. This will produce the merged data on the inner nest grid (*data_merge_d03*).
7. Run *hwrf_inter_2to2.exe* to interpolate the data in file *new_ght_d02, new_gfs_d02*, and *new_hdas_d01* to the middle nest domain grid d02. This will produce the merged data on the inner nest grid (*data_merge_d02*).
8. Run *diffwrf_3dvar.exe* to convert the unformatted files *data_merge_d01, data_merge_d02*, and data_merge_d03 to the netCDF format files *wrfinput_d01*, *wrfinput_d02*, and *wrfinput_d03, respectively.*
9. Rename *wrfinput_d02 and wrfinput_d03* to *wrfanl_d02* and *wrfanl_d03* respectively.
10. *wrfinput_d01*, *wrfanl_d02* and *wrfanl_d03* 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 middle nest domain containing the new vortex. *${YYYY-MM-DD} is the model run's initial time.
wrfanl_d03_${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.

Status Check:
If you do not see "failed" in the file *stdout* and the above mentioned output files are generated, the wrapper script *merge_wrapper* and the low-level script *merge.ksh* have run successfully.

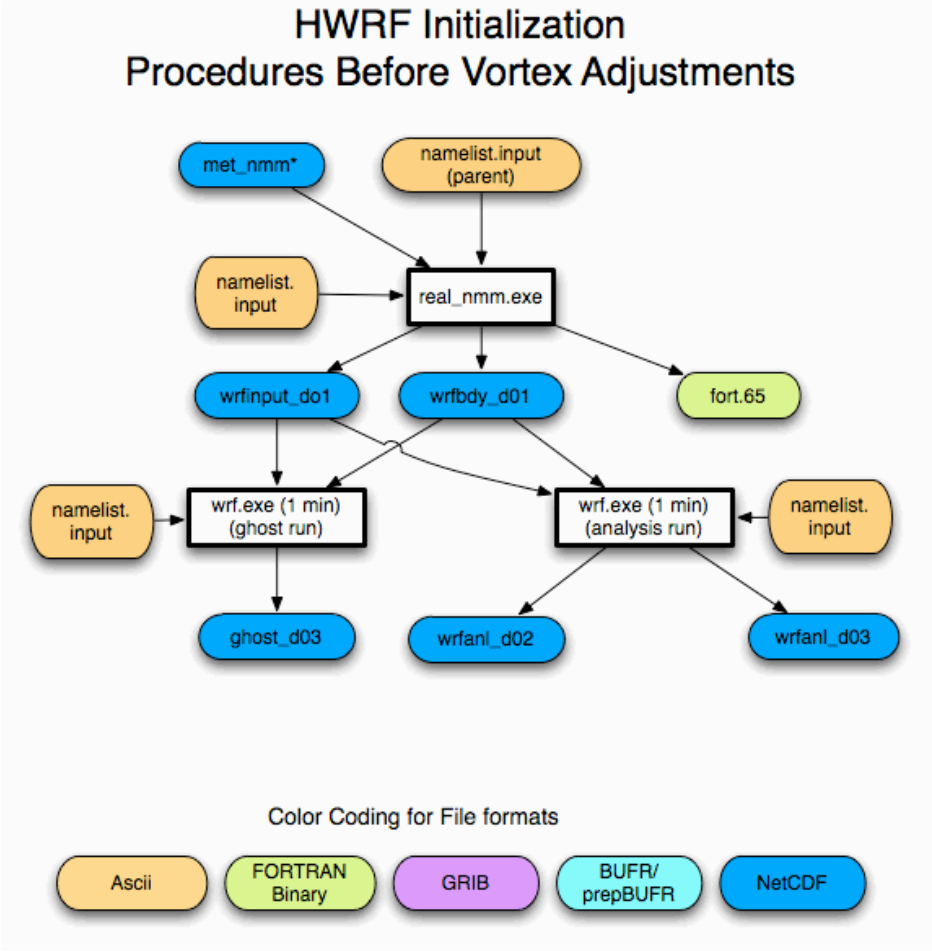

Figure 4.3. HWRF initialization procedures before vortex adjustments. Note that only the output files that are used in subsequent runs are shown.

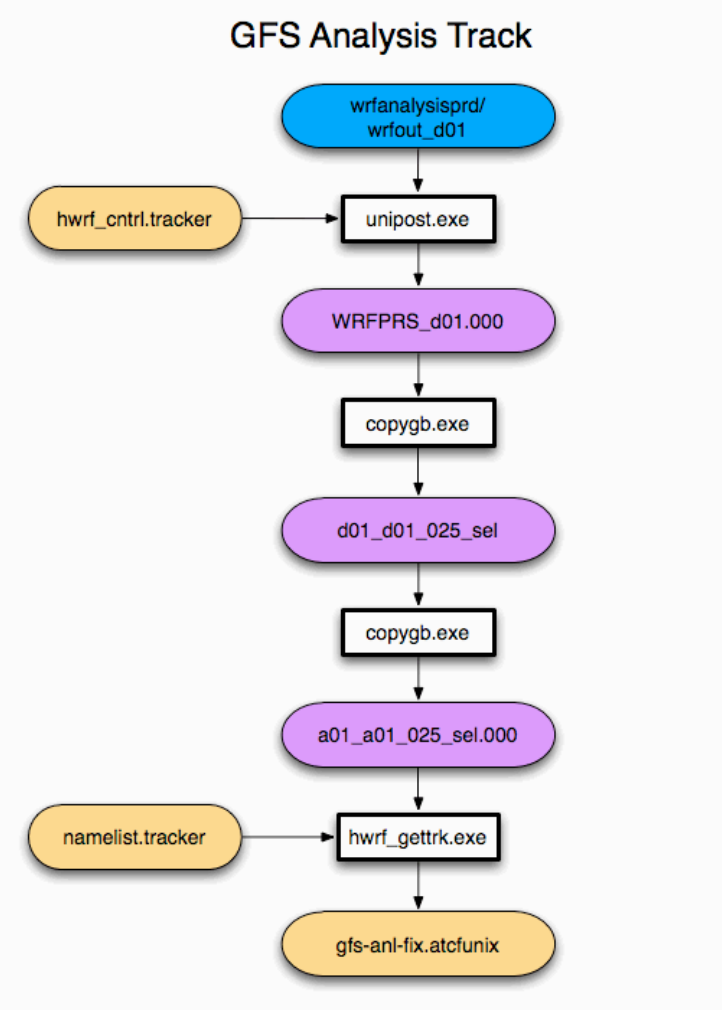

Figure 4.4. Diagram of the procedure to generate information about the storm location in the GFS input data. The color coding is described in Figure 4.3.

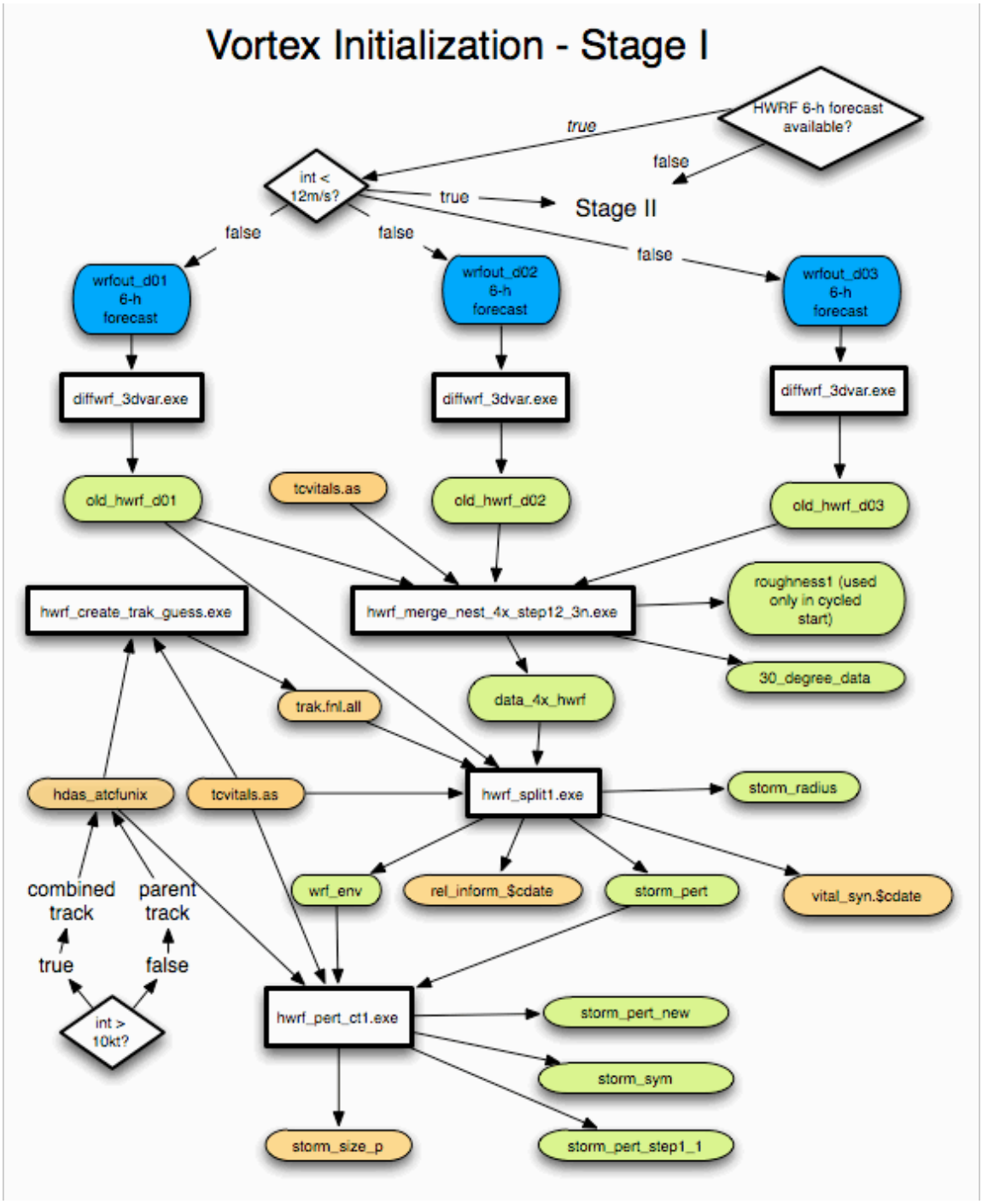

Figure 4.5. Diagram of HWRF vortex initialization Stage 1 procedures. The color coding is described in Figure 4.3.

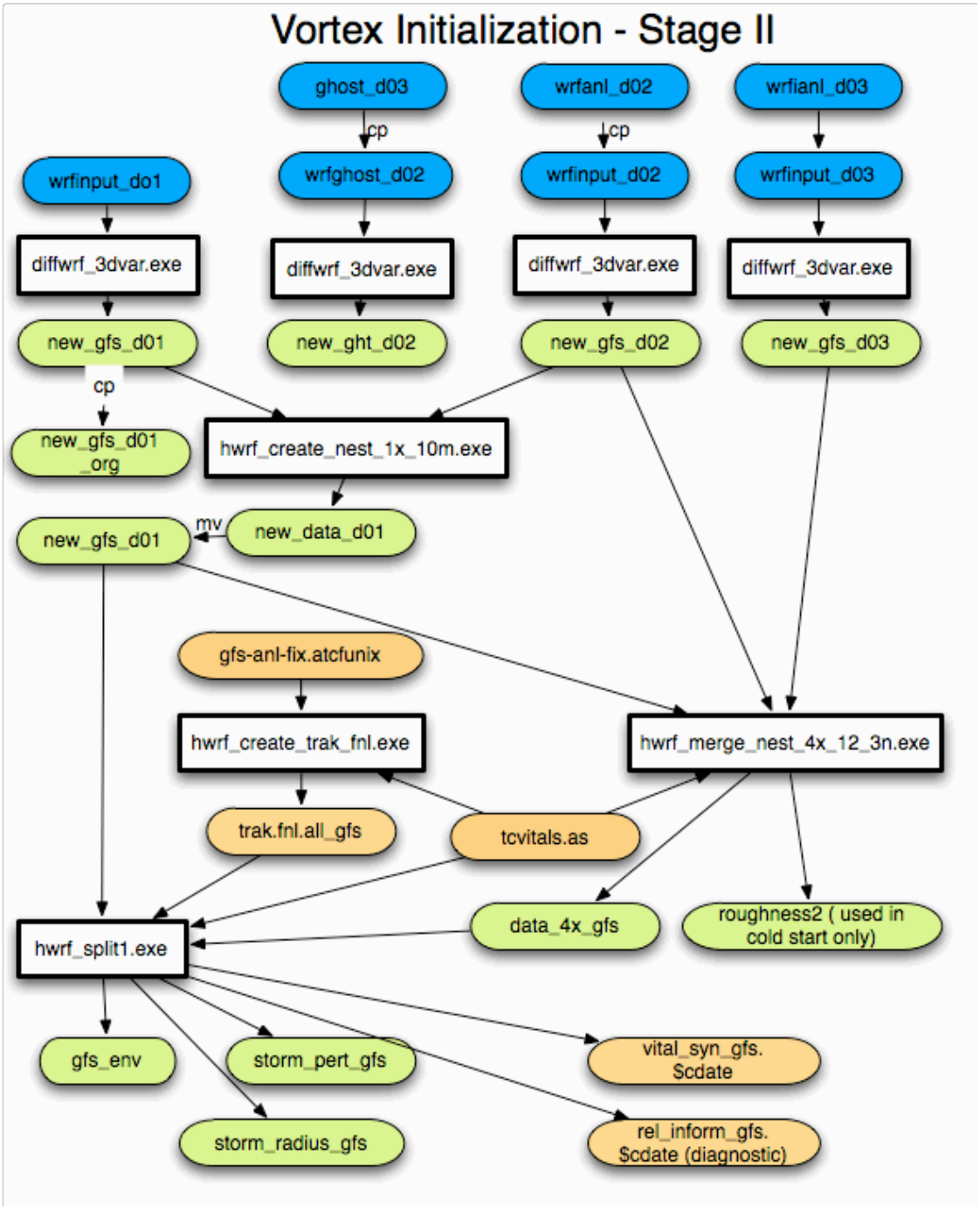

Figure 4.6. Diagram of HWRF vortex initialization Stage 2 procedures. The color coding is described in Fig. 4.3.

