NOAA Technical Memorandum OAR GSD-43

COMMUNITY HWRF USERS GUIDE V3.5A

AUGUST 2013

THE DEVELOPMENT TESTBED CENTER

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

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

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

Community HWRF Users' Guide V3.5a

August 2013

The Developmental Testbed Center

Contributors to this Guide:

Ligia Bernardet[1], *Shaowu Bao*[2], *Richard Yablonsky*[3], *Don Stark*[4], *and Timothy Brown*[1]

Please send questions to: *wrfhelp@ucar.edu*

[1] *NOAA/ESRL/GSD, Developmental Testbed Center and CIRES/CU*

[2] *NOAA/ESRL/GSD and CIRES/CU and currently affiliated with S. Carolina Coastal University*

[3] *University of Rhode Island*

[4] *NCAR/RAL/JNT, Developmental Testbed Center*

Acknowledgments: The authors wish to thank Karen Griggs of NCAR for offering her desktop publishing expertise in the preparation of this document.

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 Centers for Environmental Prediction (NCEP) of the National Weather Service (NWS) to provide numerical guidance to the National Hurricane Center for the forecasting of tropical cyclones' track, intensity, and structure. HWRF v3.5a and this Users' Guide match the operational 2013 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 the WRF model (WRF-NMM), with a parent and two nest domains. The parent domain covers roughly 80^o x 80^o on a rotated latitude/longitude E-staggered grid. The location of the parent domain is determined based on the initial position of the storm and on the National Hurricane Center (NHC) forecast of the 72-h position, if available. The middle nest domain, of about 11° x 10°, and the inner nest domain, of about 7.2° x 6.5°, move along with the storm using two-way interactive nesting. 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 dynamic time steps are 45, 15, and 5 s, respectively, for the parent, middle nest, and inner nest domains.

The model physics originated primarily from the Geophysical Fluid Dynamics Laboratory (GFDL) hurricane model, and 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, which also employ 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. The HWRF physics includes parameterization of dissipative heating.

The NCEP Global Data Assimilation System (GDAS) 6-h forecast, initialized 6-h before a given HWRF initialization, is used to generate the initial conditions for the hurricane model in the operational configuration. When GDAS data is not available, the NCEP Global Forecast System (GFS) analysis can be used. The first guess is modified using the HWRF Data Assimilation System (HDAS) by ingesting observations in a hybrid ensemble-variational data assimilation system called Gridpoint Statistical Interpolation (GSI). The ensemble information is obtained from the Global Ensemble Forecast System (GEFS). HWRF has the capability of assimilating hurricane inner core observations, such as the NOAA's P3 aircraft tail Doppler Radar (TDR). When TDR data is assimilated, First Guess at Appropriate Time (FGAT) is used to make sure that GSI uses innovations calculated by comparing observations with corresponding model analysis fields valid at the time when the observation was collected. The GFS forecasted fields every 6 hours are used to provide lateral boundary conditions during each forecast.

The analysis is also modified by removing the vortex present in the first guess fields and inserting a new vortex. Depending on the observed intensity of the storm and on the run being cold-started or cycled, the new vortex may derive from a bogus procedure, from the 6-h forecast of the HWRF model initialized 6-h previously, or from HDAS. In any case, the vortex is modified so that the initial storm position, structure, and intensity conform to the NHC storm message.

The time integration is performed with a forward-backward scheme for fast waves, an implicit scheme for vertically propagating sound waves and the Adams-Bashforth scheme for horizontal advection and for the Coriolis force. In the vertical, the hybrid pressure-sigma coordinate (Arakawa and Lamb 1977) is used. Horizontal diffusion in based on a 2nd order Smagorinsky-type following (Janjic 1990).

The Community HWRF model can be used for the following basins: North Atlantic, Eastern North Pacific, Central Pacific, West Pacific, and Indian (including Arabian Sea and Bay of Bengal). In the North Atlantic and Eastern North Pacific basins, for which NHC is responsible, 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 WRF 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. In the other basins, HWRF is configured to run with its atmospheric component only.

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 (SST), 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. Additionally, an idealized tropical cyclone simulation capability has been added to HWRF V3.5a. The HWRF system support to the community by the Developmental Testbed Center (DTC) includes the following three main modules.

- HWRF atmospheric components
  - WRF-NMM (which has tropical physics schemes and a vortex- following moving nest)
  - WRF Preprocessing System (WPS)
  - Vortex initialization
  - Gridpoint Statistical Interpolation (GSI)
  - Unified Post-Processor (UPP)
  - 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

All HWRF components are under the Subversion revision control system. The code repositories are hosted and maintained as community codes at the National Center for Atmospheric Research (NCAR), except for GSI, which is housed at the National Oceanic and Atmospheric Administration (NOAA). A HWRF code management protocol has been established for proposing HWRF-related modifications to the software, whether the modifications are simply updates to the current features, bug fixes, or the addition of new features. HWRF code development must be carried out in the branches of the repositories and frequently synchronized with the trunks. Proposed software modifications must be thoroughly tested prior to being committed to the code repository to protect the integrity of the evolving code base.

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. Please also contact this email if you would like more information on the protocols for HWRF development.

1.3 HWRF Source Code Directory Structure

The HWRF system source code has the following eight components:

- WRF Atmospheric Model
- WPS
- 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).

- *hwrfv3.5a_utilities.tar.gz*
- *hwrfv3.5a_pomtc.tar.gz*
- *hwrfv3.5a_gfdl_vortextracker.tar.gz*
- *hwrfv3.5a_ncep-coupler.tar.gz*
- *hwrfv3.5a_wrf.tar.gz*
- *hwrfv3.5a_wps.tar.gz*
- *hwrfv3.5a_upp.tar.gz*
- *hwrfv3.5a_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. Note that these are the directories after the code is compiled. Before compilation not all of these directories are present.

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;
| | | relevant WRF lookup tables from run subdirectory, some of which are
| | | modified for HWRF)
| | |____*scripts/* (scripts used to run the HWRF system)
| | | |____*funcs (shell functions used by the scripts)*
| | |____*tools/* (source code for tools to run the HWRF system)
| | | *Makefile* makefile for tools code
| | | |____*grbindex*
| | | |____*hwrf_data_flag*
| | | |____*hwrf_prep_hybrid*
| | | |____*hwrf_readtdrstmid*
| | | |____*hwrf_readtdrtime*
| | | |____*hwrf_wrfout_newtime*
| | | |____*wgrib*
| | |____*vortex_init/* (source code for the HWRF vortex initialization)
| | | *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 the HWRF system)

2. pomtc (POM-TC Ocean model)

| |____*arch/* (compile options)
| |____*clean* script to clean created files and executables
| |____*compile* script to compile the POM-TC code
| |____*configure* script to create the configure.pom file for compile
| |____*fix* fixed data files needed to run POM-TC
| |____*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
| | |____*date2day*
| | |____*day2date*
| | |____*eastatl*
| | |____*eastpac*
| | |____*ext_eastat*
| | |____*getsst*
| | |____*gfdl_find_region*
| | |____*sharp_mcs_rf_l2m_rmy5*
| | |____*united*
| |____*ocean_main/* (source code for the ocean forecast model)
| | *Makefile* makefile for the ocean model code
| | |____*ocean_eastatl*
| | |____*ocean_eastatl_ext*
| | |____*ocean_eastpacl*
| | |____*ocean_united*
| |____*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)
| |____*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/* (Advanced Research WRF dynamic modules, not used by 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)
| |____*hydro/* (hydrology module, not used by HWRF)
| |____*inc/* (include files)
| |____*main/* (WRF main routines, such as wrf.F)
| |____*phys/* (physics modules)
| |____*run/* (run directory, not used by HWRF)
| |____*share/* (modules for WRF mediation layer and WRF I/O)
| |____*test/* (sub-dirs for specific configurations of WRF, such as idealized HWRF)
| |____*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, used in idealized simulations
| |____*metgrid/* (source code for metgrid.exe)

| |____*test_suite/* (WPS test cases)
| |____*ungrib/* (source code for ungrib.exe)
| |____*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 to control UPP 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
| |____*fix/* (fix files for GSI)
| |____*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 and executables.

1. *Tcvitals* (TC Vitals data)

syndat_tcvitals.${YYYY}

2. *abdecks* (a-deck and b-deck files)

a[al|ep]${MM}${YYYY}.dat A deck file.

b[al|ep]${MM}${YYYY}.dat B deck file.

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

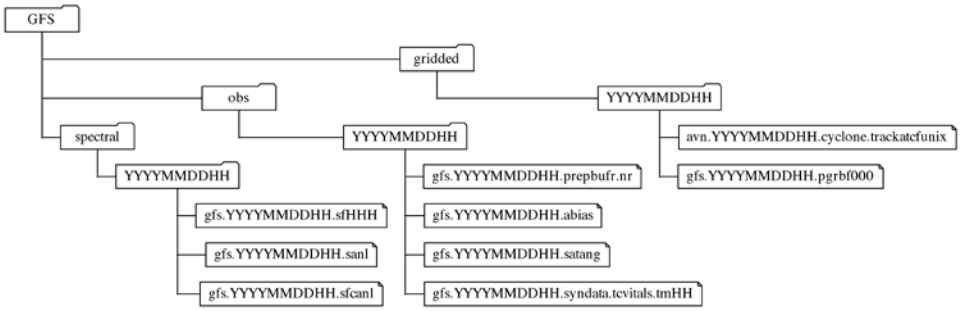
hwrf_gfdl_loop_current_wc_ring_rmy5.dat.${YYYYMMDD}* and *hwrf_gfdl_loop_current_rmy5.dat.${YYYYMMDD}, where ${YYYYMMDD} is the date. For example, *hwrf_gfdl_loop_current_rmy5.dat.20121028* and *hwrf_gfdl_loop_current_wc_ring_rmy5.dat.20121028* contain the loop current and warm-core ring information for October 28, 2012.

4. *GFS* (input data from GFS)

The dataset is arranged in three subdirectories

- *gridded (GFS gridded data, for atmosphere initialization)*
- *spectral (GFS spectral data, for ocean and atmosphere initialization)*
- *obs (observational data for GSI in prepBUFR and BUFR formats)*

Within these subdirectories there are further level of subdirectories based on the initial time *${YYYYMMDDHH}*, eg. 2012102806.



The *gridded* data directory contains two files

a. *avn.${YYYYMMDDHH}.cyclone.trackatcfunix*
b. *gfs.${YYYYMMDDHH}.pgrbf${hhh}*

Where *${YYYYMMDDHH}* is the initial time and *${hhh}* is the forecast hour. For example, *gfs.2012102806.pgrbf024* is a GFS 24h forecast in gridded GRIB format whose initial time is October 28 06Z 2012.

The *spectral* data directory contains at least three GFS spectral files

a. *gfs.${YYYYMMDDHH}.sf${hhh}*
   Forecast hours range from 0 to 126 in 3 hr increments.
b. *gfs.${YYYYMMDDHH}.sanl*
c. *gfs.${YYYYMMDDHH}.sfcanl*

The *obs* data directory contains at least five files

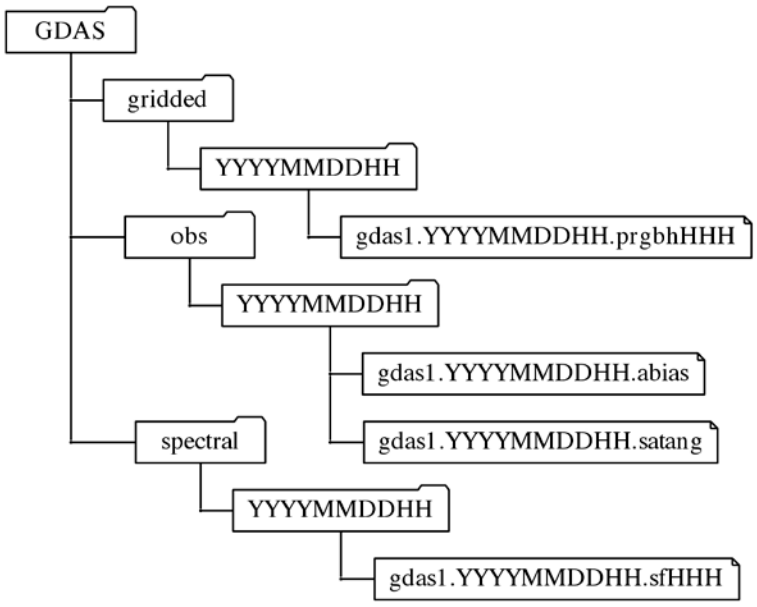
a. *gfs.${YYYYMMDDHH}.${SATELLITE}.tm${hhh}.bufr_d*
   Known satellites are 1bamua, 1bamub, 1bhrs3, 1bhrs4, 1bmhs, airsev and goesfv.
b. *gfs.${YYYYMMDDHH}.prepbufr.nr*
c. *gfs.${YYYYMMDDHH}.abias*
d. *gfs.${YYYYMMDDHH}.satang*
e. *gfs.${YYYYMMDDHH}.syndata.tcvitals.tm${hhh}*

5. *GDAS* (input data from GDAS)

The dataset is arranged in three subdirectories

- *gridded (GDAS gridded data, for atmosphere initialization)*
- *spectral (GDAS spectral data, for atmosphere initialization)*
- *obs (observational data for GSI in prepBUFR and BUFR formats)*

Within these subdirectories there are further level of subdirectories based on the initial time *${YYYYMMDDHH}*, eg. 2012102806.



The *gridded* data directory contains three files

a. *gdas1.${YYYYMMDDHH}.prgbh${hhh}*

Where *${YYYYMMDDHH}* is the initial time and *${hhh}* is the forecast hour. There should be three forecast hour files 3, 6 and 9.

The *obs* data directory contains two files

a. *gdas1.${YYYYMMDDHH}.abias*
b. *gdas1.${YYYYMMDDHH}.satang*

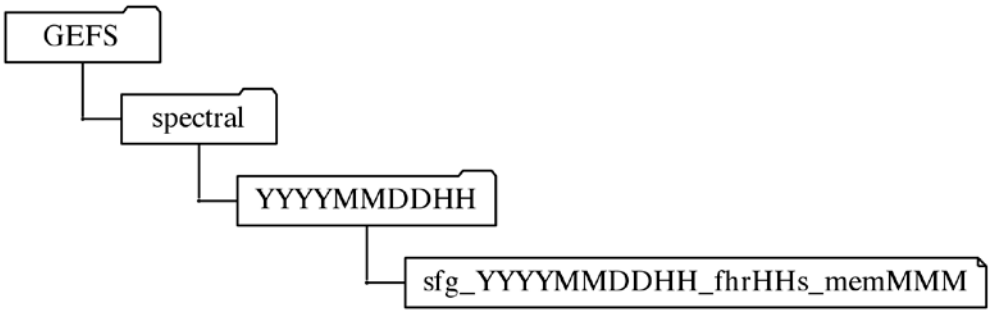
The *spectral* data directory contains at least three GFS spectral files

a. *gdas1.${YYYYMMDDHH}.sf${hhh}*
   Forecast hours are 3, 6 and 9.

6. *GEFS* (input data from GEFS)

The dataset is contains only one subdirectory

- *spectral (GEFS spectral data to provide ensemble information for data assimilation)*



The *spectral* data directory contains eighty files

a. *sfg_${YYYYMMDDHH}_fhr${hh}s_mem${mmm}*

   Where *${YYYYMMDDHH}* is the initial time, *${hh}* is the forecast hour and *${mmm}* is the ensemble member ID, which ranges from 1 to 80.

7. TDR (tail doppler radar)

The data is in a file in BUFR format contained within a directory structure based upon the storm name and observation time. For example a TDR file for the storm Sandy on October 28, 2012 would be in *TDR/SANDY/20121028/gdas1.2012102812.tldplr.tm00.bufr_d*.

8. *Fix* (time-independent files)

8.1 *upp* (for UPP)

8.2 *gsi* (for GSI)

CRTM_Coefficients

anavinfo_hwrf
atms_beamwidth.txt
bufrtab.012
global_ozinfo.txt
global_scaninfo.txt
hwrf_basinscale_satinfo.txt
hwrf_convinfo.txt
hwrf_hybens_d01_locinfo
nam_errtable.r3dv
nam_glb_berror.f77.gcv
nam_glb_berror.f77.gcv_Little_Endian
nam_global_ozinfo.txt
nam_global_pcpinfo.txt
nam_global_satangbias.txt
nam_regional_convinfo.txt
nam_regional_satinfo.txt
prepobs_errtable.hwrf
prepobs_prep.bufrtable

8.3 *ocean* (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

When using the scripts included in the released tar files to run the HWRF system, the following production directories will be created and populated:

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}* will have the following sub-directories:

messages (created by *hwrfdomain.ksh*)

geoprd (created by *geogrid.ksh*)

ungribprd (created by *ungrib.ksh*)

metgridprd (created by *metgrid.ksh*)

prep_hybrid (created by prep_hybrid.sh)

prep_hybrid_GFS (created by prep_hybrid.sh)

realprd (created by *real.ksh*)

realprd_GFS (created by real.ksh)

wrfghostprd (created by *wrf.ksh* run in "ghost" mode)

wrfanalysisprd (created by *wrf.ksh* run in "analysis" mode)

trkanalysisprd (created by *track_analysis.ksh*)

relocateprd (created by the vortex initialization scripts)

gsiprd (created by the GSI scripts, if GSI is run)

mergeprd (created by *merge.ksh*)

oceanprd (created by the ocean initialization scripts)

- *sharpn (created by the sharpening program, present only for the "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)*

wrfprd (created by *wrf.ksh* in "main" mode)

postprd (created by *run_unipost*)

gvtprd (created by *tracker.ksh*)

1.6 Scripts for Running HWRF

It is recommended that HWRF v3.5a be run using the shell scripts provided with the HWRF v3.5a release. The scripts are located in two directories. In *hwrf-utilities/scripts*, 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 otherwise instructed 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 the 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 MOAB/Torque, such as Jet, users can use the *qsub* command. For example, the command below requests a two-hour connection of 24 cores on the "sJet" nodes using the account "dtc-hurr".

qsub -X -I -l procs=24,walltime=2:00:00,partition=sjet -A dtc-hurr

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

Parallel code can also be submitted to the computation nodes using a batch system. For a platform that uses 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 platform that uses the MOAB/Torque batch system, the wrapper script *hwrf_wrapper* should be edited to contain the PBS options listed:

```
#PBS -A project                   # Project name
#PBS -l procs=202                 # number of total (MPI) tasks
#PBS -o stdout.txt                # Output filename
#PBS -e stder.txt                 # Error filename
#PBS -N hwrf                      # Job name
#PBS -l walltime=02:30:00         # Wallclock time
```

#PBS -q batch *# Queue name*

#PBS -d . *# Working directory of the job*

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

- On machines with LSF:
  bsub < hwrf_wrapper
- On machines with MOAB/Torque:
  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 HWRF, consists of eight components.