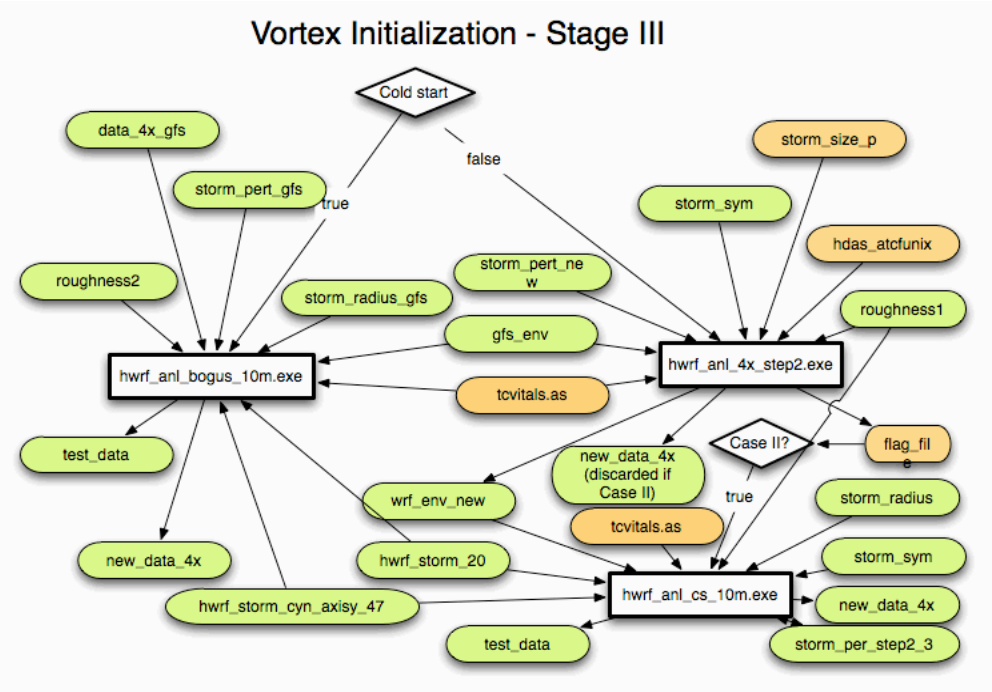

Figure 4.7. Diagram of HWRF vortex initialization Stage 3 procedures. The color coding is described in Figure 4.3.

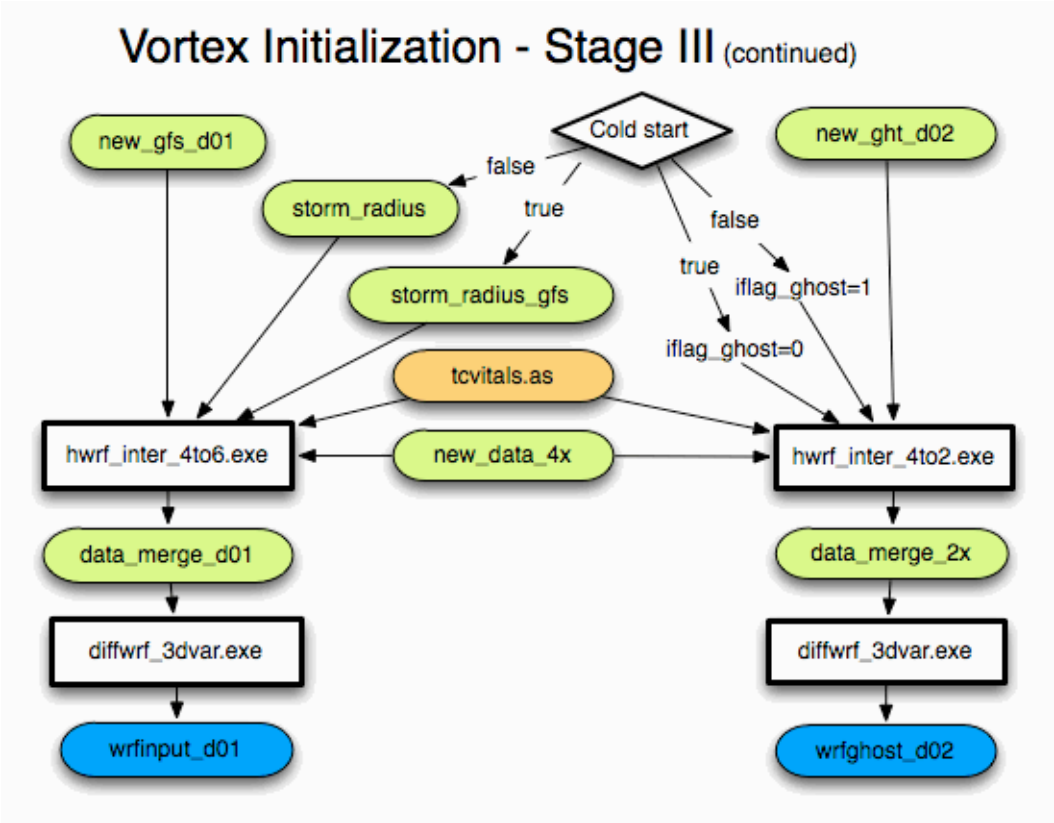

Figure 4.8. Diagram of HWRF vortex initialization Stage 3 final procedures. The color coding is described in Figure 4.3.

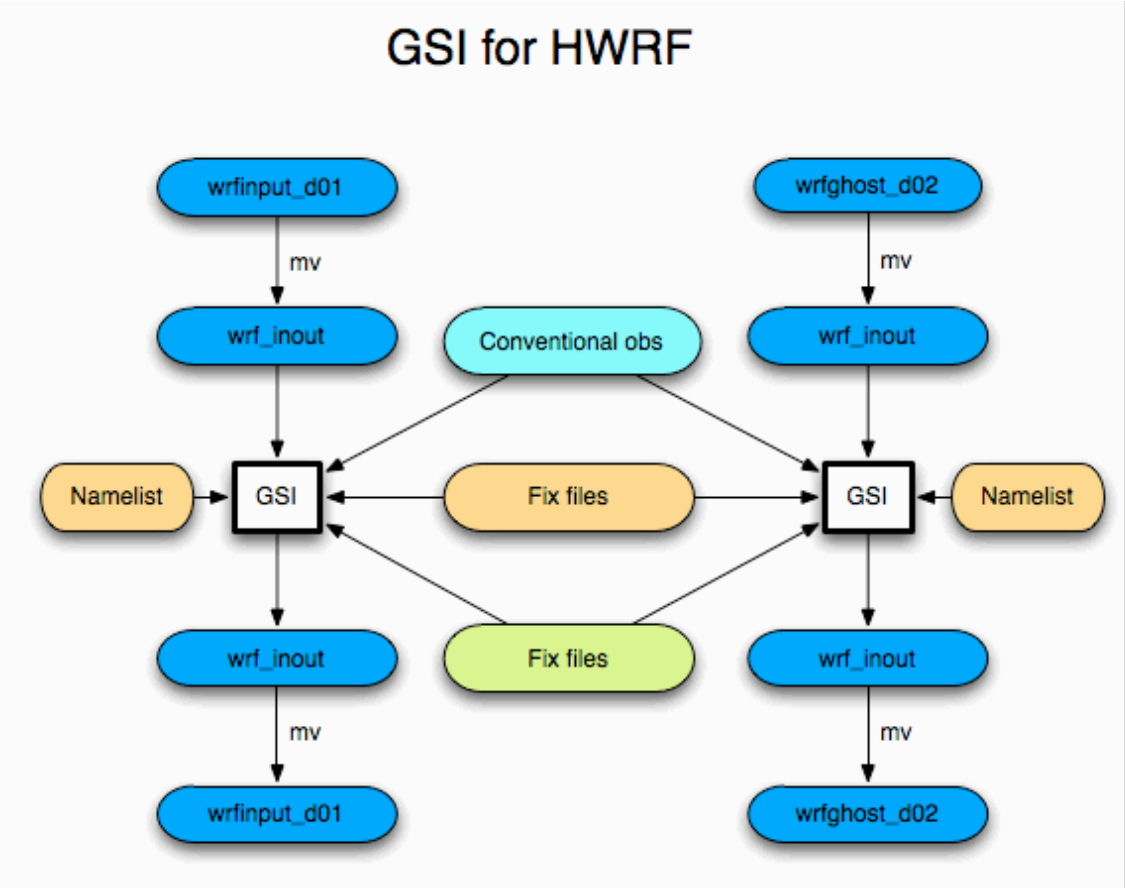

Figure 4.9. Diagram of HWRF vortex initialization GSI procedures. The color coding is described in Figure 4.3.

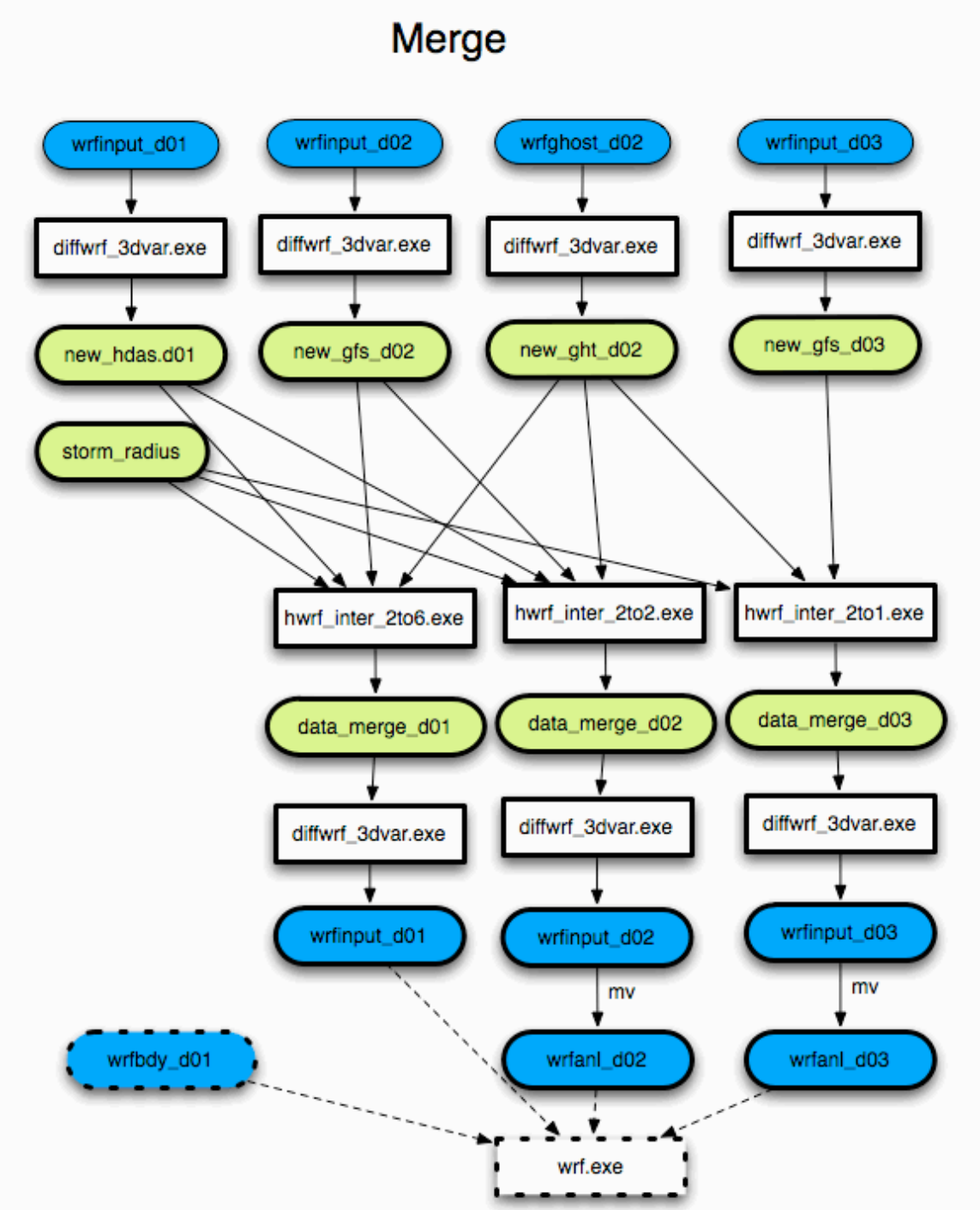

Figure 4.10. Diagram of HWRF vortex initialization merge procedures. The color coding is described in Figure 4.3.