- WRF Atmospheric Model
- WPS
- UPP
- GSI
- HWRF Utilities
- 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.

- *hwrfv3.5a_hwrf-utilities.tar.gz*
- *hwrfv3.5a_pomtc.tar.gz*
- *hwrfv3.5a_gfdl-vortextracker.tar.gz*
- *hwrfv3.5a_ncep-coupler.tar.gz*
- *hwrfv3.5a_WRFV3.tar.gz*
- *hwrfv3.5a_WPSV3.tar.gz*
- *hwrfv3.5a_UPP.tar.gz*

- *hwrfv3.5a_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 "work" partition. Prior to unpacking the tar files you have just downloaded, create a single working directory in your workspace. 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 gunzip command,

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

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

Once unpacked, there should be the eight source directories.

- *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*.

The next section covers the system requirements 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 code 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 has been tested on a variety of computing platforms. Currently the HWRF system is actively supported on Linux computing platforms using both the Intel and PGI Fortran compilers. Unforeseen build issues may occur when using older compiler versions. Typically the best results come from using the most recent version of a compiler. The known issues section of the community website provides the complete list of compiler versions currently supported.

While the community HWRF build system provides legacy support for the IBM AIX platforms, the unavailability of AIX test platforms means all AIX support is cursory at best.

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.

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 Linux systems, HWRF supports both the Intel ifort and PGI pgf90 Fortran 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 links to this version of the libraries last.

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 necessary linear algebra library routines needed by the HWRF system.

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.

In addition, starting with the current version of HWRF, the user may choose to run WRF with either real or idealized input data. The idealized data case requires setting environment flags prior to compiling the code, which creates a unique executable that should only be run with the idealized data.

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.

For Linux architectures, there are currently 55 options. For the HWRF system, only the distributed memory (dmpar) builds are recommended. Therefore as an example, the acceptable PGI options are 3, 7, 11, or 15 (shown below).

3. Linux x86_64 i486 i586 i686, PGI compiler with gcc (dmpar)
7. Linux x86_64 i486 i586 i686 PGI compiler with pgcc YELLOWSTONE (dmpar)
11. Linux x86_64, PGI compiler with pgcc, SGI MPT (dmpar)
15. Linux x86_64, PGI accelerator compiler with gcc (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 and the aggressiveness of the optimization. . 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 20 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 that 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 User's 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

Should you experience difficulty building WRF while using the PGI compiler, a helpful guide for building WRF with PGI compilers on a 32-bit or 64-bit LINUX system can be found at:

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

2.6.3 Compiling the Idealized Tropical Cyclone WRF-NMM

The HWRF idealized tropical cyclone WRF-NMM component requires different executables than for the real case. The following section will describe how to build the executables for the idealized case.

Building the idealized component requires a slightly different configuration than for the standard WRF build outlined in section 2.6.2. If a user has already built the standard WRFV3 and created ***real_nmm.exe*** and ***wrf.exe*** and now wants to build WRFV3 for idealized tropical cyclone simulations, they first need to completely clean the previous build. This is done by running a

./clean -a

that removes **ALL** build files, including the executables, libraries, and the *configure.hwrf*. To correctly configure WRF-NMM for the HWRF idealized tropical cyclone simulation, requires setting the additional environment variable *IDEAL_NMM_TC*. The total list of required environment variables for creating the idealized tropical cyclone component is:

In C-Shell use the commands:

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

In bash or ksh shells, use the commands:

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

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.

For Linux architectures, there are currently 51 options. For the HWRF system, only the distributed memory (dmpar) builds are recommended. Therefore as an example, the acceptable PGI options are 3, 7, 11, or 15 (shown below).

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

Once the configure step is complete, the code is compiled by including the target "nmm_tropical_cyclone" to the compile command.

```
./compile nmm_tropical_cyclone
```

A successful compilation produces two executables in the directory *main.*

ideal.exe: WRF initialization
wrf.exe: WRF model integration

Note that the minimum a user needs for the idealized capability is to compile WPS and WRF. If wanted, UPP may also be used. The components POM-TC, GSI and hwrf-utilities are not used in HWRF idealized tropical cyclone simulations.

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 Linux, seven 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/MKL, IBM POE (dmpar)*
6. *Linux x86_64, Intel compiler w/LAPACK (dmpar)*
7. *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, 4, or 5 if your compiler includes the MKL libraries, and option 6 or 7 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 that 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 24 executables in the directory exec/.

- *diffwrf_3dvar.exe**
- *grbindex.exe**
- *hwrf_anl_4x_step2.exe**
- *hwrf_anl_bogus_10m.exe**
- *hwrf_anl_cs_10m.exe**
- *hwrf_bin_io.exe*
- *hwrf_create_nest_1x_10m.exe**
- *hwrf_create_trak_fnl.exe**
- *hwrf_create_trak_guess.exe**
- *hwrf_data_flag.exe**
- *hwrf_inter_2to1.exe**
- *hwrf_inter_2to2.exe**
- *hwrf_inter_2to6.exe**
- *hwrf_inter_4to2.exe**
- *hwrf_inter_4to6.exe**
- *hwrf_merge_nest_4x_step12_3n.exe**
- *hwrf_pert_ct1.exe**
- *hwrf_prep.exe**
- *hwrf_readtdrstmid.exe*
- *hwrf_readtdrtime.exe*
- *hwrf_split1.exe**
- *hwrf_swcorner_dynamic.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)*
5. *Linux x86_64, Intel compiler, IBM POE (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 remove all the object files, type:

./clean

To conduct a complete clean that 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/*.

- *gfdl_date2day.exe*
- *gfdl_day2date.exe*
- *gfdl_find_region.exe*
- *gfdl_getsst.exe*
- *gfdl_ocean_eastatl.exe*
- *gfdl_ocean_eastpac.exe*
- *gfdl_ocean_ext_eastatl.exe*
- *gfdl_ocean_united.exe*
- *gfdl_sharp_mcs_rf_l2m_rmy5.exe*
- *hwrf_ocean_eastatl.exe*
- *hwrf_ocean_eastatl_ext.exe*
- *hwrf_ocean_eastpac.exe*
- *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_PATH ${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 *gfdl-vortextracker* 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 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)*
6. *Linux x86_64, Intel compiler, IBM POE (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/*.

1. hwrf_gettrk.exe
2. hwrf_tave.exe
3. 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 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)*
5. *Linux x86_64, Intel compiler, IBM POE (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 ${SCRATCH}/HWRF/WRFV3/

For bash/ksh, use:

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

In order to run the WRF Domain Wizard (http://esrl.noaa.gov/gsd/wrfportal/), 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 Linux computers, there are 38 listed options. The first 28 are the most relevant to HWRF. Select if you want GRIB 2 support, or if you don't.

1. *Linux x86_64, gfortran (serial)*
2. *Linux x86_64, gfortran (serial_NO_GRIB2)*
3. *Linux x86_64, gfortran (dmpar)*
4. *Linux x86_64, gfortran (dmpar_NO_GRIB2)*

5. *Linux x86_64, PGI compiler (serial)*
6. *Linux x86_64, PGI compiler (serial_NO_GRIB2)*
7. *Linux x86_64, PGI compiler (dmpar)*
8. *Linux x86_64, PGI compiler (dmpar_NO_GRIB2)*
9. *Linux x86_64, PGI compiler, SGI MPT (serial)*
10. *Linux x86_64, PGI compiler, SGI MPT (serial_NO_GRIB2)*
11. *Linux x86_64, PGI compiler, SGI MPT (dmpar)*
12. *Linux x86_64, PGI compiler, SGI MPT (dmpar_NO_GRIB2)*
13. *Linux x86_64, IA64 and Opteron (serial)*
14. *Linux x86_64, IA64 and Opteron (serial_NO_GRIB2)*
15. *Linux x86_64, IA64 and Opteron (dmpar)*
16. *Linux x86_64, IA64 and Opteron (dmpar_NO_GRIB2)*
17. *Linux x86_64, Intel compiler (serial)*
18. *Linux x86_64, Intel compiler (serial_NO_GRIB2)*
19. *Linux x86_64, Intel compiler (dmpar)*
20. *Linux x86_64, Intel compiler (dmpar_NO_GRIB2)*
21. *Linux x86_64, Intel compiler, SGI MPT (serial)*
22. *Linux x86_64, Intel compiler, SGI MPT (serial_NO_GRIB2)*
23. *Linux x86_64, Intel compiler, SGI MPT (dmpar)*
24. *Linux x86_64, Intel compiler, SGI MPT (dmpar_NO_GRIB2)*
25. *Linux x86_64, Intel compiler, IBM POE (serial)*
26. *Linux x86_64, Intel compiler, IBM POE (serial_NO_GRIB2)*
27. *Linux x86_64, Intel compiler, IBM POE (dmpar)*
28. *Linux x86_64, Intel compiler, IBM POE (dmpar_NO_GRIB2)*

Select the appropriate “dmpar” option for your choice of compiler.

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 that 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 successful compilation of WPS produces the three symbolic links: *geogrid.exe*, *ungrib.exe*, and *metgrid.exe* in the main directory, and a single symbolic link *mod_levs.exe*, used for the idealized tropical cyclone configuration, in the *util* directory. 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 ${SCRATCH}/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 the LINUX operating systems, there are 12 options. Select the appropriate “dmpar” option compatible with your system.

1. *Linux x86_64, PGI compiler (serial)*
2. *Linux x86_64, PGI compiler (dmpar)*
3. *Linux x86_64, PGI compiler, SGI MPT (serial)*
4. *Linux x86_64, PGI compiler, SGI MPT (dmpar)*
5. *Linux x86_64, Intel compiler (serial)*
6. *Linux x86_64, Intel compiler (dmpar)*
7. *Linux x86_64, Intel compiler, SGI MPT (serial)*
8. *Linux x86_64, Intel compiler, SGI MPT (dmpar)*
9. *Linux x86_64, Intel compiler, IBM POE (serial)*
10. *Linux x86_64, Intel compiler, IBM POE (dmpar)*
11. *Linux x86_64, gfortran compiler (serial)*
12. *Linux x86_64, gfortran compiler (dmpar)*

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. 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 compiler (ifort & icc), IBM POE (dmpar,optimize)*
8. *Linux x86_64, Intel/gnu compiler (ifort & gcc) (dmpar,optimize)*

Select the appropriate “dmpar” option for your platform and compiler. For a generic Linux machine, choose option (1) or (3) for a PGI build, or option (5) or (8) for an Intel build. On Jet, select option (4) for an Intel build. For a SGI Linux installation, select (2) for PGI or (6) for Intel. For an IBM Linux installation with an Intel compiler, select option (7).

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 that 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. Other tools, such as *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

In HWRF, GDAS forecasts of soil temperature and moisture in GRIB format are processed through the WPS and real_nmm programs to provide lower boundary conditions. Preliminary Initial Conditions (ICs) for the atmosphere are obtained by processing the GDAS spectral forecasts in binary format through a utility called prep_hybrid, followed by real_nmm.

When inner core data will not assimilated (Figure 3.1), the GDAS 6-h forecast valid at the time of HWRF initialization is processed. If the GDAS 6-h forecast is not available, the GFS 0-h forecast can be used but in this case no data assimilation is performed. For information on running HWRF with GFS initialization please contact the WRF help desk at wrfhelp@ucar.edu.

When inner core data are assimilated, background fields are needed at three time levels. These are the GDAS 3-, 6-, and 9-h forecasts, valid at 3-h before the HWRF initialization, at the time of HWRF initialization, and 3-h after HWRF initialization, respectively. This is required because the inner core observations are taken in a time window centered on the HWRF initialization. Throughout the HWRF scripts, array variable *FGAT* is used to define the number of hours, counting from the HWRF initialization time, for which preprocessing needs to be conducted. For example, in the operational configuration, *FGAT=-3, 0, 3*.

Most of this chapter focuses on the procedures to preprocess the global data for the case in which inner core data assimilation is not employed. Information on preprocessing data for use with inner core data assimilation is in Section 3.6. Note that, even when inner core data will not be assimilated, the data preprocessing scripts will be called for the three time levels. However, global model data will actually be preprocessed only for the HWRF initialization time.

The preliminary ICs are used to initialize two 90-second, uncoupled WRF forecasts. These runs, termed “ghost” and “analysis”, are used to enhance the HWRF ICs with data assimilation and vortex relocation procedures, which are discussed in Chapter 4.

In addition to preprocessing data for HWRF initialization, the prep_hybrid utility is used to prepare data for lateral boundary conditions (LBCs). The LBCs used in the HWRF forecast are obtained from the GFS forecast initialized at the same time as HWRF. Conversely, the LBCs used in the ghost and analysis runs are obtained from the GFS forecast initialized 6 h before the HWRF initialization. In both cases, the GFS is processed using prep_hybrid to create input to real_nmm.

This Chapter explains how to run WPS and prep_hybrid for HWRF to create the preliminary initial conditions. Prep_hybrid is a HWRF-specific tool that is distributed with the HWRF Utilities. The WPS is a set of three programs whose collective role is to prepare input to the real_nmm program for real data simulations or to the ideal program for idealized tropical cyclone simulations. For general information about working with WPS, see the WRF-NMM documentation at

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

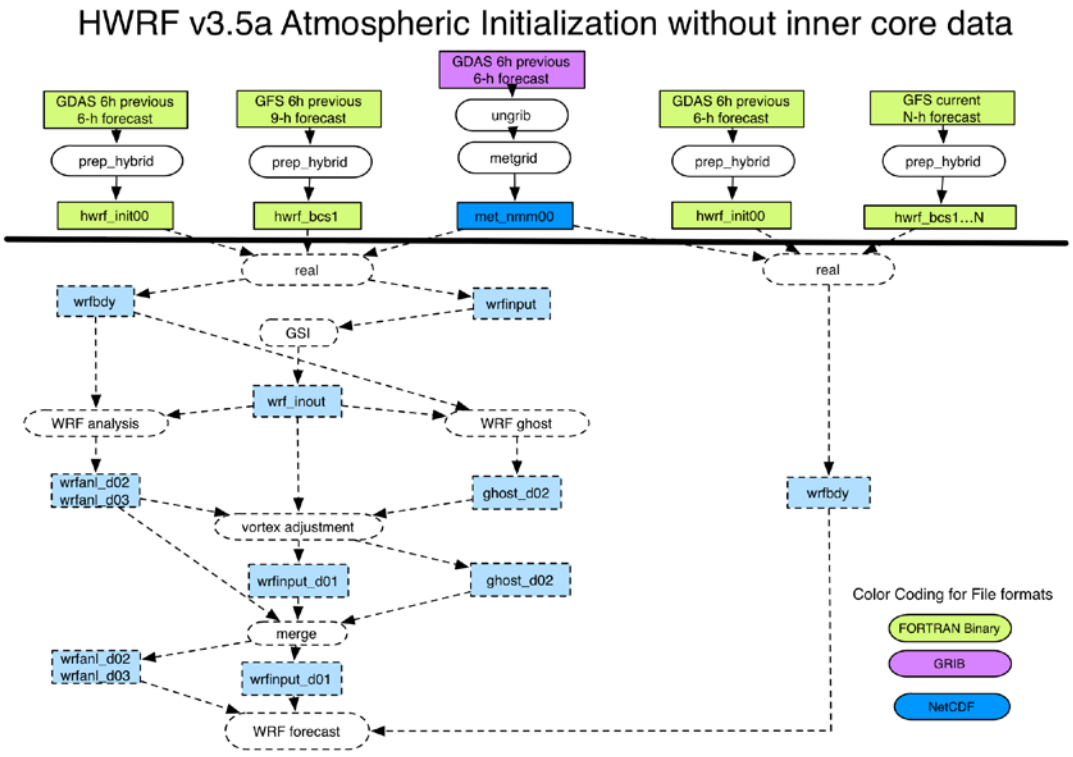


Figure 3.1 Diagram of the HWRF preprocessing procedures when hurricane inner-core data assimilation is not conducted. The processes shown above the thick horizontal line are described in this chapter.

3.2 How to Run the HWRF Preprocessing Using Scripts

Five wrapper scripts are used to preprocess data for HWRF: *hwrfdomain_wrapper, geogrid_wrapper, prep_hybrid_wrapper, ungrib_wrapper and metgrid_wrapper*. These wrapper scripts drive the five corresponding low-level scripts, *hwrfdomain.ksh, geogrid.ksh, prep_hybrid.ksh, ungrib.ksh and metgrid.ksh*, respectively. The script *hwrfdomain.ksh* defines the location of the parent domain; *geogrid.ksh* interpolates static geographical data to the three HWRF domains; *prep_hybrid.ksh* pre-process the GDAS or GFS spectral data on sigma vertical levels for input to real_nmm; *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 *${SCRATCH}/HWRF/hwrf-utilities/wrapper_scripts /global_vars.ksh* and make sure that the list of global variables is correctly customized for the desired HWRF run. In particular, make sure that variables *HWRF_DATA_DIR, HWRF_SRC_DIR*, and *HWRF_OUTPUT_DIR* point to the customized location of the input datasets, the HWRF source code, and the HWRF production directories, respectively.

String *PATH_TO_GLOBAL* can be found in each of the wrapper scripts. This string needs to be substituted with the actual path to file *global_vars*.ksh in the user's machine. To do that, it is possible to manually edit each wrapper and substitute *PATH_TO_GLOBAL* with *${SCRATCH}/HWRF/hwrf-utilities/wrapper_scripts*, where *${SCRATCH}* should be expanded. To facilitate this process, instead of doing the manual substitution, users can change to directory *${SCRATCH}/HWRF/hwrf-utilities/wrapper_scripts* and type

make.

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.

A diagram of the HWRF atmospheric initialization procedures when the inner-core data assimilation is not carried out is shown in Figure 3.1. The objects drawn with solid lines show the procedures covered in this chapter. Those drawn with dashed lines will be described in Chapter 4. For information on the HWRF preprocessing procedures when HWRF inner-core data assimilation is conducted, please see Section 3.6.

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
RUN_GSI

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 if all the environment variables are set.
2. Create working directory *${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*.
9. Check if GSI will run. GSI will run if the variable *RUN_GSI* is set to *T* in *global_vars.ksh*.

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

storm.center: file that contains the storm center latitude and longitude.

domain.center: file that contains the domain reference center latitude and longitude.

tcvital: file that contains the storm's TCVital.

tcvital.as: file that contains the storm's TCVitals.

go_gsi or *no_gsi*: file that indicates whether GSI will run (*go_gsi*) or not (*no_gsi*).

Status Check

If the five 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
FCST_INTERVAL
GEOG_DATA_PATH
HWRF_UTILITIES_ROOT
ATMOS_DOMAINS
IO_FMT
START_TIME
SID
MPIRUN

First use the *qsub* 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 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}/geoprd*

geo_nmm.d01.nc: static geographical data for the parent domain, with grid spacing of 0.18 degrees.

geo_nmm_nest.l01.nc: static geographical data that covers the parent domain, with grid spacing of 0.06 degrees.

geo_nmm_nest.l02.nc: static geographical data that covers the parent domain, with 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 *prep_hybrid_wrapper*

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

HWRF_SCRIPTS
BKG_MODE
RUN_PREP_HYB
HWRF_UTILITIES_ROOT
DOMAIN_DATA
BC_TIME
START_TIME
GFS_SPECTRAL_DIR
IO_FMT
VERT_LEV
FHR_BKG

For the operational HWRF configuration, GDAS is used to generate ICs for the atmosphere, while GFS is used to generate LBCs. The array variable *FHR_BKG* contains the times, specified in number of hours after the HWRF initialization, for which *prep_hybrid* is run. Note that, in wrapper *prep_hybrid_wrapper*, this variable is used twice. The first time it appears, it refers to the processing of the GDAS and GFS forecasts initialized 6-h before the current HWRF initialization. In this instance, *FHR_BKG=0, 3* because initial and boundary conditions are only generated up until 3 h for use in the 90-s ghost and analysis runs. The second time variable *FHR_BKG* appears, it is set to 0, 6, …., 126 to create LBCs for the entire forecast length of the HWRF run. This is passed to the low-level script *hwrf-utilities/scripts/prep_hybrid.ksh* as *BC_TIME*.

In the operational HWRF configuration, variable *BKG_MODE* is set to *GDAS* in *global_vars.ksh* to indicate that GDAS is used as the source of ICs. As stressed in Section 3.1, users that do not have access to GDAS should contact the WRF help desk to obtain information about using

alternate *BKG_MODE*. Note that in wrapper *prep_hybrid_wrapper*, variable *BKG_MODE* is overwritten to *GFS* because even when using GDAS for ICs, the boundary conditions are obtained from GFS.

Then run the wrapper script in *hwrf-utilities/wrapper_scripts* using the command:

prep_hybrid_wrapper, which, in turn, will run the low-level script

hwrf-utilities/scripts/prep_hybrid.ksh.

Overview of script *prep_hybrid.ksh*

1. Initialize the function library and check to see if all the environment variables are set and the *hwrf_prep.exe* executable exists.
2. Create working directories. For the operational HWRF configuration, those are *${DOMAIN_DATA}/prep_hybrid/${YYYYMMDDHH}* (for the ICs) and *{DOMAIN_DATA}/prep_hybrid_GFS/* (for the LBCs).
3. Make sure there is a large enough stack.
4. Copy the input.
5. Run *hwrf_prep.exe*.
6. Link the output.

Output files in directory *${DOMAIN_DATA}/prep_hybrid/${YYYYMMDDHH}*

hwrfinit_00: GDAS spectral data pre-processed by *hwrf_prep.exe* and ready to be used by *real_nmm* to generate HWRF ICs.

hwrfbcs00_${bc_index}: GFS spectral data pre-processed by *hwrf_prep.exe* and ready to be used by *real_nmm* to generate HWRF LBCs. The variable *bc_index=0,1,..,21* corresponds to the 21 GFS forecast lead times that need to be pre-processed to create the LBCs for the 126-h HWRF forecast.

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

hwrfinit_00: GFS spectral data pre-processed by *hwrf_prep.exe* and ready to be used by *real_nmm* to generate HWRF ICs, only to be used when running a non-operational configuration (*BKG_MODE=GFS).*

hwrfbcs00_${bc_index}: GFS spectral data pre-processed by *hwrf_prep.exe* and ready to be used by *real_nmm* to generate HWRF LBCs.

Status check

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

3.2.4 *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
FCST_INTERVAL
RUN_PREP_HYB
BKG_MODE
GFS_DATA_MODE
GDAS_GRIDDED_DIR
INNER_CORE_DA

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 the work directory *${DOMAIN_DATA}/ungribprd/${YYYYMMDDHH}*.
3. Create the namelist used by *ungrib.exe*.
4. Copy the ungrib table.
5. Link the GRIB files.
6. Run *ungrib.exe*.

Output files in directory *${DOMAIN_DATA}/ungribprd/${YYYYMMDDHH}*

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, *ungrib.log*, the wrapper script *ungrib_wrapper* and the low-level script *ungrib.ksh* have successfully finished.

3.2.5 *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
HWRF_UTILITIES_ROOT
DOMAIN_DATA
START_TIME
FCST_LENGTH

FCST_INTERVAL
IO_FMT
ATMOS_DOMAINS
MPIRUN
RUN_PREP_HYB
METGRID_CORES

Next use the *qsub* 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 work directory *${DOMAIN_DATA}/metgridprd/${YYYYMMDDHH}*
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 *${DOMAIN_DATA}/metgridprd/${YYYYMMDDHH}*

met_nmm.d01.YYYY-MM-DD_HH:mm:ss.nc. Here, *YYYY- MM-DD_HH:mm:ss* refers to the valid 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.

3.3 Executables

3.3.1 *geogrid.exe*

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

INPUT:
Files in geographical static data directory *${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 valid date of the interpolated data in each file.

USAGE:
${WPS_ROOT}/metgrid.exe

3.3.4 *hwrf_prep.exe*

FUNCTION:
Pre-process the GDAS or GFS spectral data on vertical sigma levels in binary format for use the by real_nmm or ideal programs

INPUT:
geogrid.out: link to *geo_nmm.d01.nc*
prep_hybrid.nl: prep_hybrid namelist
fort.11: link to *gfsbc${bc_index}*
fort.44: link to *itime* file contains *bc_index*
fort.45: link to *domain.center*
gfsbc${bc_index}: link to the global spectral file
[*gdas1|gfs].${BKG_START_TIME}.sf${BKG_FCST_TIME}*
Where *$BKG_START_TIME* is the GDAS or GFS initialization time and *$BKG_FCST_TIME* is time the GDAS or GFS forecast lead time. For example to create the 3-h LBCs for the ghost run associated with the HWRF forecast initialized at 2012102806, the GFS initialized 6-h previously is used, and these variables would be set to:

- *BKG_START_TIME = 2012102800*
- *BKG_FCST_TIME = 009*

OUTPUT:
hwrfinit_00: GDAS or GFS spectral data pre-processed by *hwrf_prep.exe* and ready to be used by *real_nmm* to generate HWRF initial conditions.

hwrfbcs00_${bc_index}: GDAS or GFS spectral data pre-processed by *hwrf_prep.exe* and ready to be used by *real_nmm* to generate HWRF lateral boundary conditions.

USAGE:
${PREP_EXE} $NX1 $NY1 $VERT_LEV $DXX $DYY
where *$NX1 $NY1 $VERT_LEV* are the grid dimensions in the meridional, zonal and vertical directions, and *$DXX $DYY* are the horizontal grid spacings.

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.
CENTLO = (STORM_LON + LONF72)/ 2
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 *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 geogrid. For more information about the WRF Domain Wizard, see

http://esrl.noaa.gov/gsd/wrfportal/DomainWizard.html

3.6 Inner-Core Data Assimilation

In order to perform inner-core data assimilation, four prerequisites need to be met.

1. The following environment variables must be set.
   a. RUN_GSI = T
   b. INNER_CORE_DA =1
   c. BKG_MODE = GDAS
2. The forecast must be cycled (no inner core data assimilation on cold starts).
3. The global model for preliminary ICs must be GDAS (*BKG_MODE=GDAS).*
4. A TDR BUFR file must exist for the time of HWRF initialization.

If these prerequisites are met, the preprocessing stages have to be run for two additional times centered on the HWRF initialization time (Figure 3.2) to provide FGAT data for GSI as described

in Section 4.1. These two extra preprocessing steps are windowed around the HWRF initialization time by ±3 hours. The data is produced in directories named by using the valid time for the preprocessed data. For example, if the HWRF initialization time is 2012102812, preprocessing needs to be run for 2012102809, 2012102812 and 2012102815. In this case, the ungrib output will be written to

- *${DOMAIN_DATA}/ungribprd/2012102809*
- *${DOMAIN_DATA}/ungribprd/2012102812*
- *${DOMAIN_DATA}/ungribprd/2012102815*

Likewise, the metgrid output will be written to

- *${DOMAIN_DATA}/metgridprd/2012102809*
- *${DOMAIN_DATA}/metgridprd/2012102812*
- *${DOMAIN_DATA}/metgridprd/2012102815*

And the prep_hybrid output will be produced in

- *${DOMAIN_DATA}/prep_hybrid/2012102809*
- *${DOMAIN_DATA}/prep_hybrid/2012102812*
- *${DOMAIN_DATA}/prep_hybrid/2012102815*
- *${DOMAIN_DATA}/prep_hybrid_GFS*

Note that the procedures at ±3 hours are simplified because only ICs are processed at those times. The LBCs for the HWRF forecast are only generated once, using the GFS initialized at the same time as the HWRF.

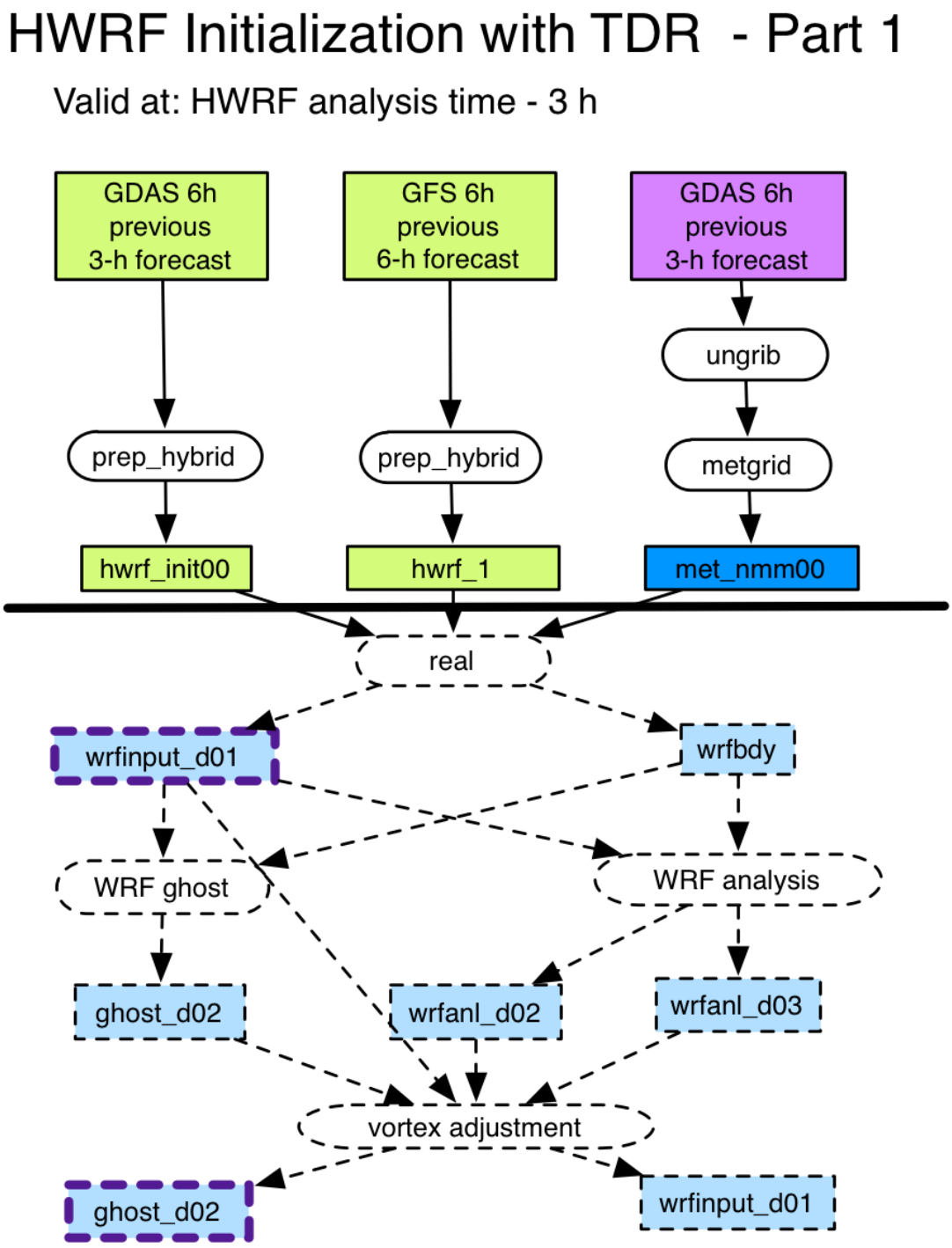


Figure 3.2- part 1. Diagram of the HWRF preprocessing procedures when hurricane inner-core data assimilation is conducted. The processes shown above the thick horizontal line are described in this chapter. Part 1 refers to the processes at valid time 3 h before the HWRF initialization.

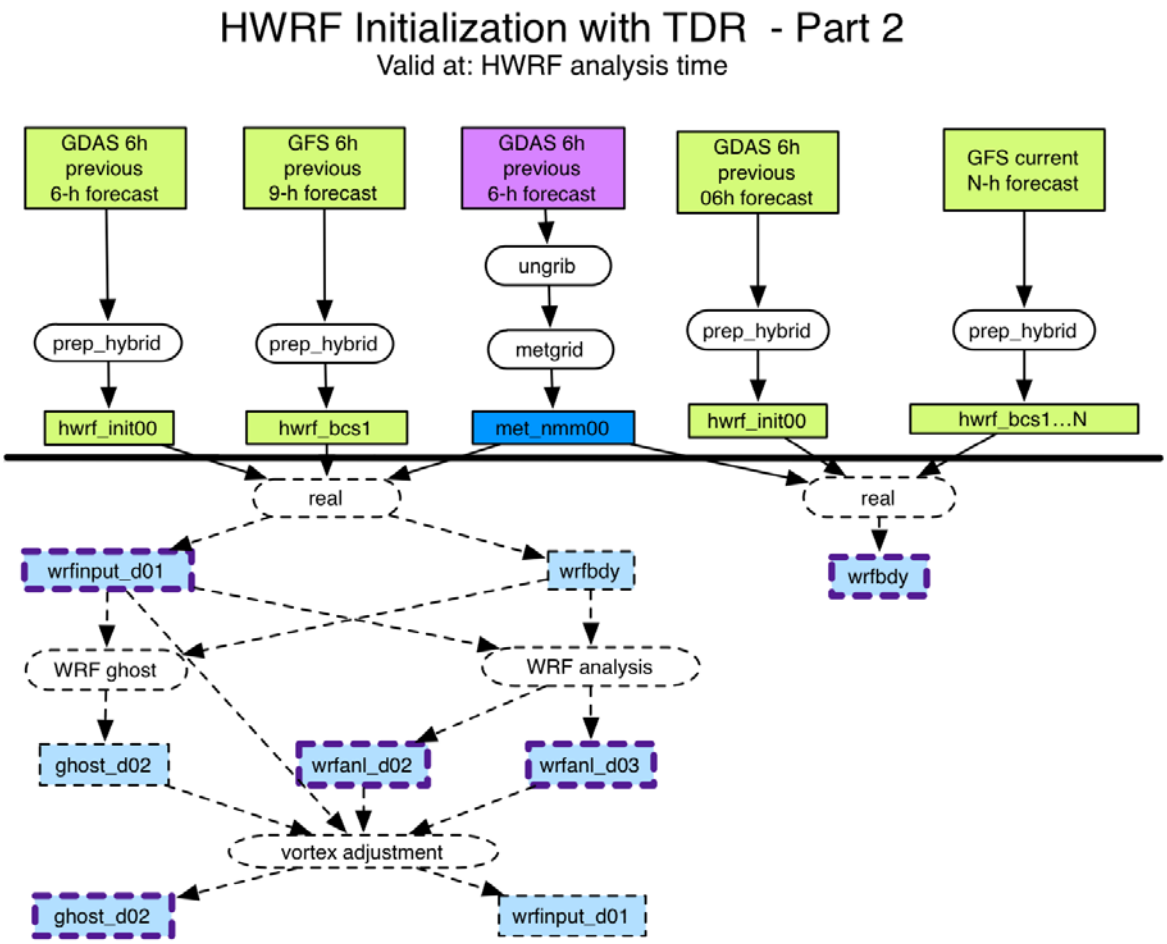


Figure 3.2- part 2. Same as Fig 3.2 - part 1, except for processes valid at the time of HWRF initialization.