4.4 HWRF Vortex Initialization Executables

4.4.1 *copygb.exe*

See Section 7.4.2.

4.4.2 *diffwrf_3dvar.exe*

a) **FUNCTION:**
Converts netCDF input to unformatted file (when first argument is "*storm_relocate*").

INPUT:
netCDF format input files (for example *wrfinput_d01*) or previous cycle 6-hr forecast.

OUTPUT:
unformatted data file.

USAGE:
diffwrf 3dvar.exe storm relocate input file flnm3 output file
The command above writes the WRF input file *input file* into an unformatted file, *output file,* which will be used in the vortex relocation procedures.

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 file.

USAGE:
diffwrf 3dvar.exe 3dvar update input file output file
The command above updates *input file* with unformatted file *output file,* which contains new vortex fields.

4.4.3 *gettrk.exe*

See Section 8.6.1.

4.4.4 *gsi.exe*

FUNCTION:
Performs the GSI 3D-VAR data assimilation analysis.

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

fix files, from *${FIX_ROOT},* which is specified in *hwrf-utilities/wrapper_scripts/global_vars.ksh*

OUTPUT:
wrf_inout: analysis results if GSI completes successfully. The format is the same as the background file.

USAGE:
On IBM/AIX machines: *mpirun.lsf ./gsi.exe < gsiparm.anl*
On Intel/Linux: *mpiexec –np 12 ./gsi.exe < gsiparm.anl*

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 3X 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-hr forecast
gfs_env (fort.26) - GFS environmental flow
storm_pert_new (fort.71) - adjusted storm perturbation from stage 1
roughness1 (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 3X domain.

USAGE:
echo $gesfhr | hwrf_anl_4x_step2.exe

4.4.6 *hwrf_anl_bogus_10m.exe*

FUNCTION:
Creates a bogus storm and adds it to the environmental flow

INPUT:
$gesfhr(=6)
tcvitals.as (fort.11) – observed storm center
gfs_env (fort.26) - GFS environmental flow
data_4x_gfs (fort.36) - merged GFS inner/outer domain data
storm_pert_gfs (fort.61) - separated GFS 3D vortex field
roughness2 (fort.46) - roughness info for boundary layer calculation
storm_radius_gfs (fort.85)
hwrf_storm_cyn_axisy_47 (fort.71-75 and fort.78) input static vortex data
hwrf_storm_20 (fort.76-77)

OUTPUT:
new_data_4x: combined environment flow and bogus field on the 3X domain

USAGE:
echo $gesfhr | hwrf_anl_bogus_10m.exe

4.4.7 *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 3X 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

4.4.8 *hwrf_create_nest_1x_10m.exe*

FUNCTION:
Rebalances inner nest data.

INPUT:
$gesfhr(=6) is used to generate the input and output file unit numbers.
new_gfs_d02 (*fort.46*)
new_gfs_d01 (*fort.26*)

OUTPUT:
new_data_d01(fort.57)- outer domain data interpolated to inner domain.
new_data_d01, which is renamed to *new_gfs_d01*

USAGE:
echo $gesfhr | hwrf_create_nest_1x_10m.exe

4.4.9 *hwrf_create_trak_guess.exe*

FUNCTION:
Guesses storm center from previous 6-hr 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-hr forecast.

OUTPUT:
trak.fnl.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_data_remv.exe*

FUNCTION:
Removes 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(=1200 km): 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

4.4.11 *hwrf_inter_2to1.exe*

FUNCTION:
Interpolates from ghost d03 domain to inner nest domain.

INPUT:
$gesfhr(=6)
new_ght_d02 (fort.26) - data on ghost d03 domain.
new_gfs_d03 (fort.36) – data on inner nest domain.

OUTPUT:
data_merge_d03 (fort.56) - interpolated data on inner domain.

USAGE:
echo ${gesfhr} | hwrf_inter_2to1.exe

4.4.12 *hwrf_inter_2to2.exe*

FUNCTION:
Interpolates from ghost d03 domain to middle nest domain.

INPUT:
$gesfhr(=6)
new_ght_d02 (fort.26) - data on ghost d03 domain.
new_gfs_d02 (fort.36) – data on middle nest domain.
new_hdas_d01 (fort.46) – data on outer domain.

OUTPUT:
data_merge_d02 (fort.56) - interpolated data on middle nest domain.

USAGE:
echo ${gesfhr} | hwrf_inter_2to2.exe

4.4.13 *hwrf_inter_2to6.exe*

FUNCTION:
Interpolates data from ghost domain to outer domain.

INPUT:
$gesfhr (=6)
new_gfs_d02 (*fort.26*) – data on HWRF middle nest grid
new_ght_d02 (*fort.36*) - data on ghost d03 grid
new_hdas_d01 (*fort.46*) – data on outer domain grid
storm_radius (*fort.85*) - storm radius obtained from *wrf_split1* 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

4.4.14 hwrf_inter_4to2.exe

FUNCTION:
Interpolates from 3X domain onto ghost d02 domain.

INPUT:
$gesfhr (=6)
tcvitals.as (*fort.11*) - storm center obs
new_data_4x (*fort.26*) - adjusted storm on 3X domain
new_ght_d02 (*fort.36*) - ghost middle domain data

OUTPUT:
data_merge_2x (*fort.56*) - merged data on ghost d02 domain.

USAGE:
echo $gesfhr | hwrf_inter_4to2.exe

4.4.15 *hwrf_inter_4to6.exe*

FUNCTION:
Interpolates from 3X 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

4.4.16 *hwrf_merge_nest_4x_step12_3n.exe*

FUNCTION:
Merges inner and outer domains onto a 3X 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
old_hwrf_d01 or new_gfs_d01 (fort.26) - outer domain data
old_hwrf_d02 or new_gfs_d02 (fort.36) - middle domain data
old_hwrf_d03 or new_gfs_d03 (fort.46) - inner domain data

OUTPUT:
data_4x_hwrf (fort.56) - merged data from inner and outer domains
roughness1 or roughness2 (fort.66) - sea-mask (1=sea, 0=land) and ZNT (roughness length) merged onto the 3X domain.

30_degree_data (fort.61): partially merged data from inner and outer domains. Not used later.

USAGE:
echo ${gesfhr} ${st_int} ${ibgs} | hwrf_merge_nest_4x_10m2.exe

4.4.17 *hwrf_pert_ct1.exe*

FUNCTION:
Adjusts storm vortex (*storm_pert*).

INPUT:
$gesfhr(=6)
hdas atcfunix (fort.12) - storm track from previous 6-hr forecast
tcvitals.as (fort.11) - storm center obs
wrf env (fort.26) - environmental flow from previous6-hr forecast (*wrf_split1*'s output)
storm pert (fort.71) - separated 3D vortex field (*wrf split1*'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

USAGE:
echo $gesfhr | hwrf_pert_ct1.exe

4.4.18 *hwrf_split1.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 3X 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.$cdate (*fort.52*) - diagnostics file (obs-previous 6-hr forecast)
vital_syn.$cdate (*fort.55*) – information for generating bogus if storm not found in previous 6-hr forecast

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

4.4.19 *hwrf_wrfout_newtime.exe*

FUNCTION:
Changes the time stamp of WRF analysis run d01 output from 1.5 minute (t=1.5 minute) to initial time (t=0), so that it can be used in the track analysis script (*track_analysis.ksh*).

INPUT:
WRF analysis run output: *wrfout_d01_yyyy-mm-dd_hh:01:30*

OUTPUT:WRF analysis run output with its time stamp changed:
wrfout_d01_yyyy-mm-dd_hh:00:00

USAGE:
hwrf_wrfout_newtime.exe wrfout_d01_yyyy-mm-dd_hh:00:00 yyyymmddhh

4.4.20 *ssrc.exe*

FUNCTION:
Reverses 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

4.4.21 *unipost.exe*

See Section 7.4.1.

Chapter 5: Ocean Initialization of POM-TC

5.1 Introduction

This chapter 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 v3.4a Scientific Documentation.

5.2 Run Ocean Initialization Using the Wrapper Script

The wrapper script involved with running the ocean initialization, *pom_init_wrapper*, can be found in the directory

${SCRATCH}/HWRF/hwrf-utilities/wrapper_scripts/.

Before running *pom_init_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

DOMAIN_DATA
POMTC_ROOT
START_TIME
BASIN
SID
TCVITALS
LOOP_CURRENT_DIR
GFS_SPECTRAL_DIR
use_extended_eastatl
HWRF_SCRIPTS
OCEAN_FIXED_DIR

After confirming the environment variables listed above are defined correctly, the user can run the wrapper script by typing the command:

./pom_init_wrapper.

The two relevant scripts are "*pom_init.ksh,*" which runs the ocean initialization, and "*gfdl_pre_ocean_sortvit.sh*," which is called from within "*pom_init.ksh*." The wrapper script will call the low-level scripts *pom_init.ksh*.

Script "*pom_init.ksh*" is composed of the following seven functions.

function main
function get_tracks
function get_region
function get_sst
function sharpen
function phase_3
function phase_4

Scripts *pom-tc-united-grads.sh and pom-tc-eastatl-grads.s,h* to plot ocean output in the Atlantic basin using GRADS, can be found in the directory

${SCRATCH}/HWRF/pomtc/ocean_plot.

5.3 Functions in Script "*pom_init.ksh*"

5.3.1 *main*

1. Initialize the function library.
2. Check to see if all the variables are set.
3. Alias the executables/scripts.
4. Check to see if all the executables/scripts exist.
5. Set the stack size.
6. Create a working directory (and cd into it).
7. Get the existing storm track information using function *get_tracks*.
8. Find the ocean region using function *get_region* and then set it accordingly.
9. Get the GFS sea surface temperature using function *get_sst*.
10. Run the feature-based sharpening program using function sharpen.
11. Run POM-TC phase 1 (a.k.a. phase 3) using function *phase_3*.
12. Run POM-TC phase 2 (a.k.a. phase 4) using function *phase_4*.

5.3.2 *get_tracks*

1. Get the entire existing storm track record from the *syndat_tcvitals* file using script *gfdl_pre_ocean_sortvit.sh* and store it in file *track.allhours*.
2. Add a blank record at the end of the storm track in file *track.allhours*.
3. Remove all cycles after the current cycle from the storm track record and store it in file *track.shortened*.
4. Use *track.shortened* as track file if it is not empty; otherwise, use *track.allhours*.

5. Extract various storm statistics from the last record in the track file to generate a 72-hour projected track that assumes storm direction and speed remain constant; save this projected track in file *shortstats*.

5.3.3 *get_region*

1. Run the find region code, which selects the ocean region based on the projected track points in the *shortstats* file; this region is east_atlantic or west_united.
2. Store the ocean region from the find region code in file *ocean_region_info.txt*.
3. If the ocean basin is the East Pacific, reset the ocean region to east_pacific.
4. Set region variable to eastpac, eastatl, or united; run uncoupled if a storm is not in one of these regions.
5. Store the region variable in file *pom_region.txt*.

5.3.4 *get_sst*

1. Create the directory for the GFS SST, mask, and lon/lat files.
2. Create symbolic links for the GFS spectral input files.
3. Run the *getsst* code.
4. Rename the GFS SST, mask, and lon/lat files for POM-TC phase 3.