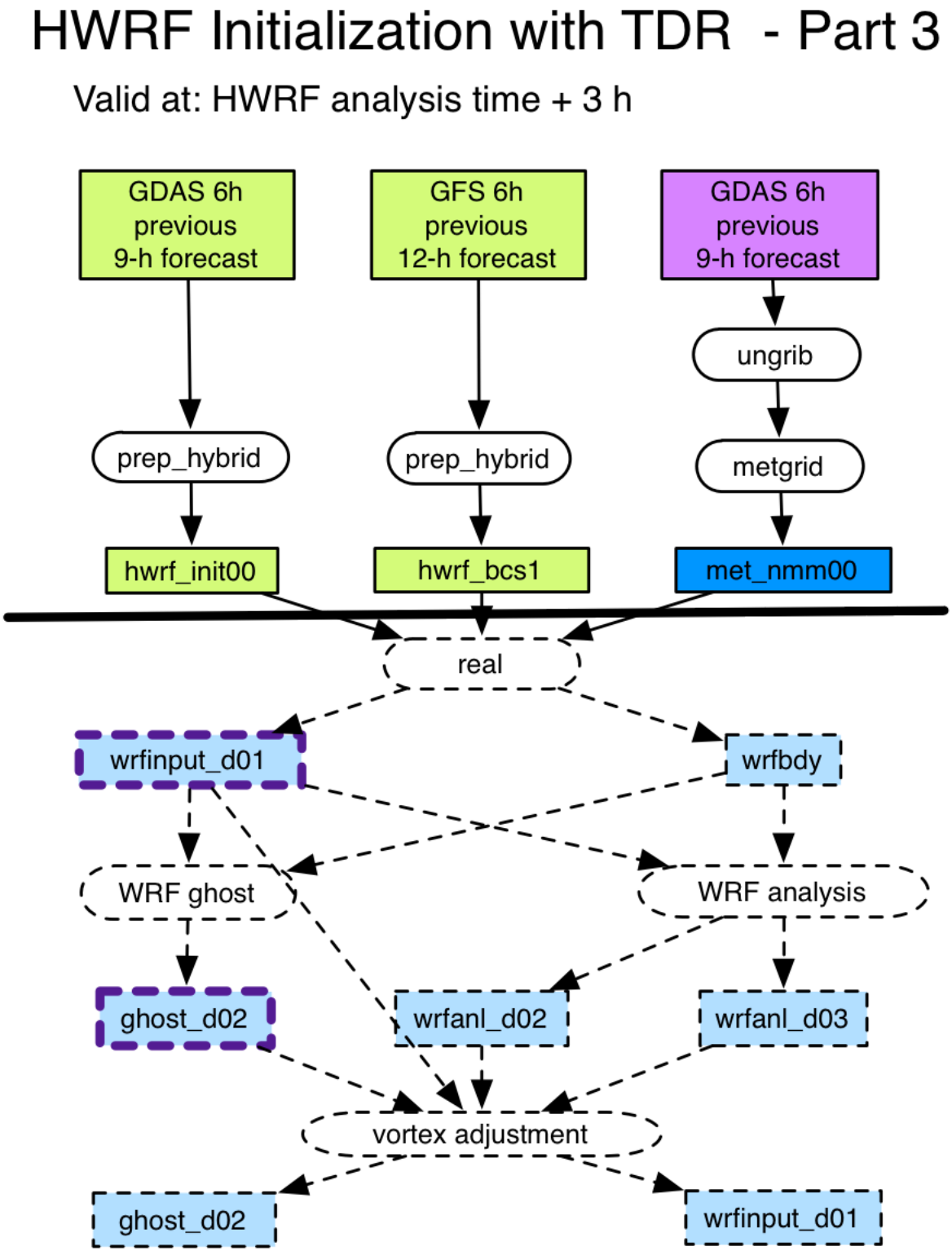


Figure 3.2- part 3. Same as Fig 3.2 - part 1, except for processes at valid time 3 h after the HWRF initialization.

Chapter 4: HWRF Atmospheric Initialization

4.1 Overview

HWRF's atmospheric component, the WRF-NMM, needs ICs and LBCs to produce forecasts. The GDAS forecasts are used to create the preliminary atmospheric fields, which are further improved through the GSI data assimilation system and the vortex adjustment procedures to provide the final IC. Most of this chapter focuses on the procedures to initialize HWRF for the case in which inner core data assimilation is not employed (Figure 4.1). Information on initializing HWRF for use with inner core data assimilation is in Section 4.5.

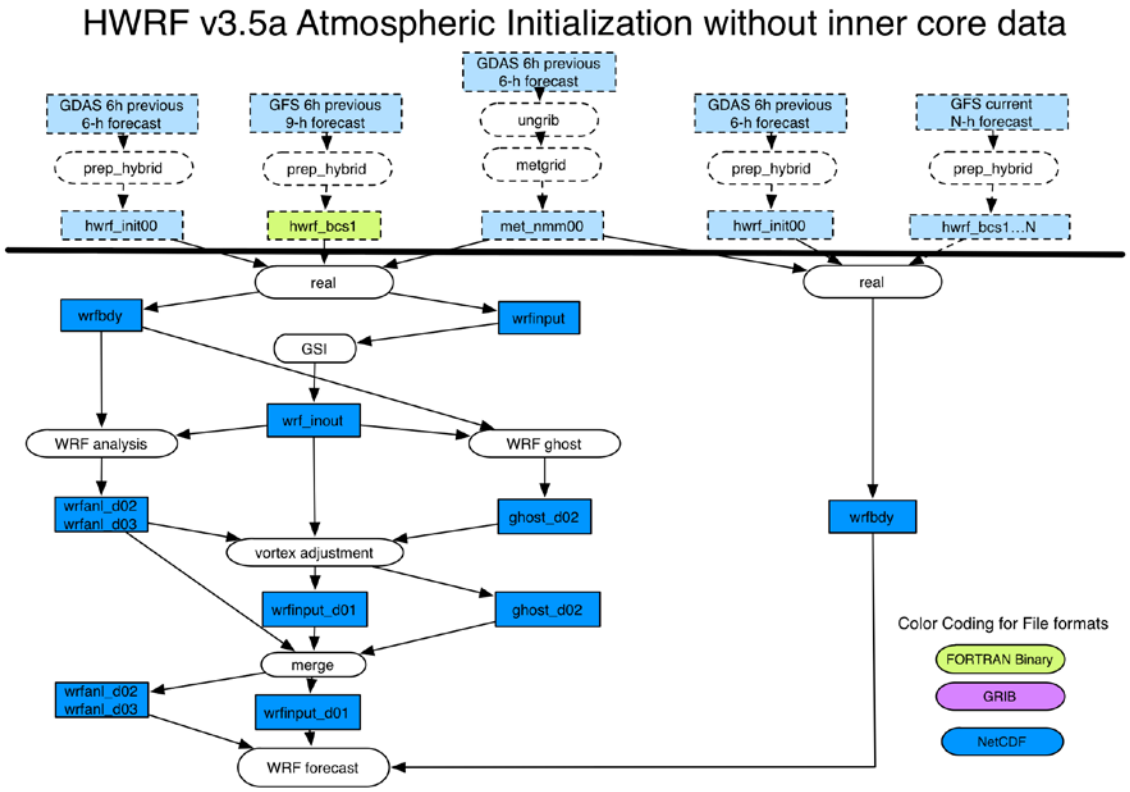


Figure 4.1 Diagram of the HWRF atmospheric initialization procedures without inner core data assimilation. Processes shown below the thick horizontal line are described in this chapter.

The vortex adjustment procedures are necessary because the initial vortex is often not realistically represented in the preliminary ICs since it originates from a low-resolution global data source, such as GDAS. Therefore, HWRF employs a sophisticated algorithm to adjust the vortex to match the current time's observed storm intensity, location and structure.

Program *real_nmm* is used to create the preliminary d01 ICs and LBCs from the output of *prep_hybrid* and WPS' *metgrid* program. The ICs created by *real_nmm* are a first guess for the HWRF outer domain (d01). They are further modified by using GSI to assimilate observational data, creating the d01 input to the vortex relocation procedure. The term HWRF Data Assimilation System (HDAS) refers to the process of running GSI for data assimilation in

HWRF. The fields obtained by running GSI on the HWRF parent domain are typically called HDAS fields.

The data assimilation in HWRF is performed using the hybrid ensemble-variational method. This indicates that the background error covariance information is a combination of two sources: a static, pre-generated matrix for the global model, and a flow-dependent matrix derived from the Global Ensemble Forecast System (GEFS) 6-h forecasts. Because the HWRF uses the GEFS forecast but does not feedback onto the GEFS, this procedure is termed "one-way hybrid". For more information on the ensemble-variational method, refer to the HWRF v3.5a Scientific Documentation available from the DTC website (www.dtcenter.org/HurrWRF/users). In the operational HWRF, for the cases in which inner core observations are not assimilated, GSI is run only in the parent domain. If desired, data assimilation can be completely turned off by setting *RUN_GSI=F* in the global variable file *global_vars.ksh*.

Initial conditions for HWRF d02 and d03 are created by ingesting HDAS fields onto the HWRF vortex initialization procedure. To prepare the fields for input in the vortex initialization, two 90-s atmosphere-only forecasts are conducted. These runs are referred to as the WRF analysis and WRF ghost runs, and their domains are detailed in Section 4.2.

The analysis runs use the same domain configuration as the HWRF forecast runs. The differences between the two are the forecast length (analysis only runs for 90 s), coupling (analysis is atmosphere-only) and initialization (analysis is initialized from HDAS). The WRF ghost run uses the same parent domain as the HWRF forecast runs, but its d02 and d03 are larger. While the ghost runs generate output on all three domains, only d03 is used. The ghost d03 is mainly used as a first guess for high-resolution (0.02^o) data assimilation. In the operational HWRF, when inner core observations are not used, data assimilation is only performed in the parent domain (0.18^o). However, the ghost run is still performed and input to the vortex relocation procedures as a placeholder.

The HWRF vortex relocation process has three possible stages, which are determined based on the intensity of the observed storm and on the availability of the 6-h forecast of the previous HWRF run (Figure 4.2). If the previous cycle HWRF forecast exists, and if the observed storm intensity is equal to or greater than 16 ms^{-1}, HWRF is run in cycled mode. In this case, the previous HWRF cycle's 6-h forecast vortex, adjusted according to the TCVitals, is used for initializing the current cycle. If those conditions are not met, the HWRF initialization is a "cold start".

For a cold start of storms with observed intensity less than 30 ms^{-1}, the HDAS vortex is adjusted and then used. Conversely, for storms with observed intensity equal to or greater than 30 ms^{-1}, a bogus vortex is used. A cycled run will go through all the three stages, while a "cold start" run will go through stages 2 and 3 only.

Stage 1: The previous cycle 6-hr HWRF forecast is separated into environment fields and a storm vortex. This step is run only for cycled cases.
Stage 2: The preliminary IC generated by real_nmm and the WRF ghost and analysis runs is separated into environment fields and a storm vortex.
Stage 3: The storm vortex from the previous cycle's 6-hr forecast (for cycled runs), from the HDAS, or from the bogus vortex is adjusted to match the current time observed location, intensity and structure provided by the NHC. Then the vortex and environment fields are combined.

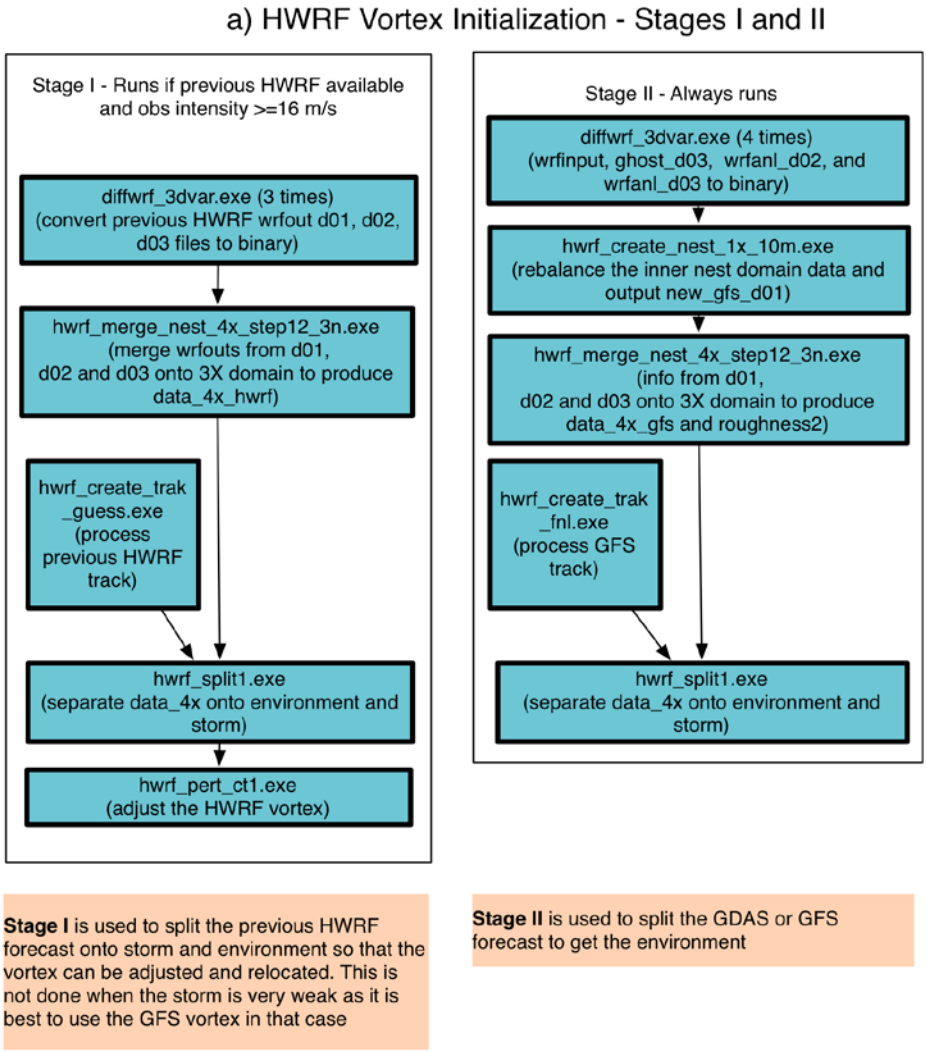


Figure 4.2a. Simplified flow diagram of the vortex initialization stages 1 and 2.

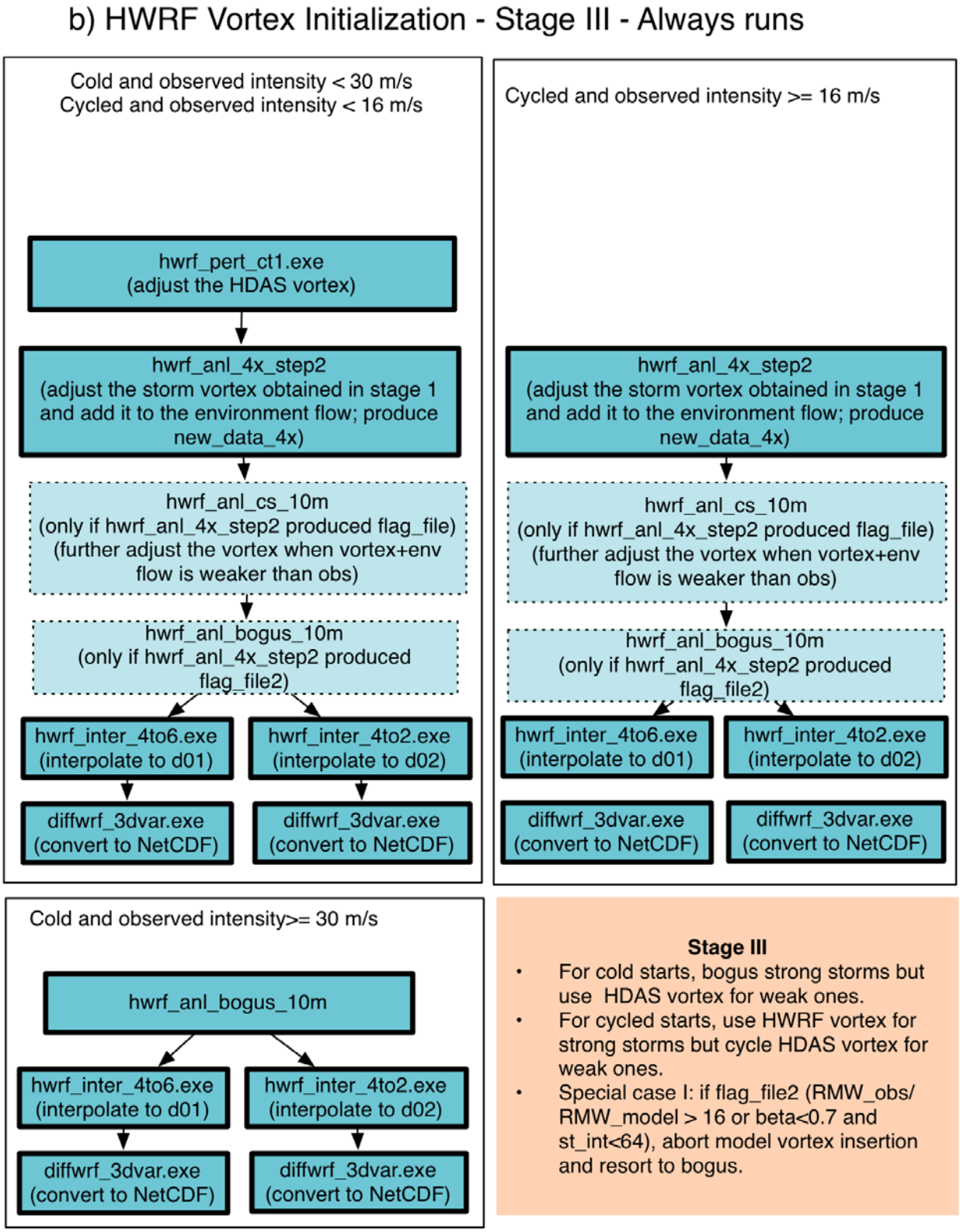


Figure 4.2b. Simplified flow diagram of the vortex initialization stage 3.

Since the vortex relocation stage 2 involves the removal of the cortex from the HDAS fields, it is necessary for the location of the vortex in those fields to be known. To that effect, before the vortex relocation, a procedure called “track analysis” is run. This entails running postprocessing the WRF analysis runs using UPP (unipost.exe and copygb.exe) and the GFDL vortex tracker to obtain the 0-h storm location.

After the three vortex relocation steps are finished, the adjusted IC for all domains are merged to provide the final IC for the HWRF 5-day forecast (Figure 4.3). Note that even though one of the input files to the merge process is called *wrfghost_d02*, this file is actually *wrfghost_d03*, renamed. This nomenclature is a legacy of the 2011 configuration of HWRF, for which the highest resolution domain was d02.

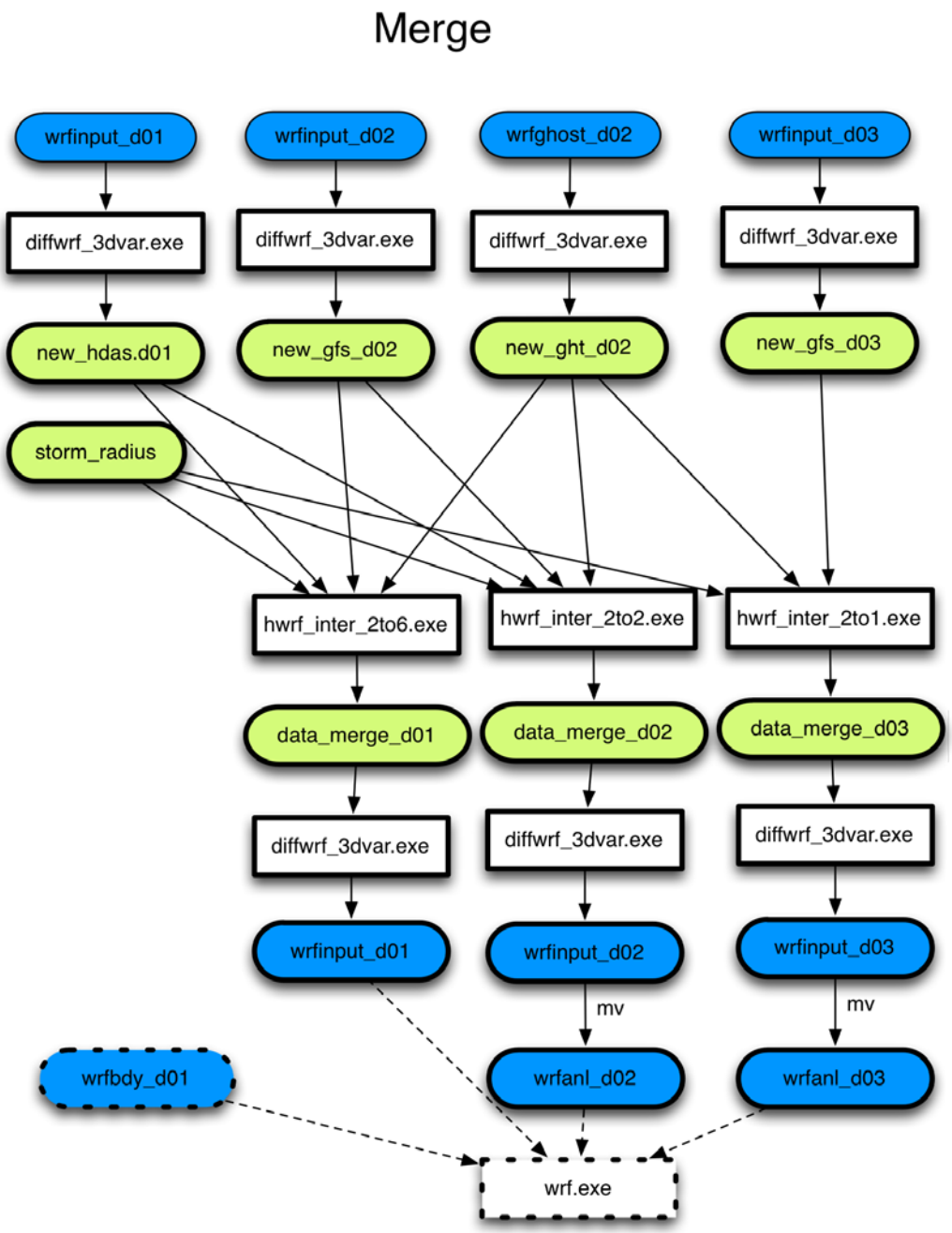


Figure 4.3. Diagram of vortex initialization merge procedures. The color coding is described in Figure 4.1.

4.2 Domains Used in HWRF

Figure 4.4 and Table 4.1 show the grids used in the HWRF forecast, data assimilation, and vortex initialization process. The ghost, analysis and 3X domains are not used during the forecast model integration — they are employed only in the HWRF initialization procedures

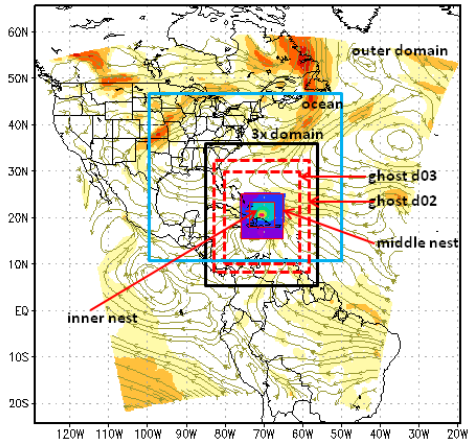


Figure 4.4. The domains used in HWRF.

	D01	**D02**	**D03**
Grid spacing (deg)	0.18	0.06	0.02
HWRF Forecast	216x432 - 80^ox80^o	88x170 - 11^ox10^o	180x324 - 7.2^ox6.5^o
Analysis run	216x432 - 80^ox80^o	88x170 - 11^ox10^o	180x324 - 7.2^ox6.5^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, number of grid points, and size of the HWRF atmospheric grids.

Note that the 3X domain is used in the removal of the vortex from the HDAS fields during the vortex adjustment. The domain is large enough so that the HDAS vortex is completely filtered out, yet small enough to save computing resources.

The ocean model grid placement depends on the position of the observed and 72-h NHC forecast of the observed storm. As an example, in Figure 4.4 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.

4.3 How to Run the HWRF Initialization Using Scripts

The HWRF vortex initialization scripts come in the tarfile hwrfv3.5a_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, 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 the LSF and MOAB/Torque systems, please see Section 1.6.

4.3.1 *real_wrapper*

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

WRF_ROOT
HWRF_UTILITIES_ROOT
DOMAIN_DATA
IO_FMT
GFS_GRIDDED_DIR
HWRF_SCRIPTS
START_TIME
FCST_LENGTH
FCST_INTERVAL
ATMOS_DOMAINS
MPIRUN
REAL_CORES
BASIN
DTT
DYY
DXX
NX1
NY1
RUN_PREP_HYB
BKG_MODE

Next use the *qsub* 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.

Overview of script *real.ksh*:

1. Initialize the function library, and check 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 *${DOMAIN_DATA}/realprd/${YYYYMMDDHH}*

wrfinput_d01: ICs created from GDAS
wrfbdy_d01: LBCs created from GFS for the ghost and analysis runs
fort.65: high-resolution sea mask data

Output files in directory ${DOMAIN_DATA}/realprd_GFS/${YYYYMMDDHH}:

wrfinput_d01: ICs created from GFS
wrfbdy_d01: LBCs created from GFS for the HWRF forecast
fort.65: high-resolution sea mask data

Status Check

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

4.3.2 *gsi_wrfinput_wrapper*

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

START_TIME
CYCLE_DATA
DOMAIN_DATA
GFS_GRIDDED_DIR
GSI_ROOT
GSI_CORES
MPIRUN
HWRF_UTILITIES_ROOT
DOMAIN
SID
IO_FMT
GDAS_OBS_DIR
GFS_OBS_DIR
GSI_FIXED_DIR
GSI_CRTM_FIXED_DIR
GEFS_ENS_FCST_DIR
RUN_GSI
STORM_NAME
BKG_MODE

GSI_ENS_REG
GSI_ENS_REG_SIZE
GSI_ENS_REG_OPT
GSI_USE_RAD

Note that in the operational HWRF radiances are not assimilated, so *global_vars.ksh* has *GSI_USE_RAD=F*. If radiance assimilation is desired, this variable can be set to *T*.

Next use the *qsub* 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_wrfinput_wrapper*, which in turn will run the low-level script *gsi.ksh*, on the parent domain.

Overview of script *gsi.ksh*

1. Initialize the function library and check if all the environment variables are set and the executables exist.
2. Create and enter the work directory.
3. Flag the observational data near the storm center so it is not assimilated.
4. Copy the fixed data and background analysis to the working directory.
5. Create a namelist for GSI analysis.
6. Copy the ensemble data to the working directory, if *GSI_ENS_REG* is set to *T*.
7. Run the executable *gsi.exe*

Output files in directory *${DOMAIN_DATA}/gsiprd/wrfinput/*

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 *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_v3.2.pdf*).

4.3.3 *wrfanalysis_wrapper*

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

WRF_ROOT
HWRF_UTILITIES_ROOT
DOMAIN_DATA
START_TIME
FCST_LENGTH

FCST_INTERVAL
WRF_MODE
IO_FMT
ATMOS_DOMAINS
MPIRUN
WRF_ANAL_CORES
BASIN
DTT
DYY
DXX
NX1
NY1
VERT_LEV
BKG_MODE

Note that in the wrapper script *wrfanalysis_wrapper*, the following variable is defined and should not be altered.

WRF_MODE = analysis

Next, use the *qsub* 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_ANAL_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-s 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 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 *${DOMAIN_DATA}/wrfanalysisprd/${YYYYMMDDHH}*

"*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.**.

4.3.4 *wrfghost_wrapper*

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

WRF_ROOT
HWRF_UTILITIES_ROOT
DOMAIN_DATA
START_TIME
FCST_LENGTH
FCST_INTERVAL
WRF_MODE
IO_FMT
ATMOS_DOMAINS
MPIRUN
WRF_GHOST_CORES
BASIN
DTT
DYY
DXX
NX1
NY1
VERT_LEV
BKG_MODE

Note that in the wrapper script *wrfghost_wrapper*, the following variable is defined and should not be altered.

WRF_MODE = ghost

Next use the *qsub* 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_GHOST_CORES.*

Then 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-s run of *wrf.exe* and generate output for the middle and inner "ghost" domains (see Figure 4.3). The ghost d03 domain is primarily used in the assimilation of inner core data (see Section 4.5), but it must be run as a placeholder even when inner core data is not available.

Overview of script *wrf.ksh*

1. Initialize the function library, and check 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 *${DOMAIN_DATA}/wrfghostprd/${YYYYMMDDHH}*

"*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.*

4.3.5 *track_analysis_wrapper*

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

HWRF_UTILITIES_ROOT
UPP_ROOT
TRACKER_ROOT
DOMAIN_DATA
START_TIME
SID
UNI_CORES
IO_FMT
ATMOS_DOMAINS
MPIRUN
FHR

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 4.3.3) *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 4.5).

Overview of script *track_analysis.ksh*

1. Initialize the function library, and check 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 timestamp in this 90-s *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 *${DOMAIN_DATA}/trkanalysisprd/${YYYYMMDDHH}*

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.

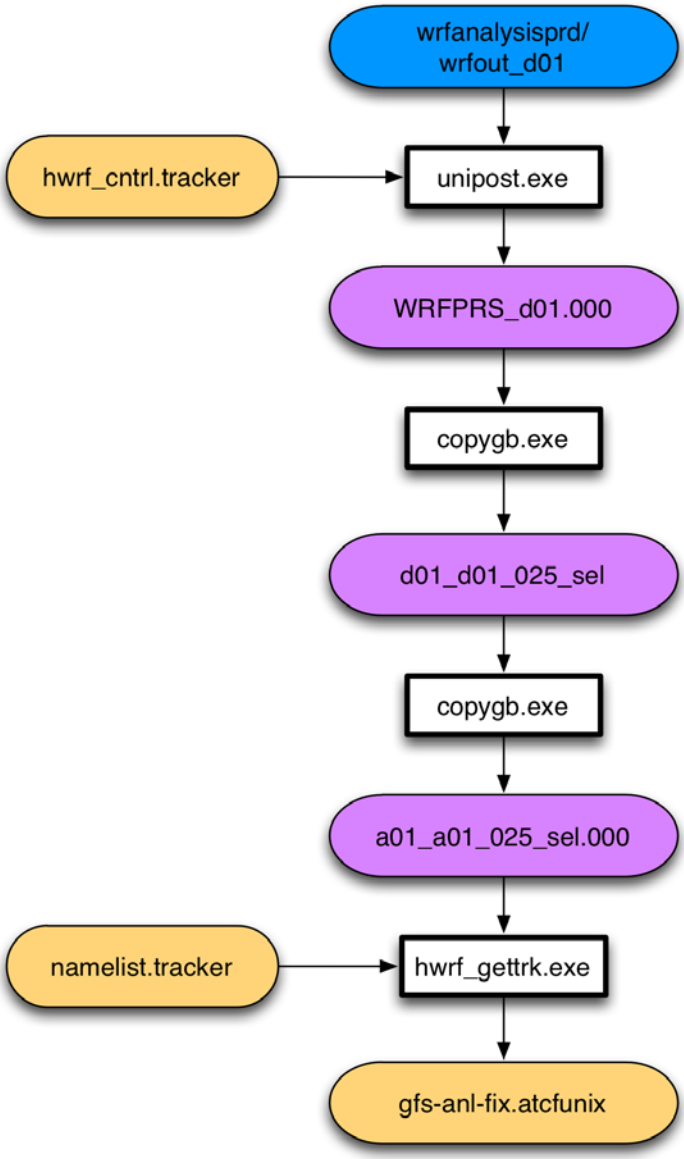


Figure 4.5. Diagram of the procedure to generate information about the position of the storm in the GDAS input data. The color coding is described in Figure 4.1, with the addition of orange which represents ASCII files.

4.3.6 *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
FCST_LENGTH
RUN_PREP_HYB
ATCFNAME
SID
MPIRUN
BASIN
INNER_CORE_DA

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 16 ms^{-1} (a cycled run), the previous forecast will be interpolated onto the 3X domain and separated into environment fields and storm vortex fields. The storm vortex fields will be adjusted. The 3X domain is approximately $30^{o}x30^{o}$, has the resolution of the inner nest domain, and is centered based on the NHC storm message data (see Figure 4.4). Be aware that the 3X ($30^{o}x30^{o}$) 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.

Overview of script relocate_stage1_3d.ksh

1. Initialize the function library and check 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 if the previous cycle forecast exists and the storm intensity is greater than 16 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 *${DOMAIN_DATA}/relocateprd/${YYYYMMDDHH}/stage1*

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.

4.3.7 *relocate2_wrapper*

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

HWRF_UTILITIES_ROOT
DOMAIN_DATA
START_TIME
SID
BASIN
FCST_INTERVAL
IO_FMT
ATMOS_DOMAINS
GFS_GRIDDED_DIR
BKG_MODE
RUN_GSI
RUN_GSI_WRFINPUT
INNER_CORE_DA

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 the 3X domain grid. The merged fields will be separated into environment fields and storm vortex.

Overview of script *relocate_stage2.ksh*

1. Initialize the function library and check 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*.
6. For example, for a forecast of hurricane Sandy starting at 06 UTC on 10/28/2012, the storm ID is 18L, file *atcfunix* shows the following storm information:

AL, 18, 2012102806, 03, HWRF, 000, 316N, 737W, 40, 962, XX, 34, NEQ, 0350, 0332, 0308, 0327, 0, 0, 108
AL, 18, 2012102806, 03, HWRF, 000, 316N, 737W, 40, 962, XX, 50, NEQ, 0000, 0000, 0201, 0000, 0, 0, 108

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

72HDAS121028 6 316 737 316 737 316 737 316 737 0 0 0 0 0 0 18L

where '72HDAS' is a fixed field, 121028 6 means 10/28/2012 06 UTC, 316 and 737 are the latitude and longitude multiplied by 10 (31.6N and 73.7W), and 18L is the storm ID.

1. 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*).
2. 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 *${DOMAIN_DATA}/relocateprd/${YYYYMMDDHH}/stage2*

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.