5.3.5 *sharpen*

1. Prepare symbolic links for most of the input files for the sharpening program.
2. Continue with function *sharpen* only if the region variable is set as united.
3. Create the directory for the sharpening program output files.
4. Continue with function *sharpen* only if the Loop Current and ring files exist.
5. Use backup GDEM monthly climatological temperature and salinity files if they exist but the Loop Current and ring files do not exist; warn the user accordingly.
6. Exit the ocean initialization with an error if neither the Loop Current and ring files nor the backup climatological temperature and salinity files exist.
7. Assuming the Loop Current and ring files exist, use the simulation start date to select the second of two temperature and salinity climatology months to use for time interpolation to the simulation start date.
8. Choose the climatological input based on *input_sharp* (hardwired to GDEM).
9. Create symbolic links for all input files for the sharpening program.
10. Run the sharpening code.
11. Rename the sharpened climatology file as *gfdl_initdata* for POM-TC phase 3.

5.3.6 *phase_3*

1. Create the directory for the POM-TC phase 3 output files.
2. Prepare symbolic links for some of the input files for POM-TC phase 3.

3. Modify the phase 3 parameter file by including the simulation start date.
4. Prepare symbolic links for the sharpened (or unsharpened) temperature and salinity input file and the topography and land/sea mask file based on whether the region variable is united, eastatl, or eastpac. If the region variable is eastatl, choose whether or not to use the extended east Atlantic domain based on whether or not the value of variable *use_extended_eastatl* is set to true.
5. Create symbolic links for all input files for POM-TC phase 3. These links include extra input files for defining the domain center and the land/sea mask if the region variable is eastpac.
6. Run the POM-TC code for phase 3.
7. Rename the phase 3 restart file as *RST.phase3* for POM-TC phase 4.

5.3.7 *phase_4*

1. Create the directory for the POM-TC phase 4 output files.
2. Prepare symbolic links for some of the input files for POM-TC phase 4.
3. If the track file has less than three lines in it, skip POM-TC phase 4 and use *RST.phase3* for initializing the coupled HWRF simulation.
4. Back up three days to end phase 4 at the coupled HWRF start date.
5. Modify the phase 4 parameter file by including the simulation start date, the track file, and *RST.phase3*.
6. Prepare symbolic links for the sharpened (or unsharpened) temperature and salinity input file and the topography and land/sea mask file based on whether the region variable is united, eastatl, or eastpac. If the region variable is eastatl, choose whether or not to use the extended east Atlantic domain based on whether or not the value of variable *use_extended_eastatl* is set to true.
7. Create symbolic links for all input files for POM-TC phase 4, including the track file.
8. Run the POM-TC code for phase 4.
9. Rename the phase 4 restart file as *RST.final* for the coupled HWRF simulation.

5.4 Executables

5.4.1 *gfdl_find_region.exe*

FUNCTION:
Select the ocean region based on the projected track points in the *shortstats* file; this region is east_atlantic or west_united.

INPUT:
shortstats

OUTPUT:
fort.61 (ocean_region_info.txt)

USAGE:

${SCRATCH}/HWRF/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:
for11 (gfs.${start_date}.t${cyc}z.sfcanl)
fort.11 (gfs.${start_date}.t${cyc}z.sfcanl)
fort.12 (gfs.${start_date}.t${cyc}z.sanl)

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

USAGE:
${SCRATCH}/HWRF/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 (mmdd.dat)
fort.31 (hwrf_gfdl_loop_current_rmy5.dat.${yyyymmdd})
fort.32 (hwrf_gfdl_loop_current_wc_ring_rmy5.dat.${yyyymmdd})

OUTPUT:

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

USAGE:
${SCRATCH}/HWRF/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 (*empty 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:
RST.phase3.united if phase 1*; RST.final* if phase 2
phase3.out if phase 1*; phase4.out* if phase 2

USAGE:
Phase 1: *${SCRATCH}/HWRF/pomtc/ocean_exec/gfdl_ocean_united.exe > phase3.out*
Phase 2: *${SCRATCH}/HWRF/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:
fort.10 (parameters.inp)
*fort.15 (*empty 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.eastatl.${mm})

fort.66 (gfdl_Hdeepgsu.eastatl)
*fort.14 (*not used if phase 1; *RST.phase3.eastatl* if phase 2*)*
OUTPUT:
RST.phase3.eastatl if phase 1*; RST.final* if phase 2
phase3.out if phase 1*; phase4.out* if phase 2

USAGE:
Phase 1: *${SCRATCH}/HWRF/pomtc/ocean_exec/gfdl_ocean_eastatl.exe > phase3.out*
Phase 2: *${SCRATCH}/HWRF/pomtc/ocean_exec/gfdl_ocean_eastatl.exe > phase4.out*

5.4.6 *gfdl_ocean_ext_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 extended East Atlantic region.

INPUT:
fort.10 (parameters.inp)
*fort.15 (*empty 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_extn)
*fort.14 (*not used if phase 1; *RST.phase3.eastatl* if phase 2*)*

OUTPUT:
RST.phase3.eastatl if phase 1*; RST.final* if phase 2
phase3.out if phase 1*; phase4.out* if phase 2

USAGE:
Phase 1: *${SCRATCH}/HWRF/pomtc/ocean_exec/gfdl_ocean_ext_eastatl.exe > phase3.out*
Phase 2: *${SCRATCH}/HWRF/pomtc/ocean_exec/gfdl_ocean_ext_eastatl.exe > phase4.out*

5.4.7 *gfdl_ocean_eastpac.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 Pacific region.

INPUT:
*domain.center (*used if phase 1; not used if phase 2*)*
*gfdl_pctwat (*used if phase 1; not used if phase 2*)*
fort.10 (parameters.inp)
*fort.15 (*empty if phase 1; *track* if phase 2*)*
fort.21 (sst.gfs.dat)
fort.22 (mask.gfs.dat)
fort.23 (lonlat.gfs)
fort.45 (gfdl_raw_temp_salin.eastpac.${mm} if phase 1; not used if phase 2*)*
*fort.13 (*output if phase 1; *temp_salin_levitus.eastpac* if phase 2*)*
*fort.66 (*output if phase 1; *eastpac_ocean_model_info* if phase 2*)*
*fort.14 (*not used if phase 1; *RST.phase3.eastpac* if phase 2*)*

OUTPUT:
RST.phase3.eastpac if phase 1*; RST.final* if phase 2
phase3.out if phase 1*; phase4.out* if phase 2
fort.13 (temp_salin_levitus.eastpac) if phase 1 only
fort.66 (eastpac_ocean_model_info) if phase 1 only

USAGE:
Phase 1: *${SCRATCH}/HWRF/pomtc/ocean_exec/gfdl_ocean_eastpac.exe > phase3.out*
Phase 2: *${SCRATCH}/HWRF/pomtc/ocean_exec/gfdl_ocean_eastpac.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 Wrapper Script *hwrf_wrapper*

This section describes how to use the wrapper script *hwrf-utilities/wrapper_scripts/ hwrf_wrapper* to run the coupled HWRF forecast on two types of platforms: the IBM/AIX and Linux machines. The user is responsible for understanding how to run MPMD jobs on the platform where the HWRF system will be run, if that system is not covered in this document.

Before running *hwrf_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
WRF_ROOT
HWRF_UTILITIES_ROOT
START_TIME
ATMOS_DOMAINS
FCST_LENGTH
FCST_INTERVAL
DOMAIN_DATA

Note that in the wrapper script *hwrf_wrapper*, the following two variables are defined.

WRF_MODE = main
WRF_CORES =202

Note that *WRF_CORES* includes one processor for the coupler and one for POMTC. The user can define WRF_CORES to a different number. It has been shown that WRF_ CORES can be defined as the following numbers (including the two processors for the coupler and POMTC):

6, 10, 18, 26, 34, 38, 50, 66, 92, 102, 122, 152, 182, and 202.

The following numbers do not work:

73, 39, and 43.

The user can either use a batch system to submit the HWRF job to the remote computation nodes, or, on Linux machines that use Oracle Grid Engine, interactively connect to these computation nodes and run the job. Both methods are described in Section 1.6.

6.3 Overview of the Script

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.
2. Create and enter the work directory.
3. Link the input files required by WRF, including fix files, initial and boundary condition files and geographical data files.
4. Run *hwrf_swcorner_dynamic.exe* to calculate the location of the middle nest and generate the WRF namelist, *namelist.input*.
5. Link the input files required by POM-TC, including fix files, initial and boundary conditions files, bathymetry/topography data files etc.
6. Generate a namelist for POM-TC.
7. Generate a namelist for the coupler.
8. Submit the WRF, POM-TC coupled run.

- On IBM with LSF:

  Use the command *mpirun.lsf*

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 202 processors, one for the coupler (*hwrf_wm3c.exe*), one for the United domain ocean model (*hwrf_ocean_united.exe*) and 200 for *wrf.exe:*

hwrf_wm3c.exe
hwrf_ocean_united.exe
wrf.exe
wrf.exe
(total of 200 wrf.exe)

wrf.exe
wrf.exe

- On Linux with Oracle Grid Engine, previously known as Sun Grid Engine (SGE):
  Use the command *mpiexec*
  For example, the following command will run the coupled model using 202 processors, one for the coupler (*hwrf_wm3c.exe*), one for the United domain ocean model (*hwrf_ocean_united.exe*) and 200 for *wrf.exe*
  /usr/local/esrl/bin/mpiexec -np 1 ./hwrf_wm3c.exe : -np 1 ./hwrf_ocean_united.exe : -np 200 ./wrf.exe

Note that in the examples listed above, for the POM-TC United domain, the ocean model executable *hwrf_ocean_united.exe* is used.

HWRF has the capability of running uncoupled (atmosphere standalone) runs if the changes below are made to the script. Note that this is not an operational configuration.

On IBM with LSF:
Use the command *mpirun.lsf*
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 Oracle Grid Engine, previously known as Sun Grid Engine (SGE):
Use the command *mpiexec*
For example, the following command will run the uncoupled model using 200 processors for *wrf.exe*
mpiexec -np 200 ./wrf.exe

6.4 Output Files in the Directory

Output files in directory *${DOMAIN_DATA}/wrfprd*

A successful run of the wrapper script *hwrf_wrapper* and the low-level script *wrf.ksh* will produce output files with the following naming convention.

Primary output files containing most variables, output every three hours.

wrfout_d01_yyyy-mm-dd_hh:mm:ss
wrfout_d02_yyyy-mm-dd_hh:mm:ss
wrfout_d03_yyyy-mm-dd_hh:mm:ss

Auxiliary output files containing accumulated precipitation and 10-m winds, hourly output

auxhist1_ d01_yyyy-mm-dd_hh:mm:ss
auxhist1_ d02_yyyy-mm-dd_hh:mm:ss
auxhist1_ d02_yyyy-mm-dd_hh:mm:ss

Text file with time series of storm properties.
hifreq_d03.htcf

File *hifreq_d03.htcf* has nine columns containing the following items.

1. forecast lead time (s)
2. minimum sea level pressure in the inner nest (hPa)
3. latitude of gridpoint with minimum sea level pressure
4. longitude of gridpoint with minimum sea level pressure
5. maximum wind in the inner nest at the lowest model level (kt)
6. latitude of gridpoint with the maximum wind
7. longitude of gridpoint with the maximum wind
8. latitude of the location of the center of the inner nest
9. longitude of the location of the center of the inner nest

The ocean model will produce diagnostic output files with the following naming convention.

1. *GRADS.yymmddhh,* (GrADS format, including temperature, salinity, density, U, V, average U, average V and elevation) (for united and east Atlantic basins)
2. *EL.yymmddhh* (binary, elevation) (for united, east Atlantic and east Pacific basins
3. *MLD.yymmddhh* (binary, mixed layer depth) (for united and east Atlantic basins)
4. *OHC.yymmddhh* (binary, ocean heat content) (for united and east Atlantic basins)
5. *T.yymmddh* (binary, temperature) (for united, east Atlantic and east Pacific basins)
6. *TXY.yymmddhh* (binary, momentum flux) (for united, east Atlantic and east Pacific basins)
7. *U.yymmddh* (binary east-west direction current) (for united, east Atlantic and east Pacific basins)
8. *V.yymmddhh* (binary north-south direction current) (for united, east Atlantic and east Pacific basins)
9. *WTSW.yymmddhh* (binary, heat flux and shortwave radiation) (for united and east Atlantic basins)

For example, the first POM-TC output file for a run started at 1200 UTC, 23 August 2011 would be: *GRADS.11082302 etc.*

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 Running HWRF with Alternate Namelist Options

By following the directions above, the namelist for the WRF model (namelist.input) will be constructed from a template provided in *hwrf-utilities/parm*. A sample namelist can be found in Section 6.8. This template should only be altered by advanced users because many of the options available in the WRF model are not supported for the HWRF configuration.

The WRF namelist is described in detail at http://www.dtcenter.org/wrf-nmm/users/docs/user_guide/V3/users_guide_nmm_chap1-7.pdf.

The HWRF physics suite can be altered in the ways described in the table below. These configurations are only preliminarily tested and only limited support is provided for their use.

PARAMETERIZATION	OPERATIONAL	POSSIBLE ALTERNATES (ONLY PRELIMINARILY TESTED)
Cumulus	HWRF SAS (84)	Tidtke (6), KF (1- trigger option 1), New SAS (14)
Microphysics	Tropical Ferrier (85)	None
Surface Layer	GFDL (88)	None
Planetary Boundary Layer	GFS (3)	None
Land Surface Model	GFDL (88)	Noah (2)
Long Wave Radiation	GFDL (98)	RRTMG (4) (must link in table)
Short Wave Radiation	GFDL (98)	RRTMG (4) (must link in table), Dudhia (1)

The HWRF operational configuration uses option vortex_tracker=2,2,4 for the internal nest tracking. Note the following recommendations about the choice of vortex_tracker.

1. Vortex_tracker=1 only works when max_dom=2.
2. Vortex_tracker=2 only works for d01 or when the domain has an active nest.
3. Vortex_tracker=4 should not be used for d01.

Users have the option of getting a high frequency (each time step) of ascii output of several hurricane related values (forecast lead time, minimum MSLP, location of the minimum MSLP, max wind speed, location of the maximum wind speed, location of the

domain center) by setting the "high_freq" option in the namelist to "true". Setting it to "false" will turn off this output. The "high_freq" option is in the time_control section of the namelist.

6.7 Executables

6.7.1 *wrf.exe*

FUNCTION:
Atmospheric component of HWRF

INPUT:
geogrid static files: *geo_nmm.d01.nc, geo_nmm_nest.l01.nc, and geo_nmm_nest.l02.nc*
wrfbdy file: wrfbdy_d01
wrfinput file: wrfinput_d01
wrfanl files:wrfanl_d02_${YYYY}-${MM}-${DD}_${HH}:00:00 and *wrfanl_d03_${YYYY}-${MM}-${DD}_${HH}:00:00*
fort.65 and gravity wave drag file *gwd_surface*
WRF static files
namelist.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
wrfout_d02_yyyy-mm-dd_hh:mm:ss
wrfout_d03_yyyy-mm-dd_hh:mm:ss
auxhist1_ d01_yyyy-mm-dd_hh:mm:ss
auxhist1_ d02_yyyy-mm-dd_hh:mm:ss
auxhist1_ d02_yyyy-mm-dd_hh:mm:ss
hifreq_d03.htcf

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

6.7.2 *hwrf_wm3c.exe*

FUNCTION:
Coupler that links the atmospheric component *wrf.exe* and oceanic component *hwrf_ocean_united.exe*, *hwrf_ocean_eastatl.exe* or *hwrf_ocean_eastpac.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*, *hwrf_ocean_eastatl.exe* or *hwrf_ocean_eastpac.exe* (see Section 6.2).

6.7.3 *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: *${VARIABLE}.yymmddhh,* where ${VARIABLE} includes GRADS, EL, OHC, MLD, T, U, V, WTSW and TXY. (see Section 6.4)
For example, the first POM-TC output file for a run started at 1200 UTC, 23 August 2011 would be *GRADS.11082312*

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).

6.7.4 *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_Hdeepgsu.eastatl
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: *${VARIABLE}.yymmddhh,* where ${VARIABLE} includes GRADS, EL, OHC, MLD, T, U, V, WTSW and TXY. (see Section 6.4)

For example, the first POM-TC output file for a run started at 1200 UTC, 23 August 2011 would be *GRADS.11082312*.

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).

6.7.5 *hwrf_ocean_eastatl_ext.exe*

FUNCTION:
Oceanic model for HWRF, for the East Atlantic extended domain. The East Atlantic extended domain is used when the storm is in East Atlantic region and

the environment variable ${use_extended_eastatl} = T. This is not an operational configuration.

INPUT:
gfdl_ocean_topo_and_mask.eastatl_ext
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: *${VARIABLE}.yymmddhh,* where ${VARIABLE} includes GRADS, EL, OHC, MLD, T, U, V, WTSW and TXY. (see Section 6.4)

For example, the first POM-TC output file for a run started at 1200 UTC, 23 August 2011 would be *GRADS.11082312*

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

6.7.6 *hwrf_ocean_eastpac.exe*

FUNCTION:
Oceanic model for HWRF, for the east Pacific domain.

INPUT:
temp_salin_levitus.eastpac
eastpac_ocean_model_info
RST.final
sst.gfs.dat
mask.gfs.dat

lonlat.gfs
track
Note the ocean's initial state of temperature and salinity for east Pacific basin (*temp_salin_levitus.eastpac*) comes from fixed data based on climatology.

OUTPUT:
The ocean model will produce output files with the following naming convention.
${VARIABLE}.yymmddhh, where ${VARIABLE} includes EL, T, U, V, and TXY. (see Section 6.4)

For example, the first POM-TC output file for a run started at 1200 UTC, 23 August 2011 would be *EL.11082312*.

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

6.7.7 *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 middle nest domain lower-left corner location is at (103,187) on the parent domain grid.

```
istart=00103
jstart=00187
```

USAGE:
hwrf-utilities/exec/hwrf_swcorner_dynamic.exe

6.8 Sample HWRF namelist

The HWRF namelist used for the release case, Hurricane Irene (2011), is listed below.

```
&time_control
 start_year                 = 2011, 2011, 2011,
 start_month                = 08, 08, 08,
 start_day                  = 23, 23, 23,
 start_hour                 = 12, 12, 12,
 start_minute               = 00,      00,     00,
 start_second               = 00,      00,     00,
 end_year                   = 2011, 2011, 2011,
 end_month                  = 08, 08, 08,
 end_day                    = 28, 28, 28,
 end_hour                   = 18, 18, 18,
 end_minute                 = 00, 00, 00,
 end_second                 = 00,      00,     00,
 interval_seconds           = 21600,
 history_interval           = 180,         180,        180,
 auxhist1_interval          = 60,          60,        60
 frames_per_outfile         = 1,1,1
 frames_per_auxhist1        = 1,1,1
 analysis                   = F,  T,T,
 restart                    = .false.,
 restart_interval           = 36000,
 reset_simulation_start     = F,
 io_form_input              = 2
 io_form_history            = 2
 io_form_restart            = 2
 io_form_boundary           = 2
 io_form_auxinput1          = 2
 io_form_auxhist1           =  2
 auxinput1_inname           = "met_nmm.d<domain>.<date>"
 debug_level                = 1
 override_restart_timers    = T
 /

 &fdda
 /
&domains
 time_step                  = 45,
 time_step_fract_num        = 0,
 time_step_fract_den        = 1,
 max_dom                    = 3,
 s_we                       = 1,         1,        1,
 e_we                       = 216,   88,        154,
 s_sn                       = 1,         1,        1,
 e_sn                       = 432,   170,       272,
 s_vert                     = 1,         1,        1,
```

```
 e_vert                    = 43,        43,       43,
 dx                        = 0.18,      0.06,     0.02,
 dy                        = 0.18,      0.06,     0.02,
 grid_id                   = 1,         2,        3,
  tile_sz_x                = 0,
 tile_sz_y                 = 0,
 numtiles                  = 1,
 nproc_x                   = -1, ! must be on its own line
 nproc_y                   = -1, ! must be on its own line
 parent_id                 = 0,         1,          2,
 parent_grid_ratio         = 1,         3,          3,
 parent_time_step_ratio    = 1,         3,          3,
 i_parent_start            = 0,         00103,      18,
 j_parent_start            = 0,         187,        41,
 feedback                  = 1,
 num_moves                 = -99
 num_metgrid_levels        =  27,
 p_top_requested           = 5000,
 ptsgm                     = 42000
eta_levels                 = 1.0,       .9919699, .9827400, .9721600, .9600599,
                                        .9462600, .9306099,    .9129300, .8930600,
                                        .8708600, .8462000, .8190300,    7893100,
                                        .7570800, .7224600, .6856500, .6469100, .6066099,
                                        .5651600,    .5230500, .4807700, .4388600,
                                        .3978000, .3580500, 3200099,    .2840100,
                                        .2502900, .2190100, .1902600, .1640600,
                                        .1403600,  .1190600, .1000500, .0831600,
                                        .0682400, .0551200, .0436200,    .0335700,
                                        .0248200, .0172200, .0106300, .0049200,
                                        .0000000,
 use_prep_hybrid = F,
 num_metgrid_soil_levels = 4,
/
 &physics
 num_soil_layers           = 4,
 mp_physics                = 85,        85,       85,
 ra_lw_physics             = 98,        98,       98,
 ra_sw_physics             = 98,        98,       98,
 sf_sfclay_physics         = 88,        88,       88,
 sf_surface_physics        = 88,        88,       88,
 bl_pbl_physics            = 3,         3,        3,
 cu_physics                = 84,        84,       0,
 mommix                    = 1.0,       1.0,      1.0,
 h_diff                    = 1.0,       1.0,      1.0,
 gwd_opt                   = 2, 0,      0,
 sfenth                    = 0.0,       0.0,      0.0,
```

```
 nrads                               = 80,240,720,
 nradl                               = 80,240,720,
 nphs                                = 4,12,36,
 ncnvc                               = 4,12,36,
 movemin                             = 3,3,3,

! IMPORTANT: dt*nphs*movemin for domain 2 and 3 must be 540 and 180,
respectively
! AND the history output times (10800, 10800, 3600) must be
! divisible by dt*nphs*movemin for domains 1, 2 and 3

 gfs_alpha                           = 0.5,0.5,0.5,
 sas_pgcon                           = 0.55,0.2,0.2,
sas_mass_flux                        =0.5,0.5,0.5,
 co2tf                               = 1,

! ------------------------------------------------------------------------
!                          VORTEX TRACKER
! ------------------------------------------------------------------------

vortex_tracker=2,2,4,

! Options for vortex tracker #4: the revised centroid method:

! Vortex search options:
 vt4_radius                  = 250000.0, 250000.0, 250000.0 ! search radius in m
 vt4_weightexp               = 1.0,      1.0,      0.5, ! weight exponent (1=mass)

! Noise removal options:
 vt4_noise_pmin              = 85000.,   85000.,   85000.  ! min allowed MSLP
 vt4_noise_pmax              = 103000.,  103000.,  103000. ! max allowed MSLP
 vt4_noise_dpdr              = 0.6,      0.6,      0.6,   ! max dP/dx in Pa/m
 vt4_noise_iter              = 2,        2,        2, ! noise removal distance

! Disable nest movement at certain intervals to prevent junk in the output files:
 nomove_freq                 = 0.0,      6.0,      6.0, ! hours
/
 &dynamics
 non_hydrostatic             = .true.,    .true,       .true,
 euler_adv                   = .false.
 wp                          =  0,        0,           0,
 coac                        =  0.75,3.0,4.0,
 codamp                      =  6.4,      6.4,         6.4,
 terrain_smoothing           = 2,
/ &bdy_control
 spec_bdy_width              = 1,
 specified                   = .true. /
```

```
&namelist_quilt
poll_servers                =.false.
 nio_tasks_per_group        = 0,
 nio_groups                 = 4 /
&logging
compute_slaves_silent   =.true.
io_servers_silent       =.true.
stderr_logging          =.false.
 /
```

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, Lambert Conformal, polar- stereographic, Advanced Weather Interactive Processing System grids etc.), in GRIB format. This package also combines the parent and nest domains forecasts onto one combined domain grid.

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

7.2 How to Run UPP Using the Wrapper Script *unipost_wrapper*

The UPP wrapper script *unipost_wrapper* and the low-level script *run_unipost* are distributed in the tar file *hwrfv3.4a_utilities.tar.gz* and, following the procedure outlined in Chapter 2, will be expanded in the directory of *hwrf-utilities/wrapper_scripts* and *hwrf-utilities/scripts,* respectively.

Before running *unipost_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
UPP_ROOT
HWRF_UTILITIES_ROOT
FHR
DOMAIN_DATA
ATMOS_DOMAINS

Next use the *qrsh* command to connect to the computer's remote computation nodes (see Section 1.6). Note the number of processors to which the user should connect is defined as *UNI_CORES.* Currently UPP should be run with one processor.

Then run the wrapper script by typing its name, *unipost_wrapper*.

Note that other UPP scripts are distributed in the UPP release tar file *hwrfv3.4a_upp.tar.gz* but they do not perform all the processes required for HWRF.

A script named *run_grads* is provided for running GrADS to plot the UPP output. The users can find the script *run_grads* in the directory *hwrf-utilities/scripts*. Its wrapper script, *rungrads wrapper,* is located in the directory *hwrf-utilities/wrapper_scripts*. Before running *rungrads_wrapper*, check *global_vars.ksh* to make sure the following variables are correctly defined (see Appendix).

DOMAIN_DATA
FCST_LENGTH
FCST_INTERVAL
UPP_ROOT
GRADS_BIN

Then run the wrapper script by typing its name: *rungrads_wrapper,* which will call its low-level script *run_grads.*

7.3 Overview of the UPP Script

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.