4.3.8 *relocate3_wrapper*

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

HWRF_UTILITIES_ROOT
DOMAIN_DATA
START_TIME

SID
FCST_INTERVAL
IO_FMT
ATMOS_DOMAINS
GFS_GRIDDED_DIR
CYCLE_DATA
MPIRUN
BASIN
BKG_MODE
INNER_CORE_DA

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_3d.ksh*. This will create a new storm vortex by adjusting the previous cycle 6-hr forecast vortex (for a cycled run) or the HDAS or bogus vortex (for a cold start) to match the observed storm location, intensity and structure.

Overview of script relocate_stage3_3d.ksh:

1. Initialize the function library and check 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 16 ms^{-1}),
   d. 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.
   e. 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 *${DOMAIN_DATA}/relocateprd/${YYYYMMDDHH}/stage3*

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.

4.3.9 *gsi_wrfghost_wrapper*

Only when performing inner-core data assimilation to use TDR observations, it is necessary to run *gsi_wrfghost_wrapper* to perform the GSI analysis in the high-resolution ghost domain. When using *gsi_wrfghost_wrapper*, it should be run before *merge_wrapper*. Detailed information about TDR assimilation can be found in Section 4.5.

Wrapper *gsi_wrfghost_wrapper* is similar to wrapper *gsi_wrfinput_wrapper* detailed in Section 4.3.2., with the exception that variable *DOMAIN* should be set to *wrfghost.*

4.3.10 *merge_wrapper*

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

HWRF_UTILITIES_ROOT
DOMAIN_DATA
CYCLE_DATA
BASIN
START_TIME
SID
IO_FMT
ATMOS_DOMAINS
FCST_INTERVAL
RUN_GSI
RUN_GSI_WRFINPUT
RUN_GSI_WRFGHOST

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, as it should in the HWRF operational configuration, 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 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_DATA}/mergeprd*

wrfinput_d01: initial condition for the outer domain containing the new vortex
wrfanl_d02_*${*YYYY-MM-DD_HH}:00:00*: initial condition for the middle nest domain containing the new vortex. *${YYYY-MM-DD} is the model run's initial time.
wrfanl_d03_*${*YYYY-MM-DD_HH}:00:00*: initial condition for the inner nest domain containing the new vortex. *${YYYY-MM-DD_HH} 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.

4.4 HWRF Initialization Executables

4.4.1 *copygb.exe*

See Section 7.4.2.

4.4.2 *diffwrf_3dvar.exe*

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.

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 *get_trk.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 by script *gsi.ksh* by modifying template *${HWRF_UTILITIES_ROOT}/parm/gsi_namelist.input*
filelist: ASCII file with 80 lines, each one containing a file name for a GEFS ensemble member (used for ensemble-based background covariance)
satbias_angle: file containing information on satellite angle, from dataset directory *GDAS/obs*

satbias_in: file containing information on satellite bias, from dataset directory *GDAS/obs*
wrf_inout: background file, copied from *${BK_DIR}*
prepbufr: conventional observation prepBUFR data, linked to *${PREPBUFR}*.
tldplrbufr: TDR observations (if using)

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

GEFS 6-h forecast initialized 6-h before current HWRF cycle for all 80 members (80 files)

GEFS 80-member forecast

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

USAGE:
On machines with LSF: *mpirun.lsf -procs 12 ./gsi.exe < gsiparm.anl*
On machines with MOAB/Torque: *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
storm_sym (fort.23) - symmetric part of storm
gfs_env (fort.26) - GFS environmental flow
roughness1 (fort.46) - roughness from *merge_nest_4x_step2*
storm_pert_new (fort.71) - adjusted storm perturbation from stage 1

OUTPUT:
wrf_env_new (fort.36) - new environmental flow.

new_data_4x (fort.56) - adjusted field on 3X domain.

USAGE:
echo $gesfhr $BASIN 0 1 | 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
roughness2 (fort.46) - roughness info for boundary layer calculation
storm_pert_gfs (fort.61) - separated GFS 3D vortex field
storm_radius_gfs (fort.85)
hwrf_storm_cyn_axisy_47 (fort.71,72,73,74,75,78) input static vortex data
hwrf_storm_20 (fort.76, 77) input static vortex data

OUTPUT:

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

USAGE:
echo $gesfhr $BASIN | 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.71,72,73,74,75,78) input static vortex data
hwrf_storm_20 (fort.76, 77) input static vortex data

OUTPUT:
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 $BASIN | 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 $BASIN | 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 $BASIN | hwrf_create_trak_guess.exe

4.4.10 *hwrf_data_flag.exe*

FUNCTION:
Flags the observational data near storm center so it is not assimilated

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_flag.exe > data_flag.out

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} $BASIN | 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} $BASIN | 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 $BASIN | 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 $BASIN | 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 $BASIN | 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 $BASIN | 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 $BASIN | 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 $BASIN | hwrf_split.exe

4.4.19 *hwrf_wrfout_newtime.exe*

FUNCTION:
Changes the time stamp of WRF analysis run d01 output from 90 s 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 *unipost.exe*

See Section 7.4.1.

4.5 Inner-core Data Assimilation

HWRF has the capability of assimilating tropical cyclone inner-core data such as the NOAA's P3 TDR observation using GSI. The scripts default to the assimilation of TDR data, when available. If inner core data assimilation is not desired, it can be disabled by setting *INNER_CORE_DA=0* in file *global_varks.ksh*.

To collect inner-core observations, an aircraft has to penetrate the target TC multiple times to finish one mission, which may take several hours; therefore the observations in one TDR data set are collected at different times.

In order for GSI to calculate the innovation, defined as the difference between the first guess and the analysis, it needs to have the first guess and the observations valid at the same time. To accomplish this for observations that spawn a range of times, a procedure named FGAT is used. In FGAT, first guess fields valid at various times are supplied to GSI, which then interpolates the data to the time in which each observation was taken.

For HWRF, first guess fields are created at three time levels: 3 h before the HWRF initial time (Figure 4.6-part 1), at the HWRF initial time (Figure 4.6-part 2), and 3 h after the HWRF initial time (Figure 4.6-part 3). In order to create the three first guesses, the real_nmm, short WRF forecasts, and vortex adjustment procedures are performed three times. This produces the three ghost d03 output files that are used by GSI in its FGAT operation (Figure 4.7)

The inner core data assimilation is only performed on the ghost d03 domain. Note that when inner core data assimilation is performed, the source of data to initialize the ghost and analysis runs is not HDAS but GDAS. In other words, the d01 GSI analysis is not an input to the ghost and analysis runs. This is done to avoid duplication, since the data assimilation near the storm center will be performed in the ghost d03 domain.

In order to perform the data assimilation in the ghost domain, users should run *gsi_wrfghost_wrapper* before *merge_wrapper*.

After the data is assimilated in the ghost d03, the GSI analysis on the parent domain, the middle and inner domain output from the vortex adjust procedures, and the ghost d03 GSI analysis (which used FGAT) are merged to produce the final atmospheric IC for the 5-day forecast. The merging procedure is identical for the cases with or without inner core data assimilation. Therefore, the box named "merge" in Figure 4.7 corresponds to the procedures detailed in Figure 4.3.

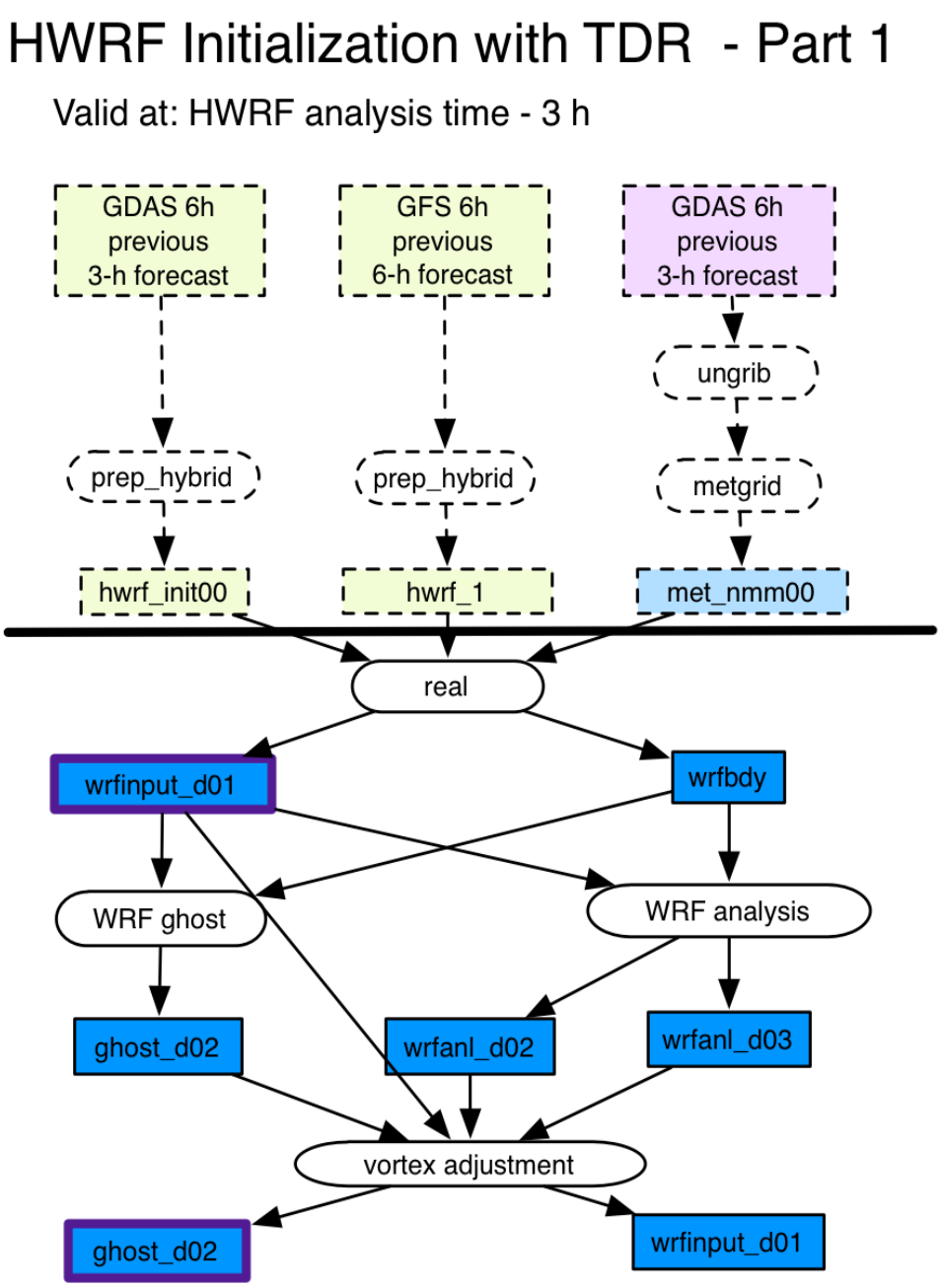


Figure 4.6- part 1. Diagram of the HWRF initialization procedures when hurricane inner-core data assimilation is conducted. The processes shown below the thick horizontal line are described in this chapter. Part 1 refers to the processes at valid time 3 h before the HWRF initialization. Some boxes are shown in Figures 4.6 and 4.7 with purple outline to indicate their correspondence.

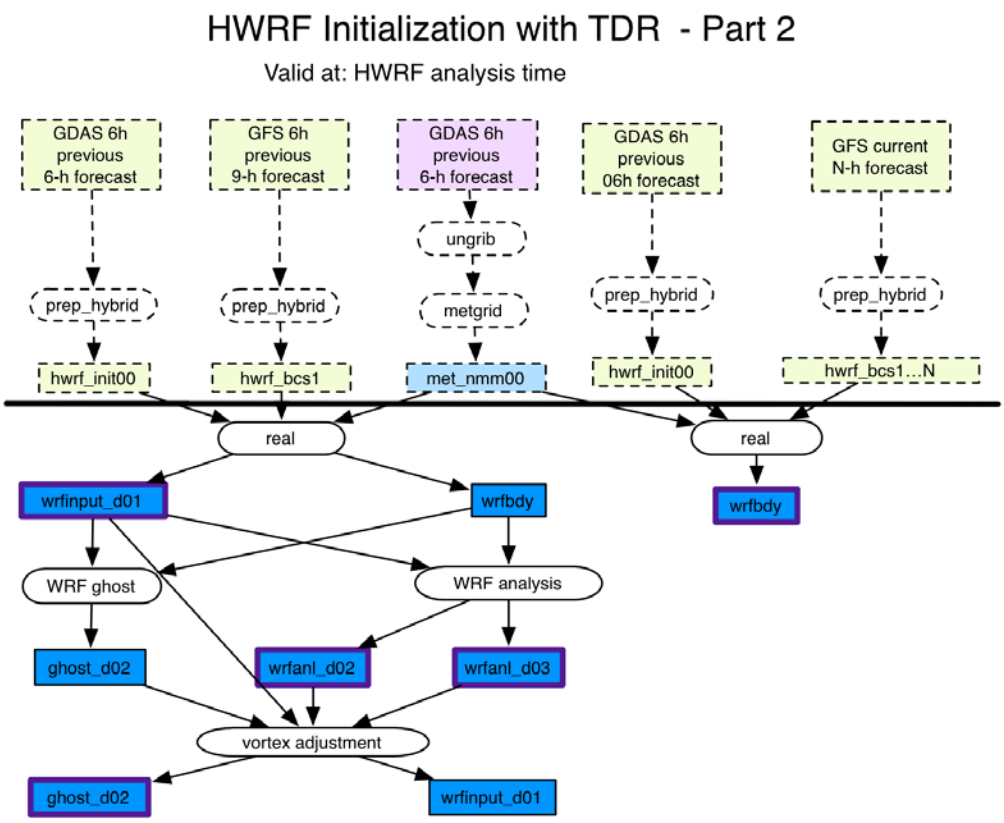


Figure 4.6- part 2. Same as Fig 4.6 - part 1, except for processes valid at the time of HWRF initialization.

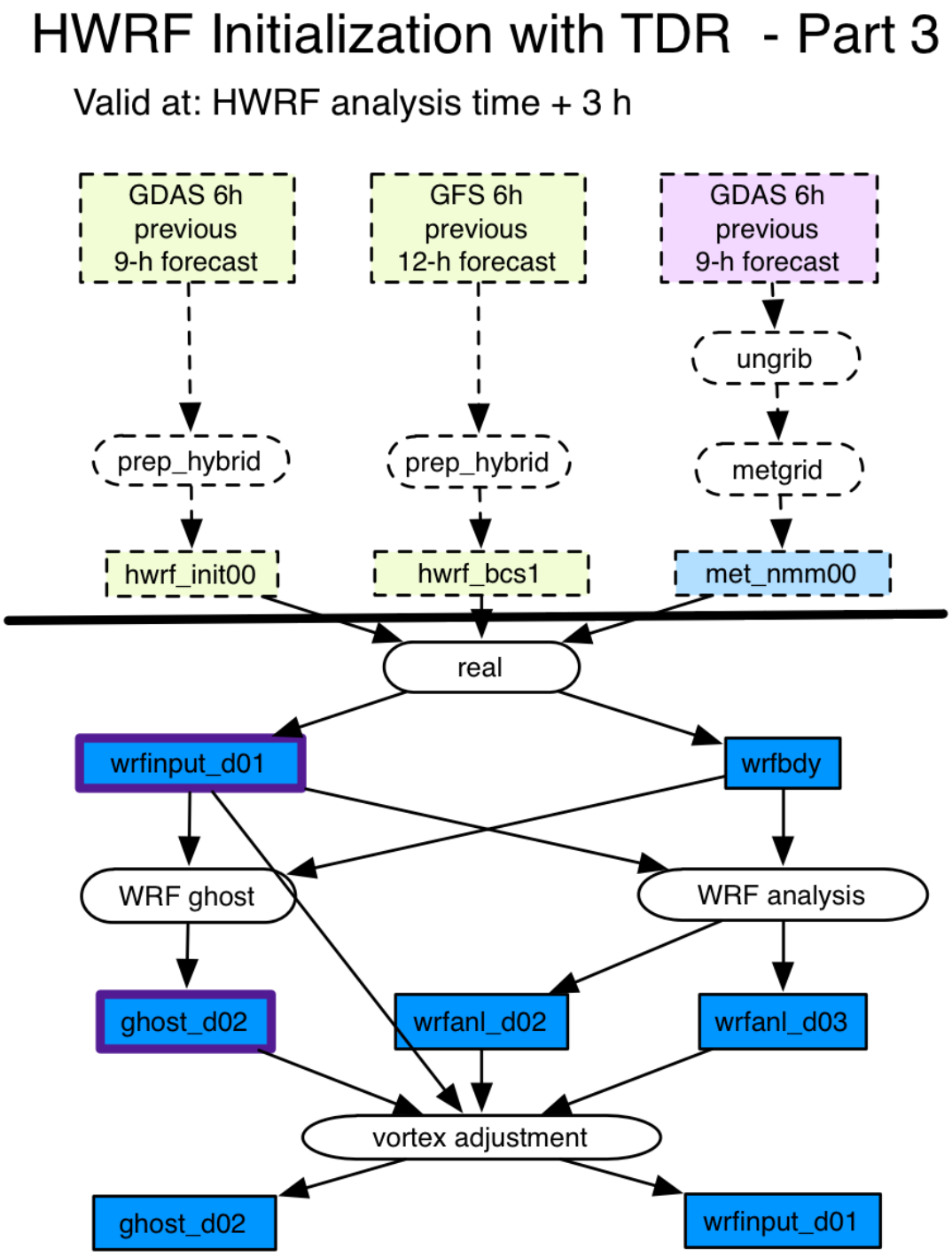


Figure 4.6- part 3. Same as Fig 4.6 - part 1, except for processes at valid time three hours after the HWRF initialization.