2. Create and enter the work directory.

3. Copy the fix files and control file required by UPP
   The file *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 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.

4. Run *unipost.exe* for each forecast valid time for the parent, middle nest and inner nest domains. A namelist, *itag*, is created for each forecast valid time and domain, and then read in by *unipost.exe* from stdin (unit 5). This namelist contains 4 lines.
   - Name of the WRF output file to be post processed
   - Format of the WRF 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)

5. Run *copygb* to horizontally interpolate the native UPP output files to a variety of regular lat/lon grids.

6. Create a merged UPP output for the GFDL vortex tracker.

Output files in the working directory *${DOMAIN_DATA}/postprd/${fhr}:*

- The following three files are in GRIB format on the HWRF native grids.

    - WRFPRS_d01.${fhr} for the HWRF parent domain
    - WRFPRS_d02.${fhr} for the HWRF middle nest domain
    - WRFPRS_d03.${fhr} for the HWRF inner nest domain

- The following files are in GRIB format on regular lat/lon grids. The name convention is "forecast domain(s)"_"interpolation domain"_"resolution"_ "variables"_"forecast lead time". For example, *d01_d01_010_all.006* is the 6-hour HWRF parent domain forecast output that has been interpolated to a regular lat/lon grid covering an area similar to the one of the parent domain, with a horizontal resolution of 0.1 degree, containing all the variables present in the *unipost.exe* output file. "d02p" is a grid that is slightly larger than the middle nest domain. "t02" is a grid that is about 10x10 degrees and used later by the GFDL vortex tracker to calculate the track. When "variables"="all", it means all the variables will be included in the interpolated GRIB file. When only those variables required by the GFDL vortex tracker are retained, "variable"="sel". Files whose name start with merge are the result of combining two domains together to generate a single output file.

    - d01_d01_010_all.${fhr}
    - d01_d01_010_sel.${fhr}
    - d01_d01_025_all.${fhr}
    - d01_d01_025_sel.${fhr}
    - d01_d02p_003_all.${fhr}
    - d01_d02p_003_sel.${fhr}
    - d02_d01_010_all.${fhr}
    - d02_d01_010_sel.${fhr}
    - d02_d02_010_all.${fhr}
    - d02_d02_010_sel.${fhr}
    - d02_d02p_003_all.${fhr}
    - d02_d02p_003_sel.${fhr}
    - d03_d01_010_all.${fhr}
    - d03_d01_010_sel.${fhr}
    - d03_d02p_003_all.${fhr}

- d03_d02p_003_sel.${fhr}
- d03_d03_003_all.${fhr}
- d03_d03_003_sel.${fhr}
- merged_d01d02d03_t02_003_sel.${fhr}
- merged_d02d03_d01_010_sel.${fhr}
- merged_d02d03_d02p_003_sel.${fhr}

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 *unipost.exe*

FUNCTION:

De-staggers the HWRF native output (*wrfout_d01, wrfout_d02, or wrfout_d03*), interpolates it vertically to pressure levels, computes derived variables, and outputs in GRIB format.

INPUT:

Table *${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_eta_micro_lookup.dat*

unipost control file *${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_cntrl.hurcn*

HWRF native output (*wrfout_d01*, *wrfout_d02* or *wrfout_d03*)

namelist *itag*

OUTPUT:

HWRF output in GRIB format *WRFPRS_d01.${fhr}, WRFPRS_d02.${fhr}* or *WRFPRS_d03.${fhr}*

USAGE:

${SCRATCH}/HWRF/ UPP/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 for function a:

User-specified grid *(${hr_grid})*

unipost.exe output *(WRFPRS_d01.${fhr}, WRFPRS_d02.${fhr}* or *WRFPRS_d03.${fhr})*

INPUT for function b:

User-specified grid *(${hr_grid})*

Two GRIB files, for example*, d01_d01_010_all.000* and *d03_d01_010_all.000*

OUTPUT:

GRIB file on the grid of ${hr_grid}. See "*Output files* in the working directory *${DOMAIN_DATA}/postprd/${fhr}*"

USAGE:

a) *${SCRATCH}/HWRF/UPP/bin/copygb.exe -xg"${hr_grid}" input_GRIB_file out_GRIB_file*

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}*.

   ${SCRATCH}/HWRF/ UPP/bin/copygb.exe -g"${hr_grid}" –xM wrfprs_d01.${fhr} wrfprs_d02.${fhr} wrfprs.${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 850 mb and 700 mb, MSLP, and geopotential height at 850 and 700 mb. Secondarily, wind speed at 10 m, and 850 mb and 700 mb are used. Winds at 500 mb are used, together with other parameters, for advecting the storm and creating a first guess position for all times beyond initialization. Many parameters are used in order to provide more accurate position estimates for weaker storms, which often have poorly defined structures/centers.

Besides the forecast file in GRIB format, the vortex tracker also ingests a GRIB index file, which is generated by running the program *grbindex*. The GRIB utility *wgrib* is also used for preparing data for the tracker. Both *grbindex* and *wgrib* were developed by NCEP and are distributed by the DTC as part of the hwrf-utilities.

This version of the tracker contains added capabilities of tracking cyclogenesis and identifying cyclone thermodynamic phases. The identification of cyclone thermodynamic phases requires that the input data contain temperature every 50 hPa from 300 to 500 mb (for the “vtt” scheme) or the geopotential height every 50 mb from 300 to 900 mb (for the “cps” scheme) (see Section 8.4).

8.2 How to Run the GFDL Vortex Tracker Using the Wrapper Script

The HWRF scripts come in the tarfile *hwrfv3.4a_utilities.tar.gz* and, following the procedures outlined in Chapters 1 and 2, will be expanded in the directories *${SCRATCH}/HWRF/hwrf-utilities/wrapper_scripts* and *${SCRATCH}/HWRF/hwrf-utilities/scripts.*

Before running tracker_wrapper, check global_vars.ksh to make sure the following variables are correctly defined. (See Appendix)

HWRF_SCRIPTS
HWRF_UTILITIES_ROOT
TRACKER_ROOT
DOMAIN_DATA
START_TIME
ATCFNAME
SID

Then run the wrapper script by typing its name, *tracker_wrapper*.

The tracker runs the combined domain. It produces a 3-hourly track and a 6-hourly track for the entire forecast length and another 3-hourly one for the 12-hr forecast, using the UPP output *merge_d01d02d03_t02_sel.${fhr}* (see Section 7.3). The track for the 12-hr forecast is used in the vortex relocation procedure for the following cycle.

8.3 Overview of the Script *tracker.ksh*

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

1. Initialize the function library and check to see if all the environment variables are set and the executables exist.
2. Create and enter the work directory.
3. Create a tracker namelist file.
4. Concatenate the UPP output files into one GRIB file that contains all the forecast lead times.
5. Run *grbindex* to get a GRIB index file for the GRIB file generated in 4.
6. Create a file, *fcst_minutes*, which contains the forecast lead times the tracker will process.
7. Link the input files (see GFDL vortex tracker software input description).
8. Run the tracker executable *hwrf_gettrk.exe*.
9. Output will be generated in *${DOMAIN_DATA}/gvtprd*.

8.4 How to Generate Phase Space Diagnostics

The released wrapper and low-level scripts do not include the phase space diagnostics. To use this function, the user should either modify the scripts or run the following procedures manually.

1) In the GFDL vortex tracker namelist set the items listed below.
   phaseflag='y'
   phasescheme='both' or 'cps' or 'vtt'
   wcore_depth=1.0
2) If phasescheme is set to 'cps', run *hwrf_vint.exe* (see Section 8.6.2) to vertically interpolate the geopotential from 300 to 900 mb at a 50 mb interval. Then append these geopotential variables to the tracker's GRIB format input file.
3) If phasescheme is set to 'vtt' , run *hwrf_vint.exe* (see Section 8.6.2) to vertically interpolate the temperature from 300 to 500 mb at a 50 mb interval. Then run *hwrf_tave.exe* (see Section 8.6.3) to obtain the average temperature between 300 and 500 mb. This average temperature field is appended to the tracker's GRIB format input file.
4) If phasescheme is set to 'both', then both steps 2) and 3) are needed.
5) When the phase space diagnostics is performed, the output will be generated in *fort.64* as fields 37-41 (see Section 8.6.1).

8.5 How to Run the Tracker in Cyclogenesis Mode

The released wrapper and low-level scripts do not include running the tracker in cyclogenesis mode. To use this function, the user should either modify the scripts or run the following procedures manually.

1) In the GFDL vortex tracker namelist set the items listed below.
   trkrinfo%westbd
   trkrinfo%eastbd
   trkrinfo%southbd
   trkrinfo%northbd
   They are the boundaries for searching for new storms in cyclogenesis mode. They do not need to match the boundaries of your grid.
2) In the GFDL vortex tracker namelist, set the item trkrinfo%type=tcgen or trkinfo%type=midlat (for the difference between "tcgen" and "midlat", see Section 8.6.1).
3) The tracker in cyclogenesis mode requires that the files *fort.12* and *fort.14* exist in the working directory, but these two files can be blank, as created by the commands "*touch fort.12*" and "*touch fort.14*", respectively.
4) In addition to *fort.64* and *fort.69*, another ATCF format output file, *fort.66*, will be produced by the tracker when it runs in cyclogenesis mode.

8.6 Executables

8.6.1 *hwrf_gettrk.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 Irene of 20.6N and 70.6W.

NHC 09L IRENE 20110823 1200 206N 0706W 295 051 0978 1008 0556 44 028 0334 0222 0222 0241 D 0167 0111 0111 0130 72 280N 780W 0083 0056 0037 0065

fort.14: TCVitals file used for tropical cyclonegenesis tracking. This file is not used in HWRF's operational configuration. 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 *grbindex*.

NAMELIST:

inp%bcc	First 2 digits of the year for the initial time of the forecast (e.g., the "20" in "2011")
inp%byy	Last 2 digits of the year for the initial time of the forecast (e.g., the "11" in "2011")
inp%bmm	2-digit month (01, 02, etc) for the initial time of the forecast
inp%bdd	2-digit day for the initial time of the forecast
inp%bhh	2-digit hour for the initial time of the forecast

inp%model	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) NGM, (6) NAM, (7) NOGAPS, (8) GDAS, (10) NCEP Ensemble, (11) ECMWF Ensemble, (13) SREF Ensemble, (14) NCEP Ensemble, (15) CMC, (16) CMC Ensemble, (18) HWRF Ensemble, (19) HWRF-DAS (HDAS), (20) Ensemble RELOCATION (21) UKMET hi-res (NHC)
inp%lt_units	'hours' or 'minutes', this defines the lead time units used by the PDS in your GRIB header
inp%file_seq	'onebig' or 'multi', this specifies if the tracker will process one big input file or multiple files for each individual lead times. ‘onebig’ is used as the default method in the community HWRF scripts.
inp%modtyp	Type of the model. Either 'global' or 'regional'. For HWRF, choose ‘regional’.
inp%nesttyp	Type of the nest grid. Either ‘moveable’ or ‘fixed’. For HWRF, choose ‘moveable’.
fnameinfo%gmodname	Defines the model name in the input files, e.g., 'hwrf'. Only when inp%file_seq='multi'
fnameinfo%rundescr	Describe the model runs in the input files, e.g., 'combined'. Only when inp%file_seq= 'multi'
fnameinfo%atcfdescr	Describe the storm information in the input files, e.g., 'irene09l'. Only when inp%file_seq='multi'
atcfnum	Obsolete; can be set to any integer
atcfname	Character model ID that will appear in the ATCF output (e.g., GFSO, HWRF, AHW, HCOM etc)
atcfymdh	10-digit yyyymmddhh date that will be used in output text track files

Atcffreq	Frequency (in centahours) of output for atcfunix.Default value is 600 (six hourly).
trkrinfo%westbd	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.
trkrinfo%eastbd	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.
trkrinfo%northbd	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.
trkrinfo%southbd	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.
trkrinfo%type	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. 'tcgen' and 'midlat' run in genesis mode and will look for new storms in addition to tracking from TCVitals. 'tcgen' will look for all parameters at the various vertical levels, while 'midlat' will only look for mslp and no checks are performed to differentiate tropical from non-tropical cyclones.For HWRF, choose 'tracker'.
trkrinfo%mslpthresh	Threshold for the minimum MSLP gradient (units mb/km) that must be met in order to continue tracking.
trkrinfo%v850thresh	Threshold for the minimum azimuthally-average 850 mb cyclonic tangential wind speed (m/s) that must be exceeded in order to keep tracking.
trkrinfo%gridtype	'global' or 'regional', this defines the type of domain grid. For HWRF or other limited area models, choose 'regional'.
trkrinfo%contint	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.
trkrinfo%out_vit	This is only set to 'y' if the tracker is running in genesis mode, and it tells the tracker to write out a "TCVitals" record for any storms that it finds at tau = 00h in a forecast. For HWRF, choose ‘n’.
phaseflag	'y' or 'n', tells the program whether or not to determine the cyclone

	thermodynamic phase
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'
wcore_depth	The contour interval (in deg K) used in determining if a closed contour exists in the 300-500 mb temperature data, for use with the vtt scheme
structflag	'y' or 'n', tells the program whether or not to determine the cyclone thermodynamic structure.
Ikeflag	'y' or 'n', tells the program whether or not to calculate the Integrated Kinetic Energy (IKE) and Storm Surge Damage Potential (SDP).
use_waitfor	'y' or 'n', for waiting for input files. Use 'n' unless for real-time operational runs
Verb	Level of detail printed to terminal. Choose from 0 (no output),1 (error messages only), 2 (more messages) ,3 (all messages).