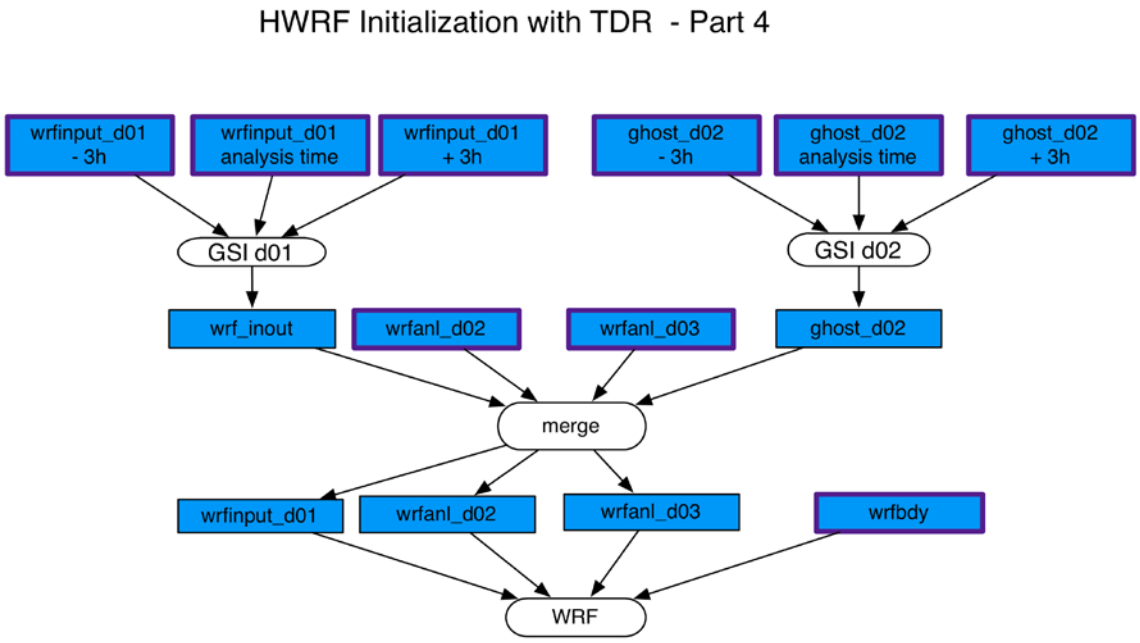


Figure 4.7 Diagram of the data assimilation procedures when inner core data assimilation is performed. Some boxes are shown in Figures 4.6 and 4.7 with purple outline to indicate their correspondence.

Chapter 5: Ocean Initialization of POM-TC

5.1 Introduction

This chapter explains how to run the initialization of the POM-TC component of the HWRF model, available from the DTC. Users are also encouraged to read the HWRF v3.5a 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 wrapper script first calls script *global_vars.ksh* to define the environmental variables. Next, it calls the low-level script *pom_init.ksh* to run the ocean initialization. Another script, *gfdl_pre_ocean_sortvit.sh*, is called from within *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

The output files from *hwrf-utilities/pom_init_wrapper* will be in *${DOMAIN_DATA}/oceanprd.*

Scripts *pom-tc-united-grads.sh*, *pom-tc-eastatl-grads.sh*, and *pom-tc-eastpac-grads.sh*, to plot the POM-TC ocean output using the GRADS software, can be found in directory *${SCRATCH}/HWRF/pomtc/ocean_plot*.

5.3 Functions in Script "*pom_init.ksh*"

The tasks accomplished by each of the functions called in script *pom_init.ksh* are described below.

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 SST 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*.

Files *track.allhours* and *track.shortened* will be produced in directory *${DOMAIN_DATA}/oceanprd*

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 *${DOMAIN_DATA}/oceanprd/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.

Files *lonlat.gfs, mask.gfs.dat, sst.gfs.dat* will be produced in *${DOMAIN_DATA}/oceanprd/getsst.*

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.

File *gfdl_initdata.${region}.10* will be produced in *${DOMAIN_DATA}/oceanprd/sharpn.*

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 for 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 *${DOMAIN_DATA}/oceanprd/phase3/RST.phase3.${region}* 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 *${DOMAIN_DATA}/oceanprd/phase4/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. This executable is not used in the operational HWRF configuration.

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 the Forecast Model

6.1 Introduction

The operational HWRF, which runs on the North Atlantic and Eastern North Pacific basins, 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.

In the non-operational basins, Central Pacific, West Pacific, and Indian Ocean, HWRF can only be run in atmosphere standalone mode, that is, uncoupled.

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*, which calls the low-level script *wrf.ksh*, to run the HWRF forecast on two batch systems: MOAB/Torque and LSF. 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).

WRF_ROOT
HWRF_UTILITIES_ROOT
DOMAIN_DATA
START_TIME
FCST_LENGTH
FCST_INTERVAL
WRF_MODE
IO_FMT
ATMOS_DOMAINS
MPIRUN
HWRF_FCST_CORES
BASIN
DTT
DYY
DXX
NX1
NY1
VERT_LEV
BKG_MODE

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

WRF_MODE = main
HWRF_FCST_CORES =202

Note that *HWRF_FCST_CORES* includes one processor for the coupler and one for POM-TC. The user can define *HWRF_FCST_CORES* to a different number. To calculate the number of cores needed:

1. Number of IO groups and servers per group. The WRF namelist defines 4 IO server groups with 4 servers per group (total of 16 processors).
2. Number of processors in the X and Y directions.
3. 1 processor for POM-TC.
4. 1 processor for the coupler.

For example:

16 + (8 * 23) + 1 + 1 = 202

It has been shown that *HWRF_FCST_CORES* can be defined as the following numbers (including the two processors for the coupler and POM-TC):

6, 10, 18, 26, 34, 38, 50, 66, 92, 102, 122, 152, 182, 202, 225, 248 and 271.

The following numbers do not work:

73, 39, and 43.

For uncoupled runs the two cores allocated for POM-TC and the coupler are not used. The user does not need to change the number of cores allocated (*HWRF_FCST_CORES*).

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

Overview of script *wrf.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. 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. For the AL and EP basins,
   a. Link the input files required by POM-TC, including fix files, initial and boundary conditions files, bathymetry/topography data files etc.
   b. Generate a namelist for POM-TC.
   c. Generate a namelist for the coupler.
6. Submit the MPI forecast run.

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 (if *high_freq=true* is set in the WRF namelist).

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.11082312 etc.*

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).*

Explanation of the MPI command for the forecast model

As mentioned in section 6.1, HWRF can be run as either a coupled or uncoupled model of the atmosphere and ocean. The operational HWRF runs coupled in the North Atlantic and Eastern North Pacific basins. Script *wrf.ksh* automatically submits coupled runs in the North Atlantic and Eastern North Pacific basins, and uncoupled runs in other basins. If an uncoupled run in the North Atlantic and Eastern North Pacific basins is desired, script *wrf.ksh* needs to be manually altered to use the uncoupled submission command described below.

- Coupled

  With LSF, using 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
  wrf.exe
  wrf.exe

  With MOAB/Torque, using the command *mpiexec*

  mpiexec -np 1 ./hwrf_wm3c.exe : -np 1 ./hwrf_ocean_united.exe : -np 200 ./wrf.exe

  For example, the previous 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*

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

- Uncoupled

  With LSF, using the command *mpirun.lsf*

  mpirun.lsf -procs 200 ${WRF_ROOT}/main/wrf.exe

  With MOAB/Torque, using the command *mpiexec*

  mpiexec -np 200 ${WRF_ROOT}/main/wrf.exe

6.3 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 6.5. 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 (PRELIMINARILY TESTED)
Cumulus	HWRF SAS (84)	Tidtke (6), KF (1, trigger option 1), New SAS (14)
Microphysics	Tropical Ferrier (85)	WSM6 (6), Thompson (8)
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=2,2,6 for the internal nest tracking. While other namelist options for tracking the vortex are available, they are not recommended, as they do not track the nest as reliably as the setup used in the operational configuration.

Users have the option of getting a high frequency (each time step) 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.

6.4 Executables

6.4.1 *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*
Lateral boundary conditions: *wrfbdy_d01*
Initial conditions for the parent domain: *wrfinput_d01*

Initial conditions for the nests: *wrfanl_d02_${YYYY}-${MM}-${DD}_${HH}:00:00* and *wrfanl_d03_${YYYY}-${MM}-${DD}_${HH}:00:00*
Gravity wave drag file *gwd_surface*
WRF static files
namelist.input
fort.65

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).
For an uncoupled run,
wrf.exe

6.4.2 *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).

6.4.3 *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.
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).

6.4.4 *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.

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).

6.4.5 *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.

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).

6.4.6 *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.
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).

6.4.7 *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 (99,225) on the parent domain grid.

istart=00099
jstart=00225

USAGE:

hwrf-utilities/exec/hwrf_swcorner_dynamic.exe

6.5 Sample HWRF namelist

The HWRF namelist used for the release case, Hurricane Sandy (2012), is listed below.

```
 &time_control
start_year                          = 2012, 2012, 2012,
start_month                         = 10, 10, 10,
start_day                           = 28, 28, 28,
start_hour                          = 06, 06, 06,
start_minute                        = 00,   00,   00,
start_second                        = 00,   00,   00,
end_year                            = 2012, 2012, 2012,
end_month                           = 11, 11, 11,
end_day                             = 02, 02, 02,
end_hour                            = 12, 12, 12,
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, 180,
s_sn = 1, 1, 1,
e_sn = 432, 170, 324,
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, 00099, 14,
j_parent_start = 0, 00225, 33,
feedback = 1,
num_moves = -99
num_metgrid_levels = 27,
p_top_requested = 5000,
ptsgm = 42000
eta_levels=1.0,.9919699,.9827400,.9710800,.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 = T,
num_metgrid_soil_levels = 2,
/
&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,
 var_ric                             = 1.0,
 coef_ric_l                          = 0.16,
 coef_ric_s                          = 0.25,
 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                                = 2,6,6,
 ncnvc                               = 2,6,6,
 movemin                             = 3,6,18,
! 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                           = 0.7,0.7,0.7,
 sas_pgcon                           = 0.55,0.2,0.2,
sas_mass_flux=0.5,0.5,0.5,
 co2tf                               = 1,
vortex_tracker=2,2,6,
! 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=.true.
 nio_tasks_per_group                 = 4,
 nio_groups                          = 4 /
 &logging
  compute_slaves_silent=.true.
```

```
 io_servers_silent=.true.
 stderr_logging=0
/
```

Chapter 7: HWRF Post Processor

7.1 Introduction

The NCEP UPP is used to de-stagger the HWRF parent and nest domain output, compute diagnostic variables, and interpolate the output from the native WRF 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.). The UPP outputs files in GRIB format. This package also merges 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 *unipost.ksh* are distributed in the tar file *hwrfv3.5a_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* and *unipost_wrapper* to make sure the following variables are correctly defined (see Appendix).

HWRF_SCRIPTS
UPP_ROOT
HWRF_UTILITIES_ROOT
FHR
DOMAIN_DATA
ATMOS_DOMAINS
CRTM_FIXED_DIR
START_TIME
SID
UNI_CORES
IO_FMT
UPP_PROD_SAT
MPIRUN

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

Note that other UPP scripts are distributed in the UPP release tar file *hwrfv3.5a_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.* . Note that script *run_grads* calls Perl script *grib2ctl.pl*. Users can download this script from *www.cps.ncep.noaa.gov/products/wesley/grib2ctl.html.* Script grib2ctl.pl needs to be edited to contain the correct path to *grib2ctl.pl.* Additionally, note that script *grib2ctl.pl* makes use of utility wgrib. Therefore, script *grib2ctl.pl* needs to be edited to point to *${SCRATCH}/HWRF/hwrf-utilities/exec/wgrib.exe.*

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. A control file is used to specify which variables will be output (for more information, see WRF-NMM documentation).
4. If *UPP_PROD_SAT=F*, no synthetic satellite images will be produced, and the control file *hwrf-utilities/parm/hwrf_cntrl.hurcn* is utilized to specify the output variables. Conversely, if *UPP_PROD_SAT=T*, synthetic satellite images will be output along with other variables by using basin-dependent control files. These control files are listed in Section 7.4.1.
5. If changes in the post-processed variables are desired, the control file can be altered. For HWRF, the following variables, which are required by the GFDL vortex tracker (see Chapter 8), should be postprocessed:
   - 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.
6. 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*. 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)
7. Run *copygb.exe* to horizontally interpolate the native UPP output files to a variety of regular lat/lon grids.

8. Create merged UPP output files. In particular, create merged file for input in the GFDL vortex tracker.

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

A general overview of the files produced by script unipost.ksh is provided in Figure 7.1.

- The following three files are in GRIB format on the HWRF native horizontal grids.
  - *WRFPRS_d01.${fhr}* for the HWRF parent domain
  - *WRFPRS_d02.${fhr}* for the HWRF middle nest domain
  - *WRFPRS_d03.${fhr}* for the HWRF inner nest domain
- The following files are in GRIB format on regular lat/lon grids. The name convention is "forecast domain(s)"_"interpolation domain"_"resolution"_ "variables"_"forecast lead time". For example, *d01_d01_010_all.006* is the 6-hour HWRF parent domain forecast output that has been interpolated to a regular lat/lon grid covering an area similar to the one of the parent domain, with a horizontal resolution of 0.1 degree, containing all the variables present in the *unipost.exe* output file.
  - Grid "d02p" is slightly larger than the middle nest domain, while grid "t02" is approximately 20^{o}x20^{o} and used by the GFDL vortex tracker to extract the track.
  - If "variables"="all", all the variables from the *WRFPRS* files are included in the interpolated GRIB file. When only those variables required by the GFDL vortex tracker are retained, "variable"="sel".
  - Files whose names start with "merge" are the result of combining two or more domains together to generate a single output file.
    - *d01_d01_010_all.${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.${fhr}*
    - *merged_d01d02d03_d01_010_sel.${fhr}*
    - *merged_d01d02d03_d02p_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.

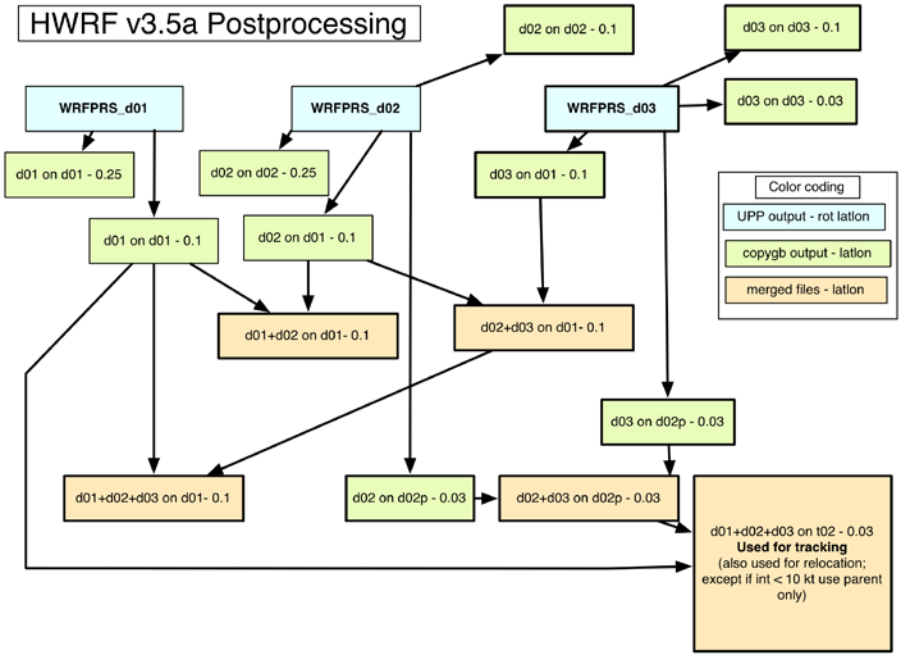


Figure 7.1. Overview of the files produced by script unipost.ksh. See text for details.

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*

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

namelist *itag*

unipost control file. When *UPP_PROD_SAT=T*, the file is *${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_cntrl.hurcn.* When *UPP_PROD_SAT=T*, a basin-dependent file

located in *${SCRATCH}/HWRF/hwrf-utilities/parm* is used. Files *hwrf_cntrl.satL, hwrf_cntrl.satE, hwrf_cntrl.sat=C, and hwrf_cntrl.satW* are used for the North Atlantic, Eastern North Pacific, Central North Pacific, West North Pacific basins, respectively. File *hwrf_cntrl.sat_other* is used for other basins.

OUTPUT:
HWRF postprocessed 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:
1. interpolates a GRIB file to a user-specified grid
2. 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:
1. ${SCRATCH}/HWRF/UPP/bin/copygb.exe -xg"${hr_grid}" input_GRIB_file out_GRIB_file
2. 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}* onto *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 metrics of the forecast storm, such as intensity (maximum 10-m winds and MSLP) and structure (wind radii for 34-, 50-, and 64-knot thresholds in each quadrant of the 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 locates 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 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 *hwrfv3.5a_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 reads the HWRF postprocessed files in 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 Section 8.6.1).
8. Run the tracker executable *hwrf_gettrk.exe*.
9. Output files 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 as this capability is not used operationally. 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 Sandy of 31.5N and 73.7W.