OUTPUT:

Two files are output, both are in a modified ATCF format: *fort.69*; and *fort.64*. When the tracker runs in cyclogenesis mode, it produces another ATCF format file: *fort.66*. And if the "ikeflag" is set to "y" in the namelist, still another output file will be created: *fort.74*.

A sample of the vortex tracker output *fort.69* is listed below:

AL, 09, 2011082312, 03, HCOM, 00000, 204N, 706W, 87, 978, XX, 34, NEQ, 0103, 0077, 0058, 0095, 0, 0, 24
AL, 09, 2011082312, 03, HCOM, 00000, 204N, 706W, 87, 978, XX, 50, NEQ, 0058, 0040, 0031, 0055, 0, 0, 24
AL, 09, 2011082312, 03, HCOM, 00000, 204N, 706W, 87, 978, XX, 64, NEQ, 0043, 0025, 0016, 0042, 0, 0, 24
AL, 09, 2011082312, 03, HCOM, 00600, 208N, 714W, 95, 964, XX, 34, NEQ, 0155, 0100, 0058, 0145, 0, 0, 21
AL, 09, 2011082312, 03, HCOM, 00600, 208N, 714W, 95, 964, XX, 50, NEQ, 0066, 0057, 0037, 0060, 0, 0, 21
AL, 09, 2011082312, 03, HCOM, 00600, 208N, 714W, 95, 964, XX, 64, NEQ, 0046, 0033, 0028, 0042, 0, 0, 21
AL, 09, 2011082312, 03, HCOM, 01200, 208N, 722W, 94, 963, XX, 34, NEQ, 0123, 0096, 0060, 0109, 0, 0, 23
AL, 09, 2011082312, 03, HCOM, 01200, 208N, 722W, 94, 963, XX, 50, NEQ, 0069, 0048, 0049, 0062, 0, 0, 23
AL, 09, 2011082312, 03, HCOM, 01200, 208N, 722W, 94, 963, XX, 64, NEQ, 0045, 0032, 0036, 0046, 0, 0, 23
AL, 09, 2011082312, 03, HCOM, 01800, 214N, 728W, 100, 960, XX, 34, NEQ, 0093, 0082, 0060, 0084, 0, 0, 19
AL, 09, 2011082312, 03, HCOM, 01800, 214N, 728W, 100, 960, XX, 50, NEQ, 0054, 0051, 0042, 0050, 0, 0, 19
AL, 09, 2011082312, 03, HCOM, 01800, 214N, 728W, 100, 960, XX, 64, NEQ, 0040, 0037, 0034, 0037, 0, 0, 19
AL, 09, 2011082312, 03, HCOM, 02400, 218N, 734W, 94, 959, XX, 34, NEQ, 0131, 0097, 0063, 0116, 0, 0, 18
AL, 09, 2011082312, 03, HCOM, 02400, 218N, 734W, 94, 959, XX, 50, NEQ, 0061, 0054, 0043, 0056, 0, 0, 18
AL, 09, 2011082312, 03, HCOM, 02400, 218N, 734W, 94, 959, XX, 64, NEQ, 0040, 0035, 0035, 0039, 0, 0, 18
AL, 09, 2011082312, 03, HCOM, 03000, 225N, 741W, 99, 955, XX, 34, NEQ, 0141, 0106, 0065, 0105, 0, 0, 24
AL, 09, 2011082312, 03, HCOM, 03000, 225N, 741W, 99, 955, XX, 50, NEQ, 0069, 0055, 0045, 0057, 0, 0, 24
AL, 09, 2011082312, 03, HCOM, 03000, 225N, 741W, 99, 955, XX, 64, NEQ, 0044, 0044, 0036, 0041, 0, 0, 24
AL, 09, 2011082312, 03, HCOM, 03600, 232N, 749W, 109, 952, XX, 34, NEQ, 0135, 0104, 0075, 0103, 0, 0, 18
AL, 09, 2011082312, 03, HCOM, 03600, 232N, 749W, 109, 952, XX, 50, NEQ, 0070, 0066, 0048, 0055, 0, 0, 18
AL, 09, 2011082312, 03, HCOM, 03600, 232N, 749W, 109, 952, XX, 64, NEQ, 0050, 0050, 0038, 0042, 0, 0, 18

Column 1: basin name. "AL" represents Atlantic and “EP” northeast Pacific.
Column 2: ATCF storm ID number. Irene was the 9th storm in the Atlantic Basin in 2011.
Column 3: model starting time.
Column 4: constant and 03 simply indicates that this record contains model forecast data.
Column 5: model ATCF name.
Column 6: forecast lead time in hours multiplied by 100 (e.g, 00900 represents 9 .00 hr).
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. Currently, that storm type character string is only used for the observed storm data in the NHC Best Track data set.
Column 12: thresholds wind speed in knots, 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 14-17: wind radii (in nm) for the threshold winds in each quadrant.
Column 18-19: not used.
Column 20: radius of maximum winds, in nautical miles.

A sample of the vortex tracker output *fort.64* is listed below:

```
AL, 09, 2011082312, 03, HCOM, 000, 204N,  706W,  87,  978, XX,  34, NEQ, 0103, 0077, 0058, 0095,   0,   0, 24,  0,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 000, 204N,  706W,  87,  978, XX,  50, NEQ, 0058, 0040, 0031, 0055,   0,   0, 24,  0,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 000, 204N,  706W,  87,  978, XX,  64, NEQ, 0043, 0025, 0016, 0042,   0,   0, 24,  0,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 006, 208N,  714W,  95,  964, XX,  34, NEQ, 0155, 0100, 0058, 0145,   0,   0, 21,  0 ,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 006, 208N,  714W,  95,  964, XX,  50, NEQ, 0066, 0057, 0037, 0060,   0,   0, 21,  0 ,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 006, 208N,  714W,  95,  964, XX,  64, NEQ, 0046, 0033, 0028, 0042,   0,   0, 21,  0 ,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 012, 208N,  722W,  94,  963, XX,  34, NEQ, 0123, 0096, 0060, 0109,   0,   0, 23,  0 ,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 012, 208N,  722W,  94,  963, XX,  50, NEQ, 0069, 0048, 0049, 0062,   0,   0, 23,  0 ,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999

AL, 09, 2011082312, 03, HCOM, 012, 208N,  722W,  94,  963, XX,  64, NEQ, 0045, 0032, 0036, 0046,   0,   0, 23,  0 ,  0,   ,  0,   ,  0,
0,       ,  ,   ,   ,  0,  0,  0,  0,     THERMO PARAMS,   -9999,   -9999, -9999, U, 10, DT, -999
```

Column 1-20: same as fort.69 except that column 6, the forecast lead time, instead of being a 5-digit integer as in fort.69, is a 3-digit integer.
Column 21-35: space fillers.
Column 36: “THERMO PARAMS,” indicating that thermodynamics parameters will follow.
Column 37-39: The three cyclone phase space parameters, and all values shown have been multiplied by a factor of 10. The values are listed below.

(1) Parameter B (left-right thickness asymmetry)
(2) Thermal wind (warm/cold core) value for lower troposphere (900-600 mb)

(3) Thermal wind value for upper troposphere (600-300 mb)

Column 40: Presence of a warm core. In this sample it is "U", which stands for "undetermined", meaning the warm core check was not performed. When the warm core check is performed, this field will be either 'Y' or 'N', indicating whether the warm core is identified or not.

Column 41: Warm core strength x 10 (in degrees). It indicates the value of the contour interval that was used in performing the check for the warm core in the 300-500 mb layer.

Column 42-43: Constant strings.

A sample of the vortex tracker output *fort.66* is listed below:

```
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 000, 204N,  706W,  87,  978, XX,  34, NEQ, 0103, 0077, 0058,
0095, 1005,  56,  24,  -999,  -9999,  -9999, U,  288,  39, 1191, 6649, 1186, 5714
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 000, 204N,  706W,  87,  978, XX,  50, NEQ, 0058, 0042, 0032,
0054, 1005,  56,  24,  -999,  -9999,  -9999, U,  288,  39, 1191, 6649, 1186, 5714
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 000, 204N,  706W,  87,  978, XX,  64, NEQ, 0043, 0027, 0019,
0041, 1005,  56,  24,  -999,  -9999,  -9999, U,  288,  39, 1191, 6649, 1186, 5714
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 006, 208N,  714W,  94,  965, XX,  34, NEQ, 0156, 0096, 0059,
0145,  976,  17,  21,  -999,  -9999,  -9999, U,  292,  45, 1164, 3306, 1116, 3302
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 006, 208N,  714W,  94,  965, XX,  50, NEQ, 0065, 0056, 0037,
0058,  976,  17,  21,  -999,  -9999,  -9999, U,  292,  45, 1164, 3306, 1116, 3302
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 006, 208N,  714W,  94,  965, XX,  64, NEQ, 0047, 0031, 0030,
0042,  976,  17,  21,  -999,  -9999,  -9999, U,  292,  45, 1164, 3306, 1116, 3302
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 012, 209N,  722W,  93,  964, XX,  34, NEQ, 0123, 0098, 0059,
0104,  979,  21,  21,  -999,  -9999,  -9999, U,  282,  42, 1187, 3668, 1122, 2694
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 012, 209N,  722W,  93,  964, XX,  50, NEQ, 0069, 0053, 0047,
0058,  979,  21,  21,  -999,  -9999,  -9999, U,  282,  42, 1187, 3668, 1122, 2694
TG, 0001, 2011082312_F000_204N_0706W_FOF, 2011082312, 03, HCOM, 012, 209N,  722W,  93,  964, XX,  64, NEQ, 0044, 0033, 0033,
0044,  979,  21,  21,  -999,  -9999,  -9999, U,  282,  42, 1187, 3668, 1122, 2694
```

Column 1: "TG", the basin id for cyclogenesis (when trkrinfo%type is set to "midlat", this id is named "ML").

Column 2: the number of cyclogenesis the tracker identified.

Column 3: the ID for the cyclogenesis, ${YYYYMMDDHH}_F${FFF}_$Lat_$Lon_FOF where YYYYMMDDHH, FFF, Lat and Lon are the model initialization time, the forecast lead time, the latitude and the longitude, respectively, in which the cyclogenesis was first identified.

Column 4-18: same as Columns 3-17 in *fort.64*.

Column 19: pressure of last closed isobar (in mb).

Column 20: radius of last closed isobar (nm).

Column 21: radius of maximum wind (nm).

Column 22-24: The cyclone phase space parameters, and all values shown have been multiplied by a factor of 10. The values are listed below.

(1) Parameter B (left-right thickness asymmetry)

(2) Thermal wind (warm/cold core) value for lower troposphere (900-600 mb)

(3) Thermal wind value for upper troposphere (600-300 mb)

Column 25: Presence of a warm core. In this sample it is "U", which stands for "undetermined", meaning the warm core check is not performed. When the warm core check is performed, this field will be either 'Y' or 'N', indicating whether the warm core is identified or not.

Column 26: storm moving direction (in degrees).

Column 27: storm moving speed (in ms^{-1}).

Column 28: mean 850 hpa vorticity (s^{-1}x10e5).

Column 29: max (gridpoint) 850 hpa vorticity (s^{-1}x10e5).

Column 28: mean 700 hpa vorticity (s^{-1}x10e5).
Column 29: max (gridpoint) 700 hpa vorticity (s^{-1}x10e5).

A sample of the vortex tracker output *fort.74* is listed below:

```
AL, 09, 2011082312, 03, HCOM, 000, 204N,  706W,  87,  978, XX,  91, IKE,    0,   23,   34,   16,    5,    0,    0,    0, 2039N,  7062W
AL, 09, 2011082312, 03, HCOM, 006, 208N,  714W,  94,  965, XX,  91, IKE,    0,   28,   42,   25,    8,    0,    0,    0, 2081N,  7142W
AL, 09, 2011082312, 03, HCOM, 012, 209N,  722W,  93,  964, XX,  91, IKE,    0,   28,   44,   25,    8,    0,    0,    0, 2088N,  7220W
AL, 09, 2011082312, 03, HCOM, 018, 213N,  728W,  99,  962, XX,  91, IKE,    0,   25,   46,   19,    9,    0,    0,    0, 2131N,  7276W
AL, 09, 2011082312, 03, HCOM, 024, 218N,  733W,  92,  962, XX,  91, IKE,    0,   27,   50,   23,    8,    0,    0,    0, 2179N,  7333W
AL, 09, 2011082312, 03, HCOM, 030, 225N,  741W,  97,  959, XX,  91, IKE,    0,   28,   51,   26,    9,    0,    0,    0, 2245N,  7415W
AL, 09, 2011082312, 03, HCOM, 036, 231N,  749W,  95,  961, XX,  91, IKE,    0,   29,   51,   27,   11,    0,    0,    0, 2314N,  7488W
AL, 09, 2011082312, 03, HCOM, 042, 239N,  756W, 100,  956, XX,  91, IKE,    0,   29,   54,   28,   11,    0,    0,    0, 2387N,  7562W
AL, 09, 2011082312, 03, HCOM, 048, 248N,  762W, 107,  953, XX,  91, IKE,    0,   30,   58,   30,   14,    0,    0,    0, 2479N,  7621W
AL, 09, 2011082312, 03, HCOM, 054, 258N,  767W, 111,  949, XX,  91, IKE,    0,   32,   62,   34,   16,    0,    0,    0, 2575N,  7668W
AL, 09, 2011082312, 03, HCOM, 060, 267N,  770W, 113,  946, XX,  91, IKE,    0,   33,   65,   38,   18,    0,    0,    0, 2668N,  7696W
AL, 09, 2011082312, 03, HCOM, 066, 277N,  773W, 111,  944, XX,  91, IKE,    0,   34,   67,   40,   21,    0,    0,    0, 2769N,  7731W
AL, 09, 2011082312, 03, HCOM, 072, 286N,  774W, 114,  944, XX,  91, IKE,    0,   35,   68,   42,   23,    0,    0,    0, 2864N,  7742W
AL, 09, 2011082312, 03, HCOM, 078, 296N,  775W, 113,  941, XX,  91, IKE,    0,   35,   73,   43,   22,    0,    0,    0, 2959N,  7753W
AL, 09, 2011082312, 03, HCOM, 084, 304N,  774W, 107,  944, XX,  91, IKE,    0,   35,   74,   43,   22,    0,    0,    0, 3037N,  7742W
AL, 09, 2011082312, 03, HCOM, 090, 312N,  774W, 108,  941, XX,  91, IKE,    0,   36,   77,   46,   23,    0,    0,    0, 3119N,  7745W
AL, 09, 2011082312, 03, HCOM, 096, 320N,  773W, 107,  942, XX,  91, IKE,    0,   37,   79,   51,   26,    0,    0,    0, 3198N,  7728W
AL, 09, 2011082312, 03, HCOM, 102, 328N,  772W, 111,  938, XX,  91, IKE,    0,   38,   78,   53,   28,    0,    0,    0, 3278N,  7719W
AL, 09, 2011082312, 03, HCOM, 108, 336N,  769W, 111,  937, XX,  91, IKE,    0,   37,   76,   51,   30,    0,    0,    0, 3360N,  7690W
AL, 09, 2011082312, 03, HCOM, 114, 347N,  766W, 106,  939, XX,  91, IKE,    0,   35,   68,   43,   21,    0,    0,    0, 3473N,  7664W
AL, 09, 2011082312, 03, HCOM, 120, 361N,  764W,  90,  950, XX,  91, IKE,    0,   32,   57,   35,   10,    0,    0,    0, 3611N,  7642W
AL, 09, 2011082312, 03, HCOM, 126, 375N,  764W,  69,  957, XX,  91, IKE,    0,   27,   42,   24,    2,    0,    0,    0, 3745N,  7637W
```

Column 1-11: Same as *fort.64*.
Column 12-13: fixed fields.
Column 14: wind damage potential (wdp) (not computed in this version, therefore is always zero).
Column 15: storm surge damage potential (SDP) (multiplied by 10).
Column 16-18: IKE, in terajoule, for 10 ms^{-1}, 18 ms^{-1} and 33 ms^{-1} winds, respectively.
Column 19-21: IKE for 25-40 ms^{-1}, 41-54 ms^{-1} and 55 ms^{-1} winds, currently not computed, therefore are always zero
Column 22-23: vortex center position (latitude and longitude multiplied by 100).

USAGE:
hwrf_gettrk.exe < namelist

8.6.2 *hwrf_vint.exe*

Program to interpolate from various pressure levels onto a regularly spaced grid, with 50-hpa vertical level intervals. Each run only processes one lead time. Therefore it is necessary to use this executable separately for all lead times.

INPUT:
fort.11: GRIB file containing the postprocessed HWRF output that must contain at least two levels temperature data: 300 and 500 hpa.
fort.16: text file containing the number of input pressure levels.
fort.31: index file of *fort.11*
Namelist: generated by *echo "&timein ifcsthour=${fhour} iparm=${gparm}/"* where *${fhour}* is the forecast lead time and *${gparm}* is the variable to be processed. For phase space diagnostics, geopotential height (when phasescheme='cps', *${gparm)*=7) or temperature (when phasescheme='vtt',

${gparm}=11) or both (when phasescheme='both') need to be processed.

OUTPUT:
fort.51: GRIB file that contains the temperature data on vertical levels 300, 350, 400, 450 and 500 hpa.

USAGE:
hwrf_vint.exe < namelist

8.6.3 *hwrf_tave.exe*

Program to vertically average temperature in the 500-300 hpa layer.

INPUT:
fort.11: GRIB file containing the temperature at least at levels 300, 350, 400, 450 and 500 hpa. This file can be generated by *hwrf_vint.exe*
fort.16: text file containing the number of input pressure levels.
fort.31: index file of *fort.11*
namelist: generated by the command: *echo "&timein ifcsthour=${fhour}, iparm=11/" > ${namelist}*

OUTPUT:
fort.51: GRIB file containing the mean temperature in the 300-500 hpa layer.

USAGE:
hwrf_tave.exe < namelist

8.7 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: *gfdl-vortextracker/trk_plot/plottrak.*

To use atcf_plot to plot the storm's track:

- Enter the directory *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 *plottrack.ctl* and the newly created *plottrak.ix* map file need to be in the directory where the script below is run.

- 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 in for GrADS (for example, */contrib/grads/lib)*.
  3. *scrdir:* path to the working directory (for example, */home/${USER}/HWRF/src/gfdl-vortextracker/trk_plot/plottrak)*.
  4. *plotdir*: path to the directory where the plot files will be created (for example, */home/${USER}/HWRF/src/gfdlvortextracker/trk plot/plottrak/tracks)*.
- Edit *atcfplot.gs* to define the following paths:
  1, *rundir:* same as scrdir in *atcfplot.sh. Note rundir must end with a "/".*
  2. *_netdir:* same as plotdir in *atcfplot.sh. Note netdir must end with a "/".*
- 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 settting 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 2011 al*" to plot the track files *aal${SID}2011*.dat in the *${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 "HCOM". 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 ATCF model name 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 "HCOM", and manually replace the string "HCOM" with "USER".

Appendix

The following environment variables are defined in *hwrf_utilities/wrapper_scripts/global_vars.ksh*

Note variable names in **bold** are independent variables that need to be explicitly defined by the user. The remaining variables are defined using these independent variables and should not be edited.

Variable Group	Variable Name	Description	Example/Default
HWRF Source Paths	**HWRF_SRC_DIR**	Path to the HWRF source code	/lfs1/projects/dtc-hurr/timbrown/cc/CC_20120525_01/sorc
	HWRF_OUTPUT_DIR	Path to the HWRF output	/lfs1/projects/dtc-hurr/dstark/CC_20120525_01/results
	HWRF_DATA_DIR	Pah to the HWRF input data sets	/lfs1/projects/dtc-hurr/datasets
	HWRF_SCRIPTS	Path to low-level scripts	${HWRF_UTILITIES_ROOT}/scripts
Storm Info	**START_TIME**	HWRF run starting time	2011082312
	START_TIME_MINUS6	6 hr before HWRF run starting time	2011082306
	STORM_NAME	Storm name issued by National Hurricane Center	IRENE
	SID	Storm ID	09L
	BASIN	Storm ocean basin (Al or EP)	AL
	FCST_LENGTH	Forecast length in hour	126
	FCST_INTERVAL	Forecast output interval in hour	6
HWRF Data Paths	**GEOG_DATA_PATH**	Path to geographical fix data	/lfs0/projects/WPS/GEOG
	GFS_DATA_DIR	Path to work directory of ungrib	${HWRF_OUTPUT_DIR}/${SID}/${START_TIME}/ungribprd
	GFS_SPECTRAL_DIR	Path to the GFS spectral input data	${HWRF_DATA_DIR}/GFS/2011//spectral
	GFS_GRIDDED_DIR	Path to the GFS GRIB input data	${HWRF_DATA_DIR}/GFS/2011/gridded
	OCEAN_FIXED_DIR	Path to the ocean fix data	${HWRF_DATA_DIR}/fix/ocean
	LOOP_CURRENT_DIR	Path to the ocean loop current and warm/cold core rings data	${HWRF_DATA_DIR}/Loop_current
	TCVITALS	Path to the TCVitals files	${HWRF_DATA_DIR}/Tcvitals
	CRTM_FIXED_DIR	Path to the CRTM data	${HWRF_DATA_DIR}/fix/HWRF_v3.4a

Output Paths	CYCLE_DATA	Path to the previous cycle output	${HWRF_OUTPUT_DIR}/${SID}/${START_TIME_MINUS6}
	DOMAIN_DATA	Path to the current cycle's output	${HWRF_OUTPUT_DIR}/${SID}/${START_TIME}
Component Paths	WRF_ROOT	Path to WRFV3 source	${HWRF_SRC_DIR}/WRFV3
	WPS_ROOT	Path to WPS source	${HWRF_SRC_DIR}/WPSV3
	UPP_ROOT	Path to UPP souce	${HWRF_SRC_DIR}/UPP
	HWRF_UTILITIES_ROOT	Path to HWRF-utilities source	${HWRF_SRC_DIR}/hwrf-utilities
	GSI_ROOT	Path to GSI source	${HWRF_SRC_DIR}/GSI
	POMTC_ROOT	Path to POM-TC source	${HWRF_SRC_DIR}/pomtc
	TRACKER_ROOT	Path to tracker source code	${HWRF_SRC_DIR}/gfdl-vortextracker
	COUPLER_ROOT	Path to coupler source code	${HWRF_SRC_DIR}/ncep-coupler
Processors	**MPIRUN**	Command used to run parallel code	mpiexec
	IO_FMT	IO format (1 for binary, 2 for NetCDF)	2 (Only 2 is currently supported)
	GEOGRID_CORES	Number of cores to run geogrid	12
	METGRID_CORES	Number of cores to run metgrid	1
	GSI_CORES	Number of cores to run GSI	24
	REAL_CORES	Number of cores to run real	1
	WRF_CORES	Number of cores to run WRF	202
	UNI_CORES	Number of cores to run UPP	1
WRF I/O Options	**IO_SERVERS**	If use I/O server	No
	IOSRV_GROUPS	Number of I/O server groups	0
	IOSRV_PERGRP	Number of I/O servers per group	0
Model Options	**ATMOS_DOMAINS**	Number of atmosphere domain grids	3
	PREP_HYB	Logical variable to determine the use of GFS data in hybrid vertical levels	F (Only F is currently supported)
	RUN_PREP_HYBRID	Same as above	F (Only F is currently supported)
	use_extended_eastatl	Logical variable to determine the use of the extended ocean domain grid when the storm is in Eastern Atlantic	F
	USE_SAT	Logical variable to determine if satellite radiance s will be included created during postprocessing	F
GSI Options	**USE_GSI**	Logical variable to determine if GSI will be run	T
	OBS_ROOT	Path to the observational data for GSI	${HWRF_DATA_DIR}/GFS/2011/obs/${START_TIME}
	PREPBUFR	Path to the conventional observational data in prepbufr format	${HWRF_DATA_DIR}/GFS/2011/obs/${START_TIME}/gfs.${START_TIME}.prepbufr.nr
	BK_DIR	Path to the background directory for GSI	${HWRF_OUTPUT_DIR}/${SID}/${START_TIME}/relocateprd
	FIX_ROOT	Path to the fix data files for GSI	${HWRF_DATA_DIR}/fix/HWRF_v3.4a
	bk_core	Dynamic core option for GSI	NMM (Only NMM is supported)
	bkcv_option	Background error covariance option for GSI	NAM (Only NMM is supported)
Path to GrADS Tools	**GRADS_BIN**	Path to GrADS	/home/dtc/grads-2.0.1.oga.1/Contents
	GADDIR	Path to GrADS libraries	/home/dtc/grads-2.0.1.oga.1/Classic/data