NHC 18L SANDY 20121028 0600 315N 0737W 040 062 0960 1006 0890 33 167 0834 0500 0500 0519 D -999 0278 0278 -999 72 405N 770W -999 -999 0167 -999

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 "2012")
inp%byy	Last 2 digits of the year for the initial time of the forecast (e.g., the "12" in "2012")
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 the model initialization time. 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 an additional 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, 18, 2012102806, 03, HCOM, 00000, 315N, 737W, 65, 949, XX, 34, NEQ, 0351, 0106, 0141, 0329, 0, 0, 36
AL, 18, 2012102806, 03, HCOM, 00000, 315N, 737W, 65, 949, XX, 50, NEQ, 0058, 0058, 0070, 0070, 0, 0, 36
AL, 18, 2012102806, 03, HCOM, 00000, 315N, 737W, 65, 949, XX, 64, NEQ, 0000, 0000, 0039, 0037, 0, 0, 36
AL, 18, 2012102806, 03, HCOM, 00300, 319N, 731W, 65, 951, XX, 34, NEQ, 0333, 0133, 0205, 0323, 0, 0, 30
AL, 18, 2012102806, 03, HCOM, 00300, 319N, 731W, 65, 951, XX, 50, NEQ, 0064, 0065, 0065, 0058, 0, 0, 30
AL, 18, 2012102806, 03, HCOM, 00300, 319N, 731W, 65, 951, XX, 64, NEQ, 0035, 0043, 0045, 0000, 0, 0, 30
AL, 18, 2012102806, 03, HCOM, 00600, 323N, 724W, 60, 956, XX, 34, NEQ, 0313, 0173, 0205, 0308, 0, 0, 39

AL, 18, 2012102806, 03, HCOM, 00600, 323N, 724W, 60, 956, XX, 50, NEQ, 0068, 0080, 0076, 0043, 0, 0, 39
AL, 18, 2012102806, 03, HCOM, 00900, 329N, 718W, 57, 959, XX, 34, NEQ, 0297, 0192, 0217, 0306, 0, 0, 43
AL, 18, 2012102806, 03, HCOM, 00900, 329N, 718W, 57, 959, XX, 50, NEQ, 0064, 0069, 0054, 0000, 0, 0, 43
AL, 18, 2012102806, 03, HCOM, 01200, 337N, 715W, 57, 959, XX, 34, NEQ, 0274, 0280, 0257, 0285, 0, 0, 42
AL, 18, 2012102806, 03, HCOM, 01200, 337N, 715W, 57, 959, XX, 50, NEQ, 0076, 0076, 0000, 0041, 0, 0, 42
AL, 18, 2012102806, 03, HCOM, 01500, 344N, 713W, 58, 958, XX, 34, NEQ, 0319, 0269, 0274, 0323, 0, 0, 41
AL, 18, 2012102806, 03, HCOM, 01500, 344N, 713W, 58, 958, XX, 50, NEQ, 0125, 0060, 0146, 0094, 0, 0, 41
AL, 18, 2012102806, 03, HCOM, 01800, 349N, 716W, 64, 957, XX, 34, NEQ, 0343, 0340, 0269, 0319, 0, 0, 106
AL, 18, 2012102806, 03, HCOM, 01800, 349N, 716W, 64, 957, XX, 50, NEQ, 0163, 0127, 0153, 0159, 0, 0, 106
AL, 18, 2012102806, 03, HCOM, 02100, 353N, 717W, 75, 955, XX, 34, NEQ, 0351, 0264, 0252, 0290, 0, 0, 39

Column 1: basin name. "AL" represents Atlantic and “EP” northeast Pacific.

Column 2: ATCF storm ID number. Sandy was the 18th storm in the Atlantic Basin in 2012.

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, 18, 2012102806, 03, HCOM, 000, 315N, 737W, 65, 949, XX, 34, NEQ, 0351, 0106, 0141, 0329, 0, 0, 36, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 000, 315N, 737W, 65, 949, XX, 50, NEQ, 0058, 0058, 0070, 0070, 0, 0, 36, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 000, 315N, 737W, 65, 949, XX, 64, NEQ, 0000, 0000, 0039, 0037, 0, 0, 36, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 003, 319N, 731W, 65, 951, XX, 34, NEQ, 0333, 0133, 0205, 0323, 0, 0, 30, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 003, 319N, 731W, 65, 951, XX, 50, NEQ, 0064, 0065, 0065, 0058, 0, 0, 30, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 003, 319N, 731W, 65, 951, XX, 64, NEQ, 0035, 0043, 0045, 0000, 0, 0, 30, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 006, 323N, 724W, 60, 956, XX, 34, NEQ, 0313, 0173, 0205, 0308, 0, 0, 39, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 006, 323N, 724W, 60, 956, XX, 50, NEQ, 0068, 0080, 0076, 0043, 0, 0, 39, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 009, 329N, 718W, 57, 959, XX, 34, NEQ, 0297, 0192, 0217, 0306, 0, 0, 43, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999

AL, 18, 2012102806, 03, HCOM, 009, 329N, 718W, 57, 959, XX, 50, NEQ, 0064, 0069, 0054, 0000, 0, 0, 43, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 012, 337N, 715W, 57, 959, XX, 34, NEQ, 0274, 0280, 0257, 0285, 0, 0, 42, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999
AL, 18, 2012102806, 03, HCOM, 012, 337N, 715W, 57, 959, XX, 50, NEQ, 0076, 0076, 0000, 0041, 0, 0, 42, 0, 0, , 0, , 0, 0, , , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, U, 10, DT, -999AL, 18, 2012102806, 03, HCOM, 015, 344N, 713W, 58, 958, XX, 34, NEQ, 0319, 0269, 0274, 0323, 0, 0, 41, 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 setting up the paths to the correct locations in your system, run the script using the command:

  atcfplot.sh ${YYYY} ${Basin}

This will start a GUI window and read in ATCF format track files *a${Basin}${SID}${YYYY}.dat* in *$rundir (${SID}* is the storm ID) for storms in year *${YYYY}* in the *${Basin}* basin.

For example, the user can use the command "*atcfplot.sh 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".

Chapter 9: HWRF Idealized Tropical Cyclone Simulation

9.1 Introduction

Initial conditions for the HWRF Idealized Tropical Cyclone case are specified using an idealized vortex superposed on a base state quiescent sounding. The default initial vortex has an intensity of 20 ms^{-1} and a radius of maximum winds of 90 km. To initialize the idealized vortex, a nonlinear balance equation in pressure-based sigma coordinates is solved within the rotated latitude–longitude E-grid framework.

The default initial ambient base state assumes a f-plane at the latitude of 12.5^{o}. The sea surface temperature is time-invariant and horizontally homogeneous, with the default set to 302 K. No land is used in the simulation domain.

The lateral boundary conditions used in the HWRF idealized simulation are the same as used in real data cases. This inevitably leads to some reflection when gravity waves emanating from the vortex reach the outer domain lateral boundaries.

The idealized simulation uses the operational HWRF triple-nested domain configuration with grid spacing at 27-, 9-, and 3-km. All the operational atmospheric physics, as well as the supported experimental physics options in HWRF, can be used in the idealized HWRF framework. The UPP (see Chapter 7) can be used to postprocess the idealized HWRF simulation output.

The setup of the idealized simulation requires the use of WPS to localize the domain *(geogrid.exe*) and to process GFS data for initial and boundary conditions (*ungrib.exe* and *metgrid.exe*). The initialization using WPS just provides a framework for the initial conditions, which are actually specified in *ideal.exe* to be composed of a quiescent environment with a prescribed vortex. The boundary conditions generated with WPS are also overwritten by *ideal.exe* to be consistent with the quiescent environment.

The initial base state temperature and humidity profile is prescribed in file *sound.d,* while the vortex properties are specified in *input.d***.** The latter file is also used to specify options for f-plane and ß-plane.

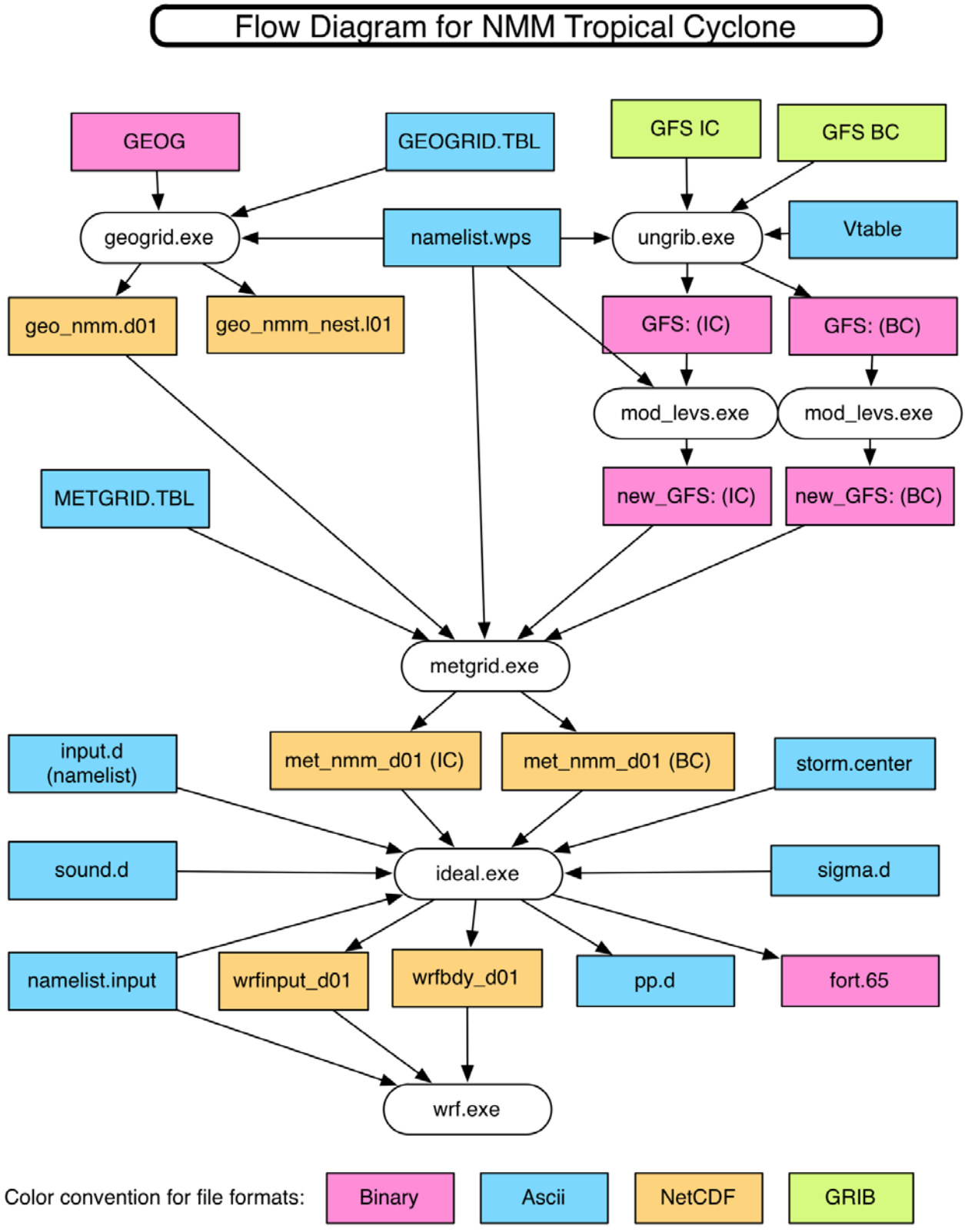


Figure 9-1. Flow diagram for the NMM idealized tropical cyclone capability.

9.2 How to Use HWRF for Idealized Tropical Cyclone Simulations

9.2.1 Source code

The HWRF V3.5a release is required, as the idealized capability is not available in previous releases. Only the WPS and WRFV3 components are required for the idealized tropical cyclone simulations. The UPP can be used for postprocessing. The other HWRF components do not need to be compiled. Please see Chapter 2 for instructions to compile the WPS, WRF and, if desired, UPP. Note that the executable file *wrf.exe* needed for the idealized simulation is not the same as the one needed for the simulation for real data. Therefore, users should follow the instructions specific for building the idealized *wrf.exe*. In this Users' Guide, we assume that the user will install HWRF in directory *${SCRATCH}/HWRF*.

9.2.2 Input files and datasets

Two GFS GRIB files are needed to provide a template for creating the initial and lateral boundary conditions. One of the GFS GRIB files should be the analysis valid at the same time of the desired HWRF initialization. The other GRIB file should be a forecast, with lead time equal to or greater than the desired HWRF simulation. The meteorological data in these files will not be used to initialize the simulation -- these files are for template purposes only.

As an example, files *0825012000000* and *0825512000000***,** are included in the tar file *http://www.dtcenter.org/HurrWRF/users/downloads/datasets/Idealized/hwrfv3.5a_idealized.tar.gz*.

Next the user must verify that all the input files below exist in *${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone*.

1. Namelist file for WPS: *namelist.wps*. Note that *geog_data_path* should be modified to point to the actual path of the geog data files.
2. Namelist file for WRF*: namelist.input.*
3. Vortex description file: *input.d*.
4. Sounding data: *sound.d*. Note that four sounding files are provided in *${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone* (*sound.d*, *sound_gfdl.d*, *sound_jordan.d*, and *sound_wet.d*). However, only the one named *sound.d* will be used. In order to use a different sounding, rename it to *sound.d.*
5. Vortex center file: *storm.center*.
6. Sigma file: *sigma.d.*

9.2.3 General instructions for running the executables

To perform the idealized simulation the following executables need to be run: *geogrid.exe*, *ungrib.exe*, *mod_levels.exe*, *metgrid.exe*, *ideal.exe*, and *wrf.exe*. Since the executables are compiled with distributed memory capability, many computing platforms require they be run on compute nodes. Instructions for running jobs on compute nodes can be found in Section 6 of Chapter 1.

The scripts and wrappers described in previous chapters for running HWRF using real data are not used for the idealized simulation. Since the workflow of the idealized simulation is fairly simple, the commands can be run manually.

9.2.4 Running *WPS* to create the initial and boundary conditions

The steps below outline the procedure to preprocess the data for the creation of initial and boundary conditions for the idealized simulation. It assumes that the run will be conducted in a working directory named *$workdir/wpsprd.*

1. Create and change into directory for running WPS
    a. *mkdir $workdir/wpsprd*
    b. *cd $workdir/wpsprd*
2. Run geogrid
    a. Make a directory for the geogrid table and change into it
        - *mkdir geogrid*
        - *cd geogrid*
    b. Link the geogrid table
        - *ln -fs ${SCRATCH}/HWRF/WPSV3/geogrid/GEOGRID.TBL.NMM ./GEOGRID.TBL*
    c. Copy the WPS namelist
        - *cd $workdir/wpsprd*
        - *cp ${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone/namelist.wps .*
    d. Edit *namelist.wps* to make sure *geog_data_path* points to the location of the WPS geographical data files
    e. Run executable *geogrid.exe* on the command line or submit it to a compute node or batch system
        - *${SCRATCH}/HWRF/WPSV3/geogrid.exe*
    f. Verify that the output files were created
        - *ls -l geo_nmm_nest.l01.nc geo_nmm.d01.nc*
3. Run ungrib
    a. Link the ungrib table
        - *ln -fs ${SCRATCH}/HWRF/WPSV3/ungrib/Variable_Tables/Vtable.GFS ./Vtable*
    b. Extract the two input GFS files
        - Download tarfile with GFS input data *http://www.dtcenter.org/HurrWRF/users/downloads/datasets/Idealized/hwrfv3.5a_idealized.tar.gz*
        - *tar xzvf hwrfv3.5a_idealized.tar.gz*
        - *gunzip 0825012000000.gz*

- *gunzip 0825512000000.gz*
- *ls -l 0825012000000 0825512000000*

c. Link the GFS files to the names expected by ungrib
- *${SCRATCH}/HWRF/WPSV3/link_grib.csh* 0825012000000 0825512000000

d. Run executable *ungrib.exe* on the command line or submitting it to a compute node or batch system
- *${SCRATCH}/HWRF/WPSV3/ungrib.exe*

e. Verify that the output files were created
- *ls -l GFS:2008-09-06_12 GFS:2008-09-11_12*

4. Run metgrid

a. Make a directory for the metgrid table and change into it
- *mkdir metgrid*
- *cd metgrid*

b. Link the metgrid table
- *ln -fs ${SCRATCH}/HWRF/WPSV3/metgrid/METGRID.TBL.NMM ./METGRID.TBL*

c. Run executable *mod_levels.exe* twice on the command line or submitting it to a compute node or batch system. This program is used to reduce the number of vertical levels in the GFS file. Only the levels listed in variable *press_pa* in *namelist.wps* will be retained.
- *cd $workdir/wpsprd*
- *${SCRATCH}/HWRF/WPSV3/util/mod_levs.exe GFS:2008-09-06_12 new_GFS:2008-09-06_12*
- *${SCRATCH}/HWRF/WPSV3/util/mod_levs.exe GFS:2008-09-11_12 new_GFS:2008-09-11_12*

d. Verify that the output files were created
- *ls -l new_GFS:2008-09-06_12 new_GFS:2008-09-11_12*

e. Run executable *metgrid.exe* on the command line or submitting it to a compute node or batch system
- *${SCRATCH}/HWRF/WPSV3/metgrid.exe*

f. Verify that the output files were created
- *ls -l met_nmm.d01.2008-09-06_12:00:00.nc met_nmm.d01.2008-09-11_12:00:00.nc*

9.2.5 Running *ideal.exe* and *wrf.exe*

The steps below outline the procedure to create initial and boundary conditions for the idealized simulation. It assumes that the run will be conducted in a working directory named *$workdir/wrfprd.*

1. Create and change into directory for running ideal and real
    a. mkdir $workdir/wrfprd
    b. cd $workdir/wrfprd
2. Run ideal
    a. Link WRF input files
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_ETAMPNEW_DATA ETAMPNEW_DATA*
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_GENPARM.TBL GENPARM.TBL*
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_LANDUSE.TBL LANDUSE.TBL*
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_SOILPARM.TBL SOILPARM.TBL*
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_VEGPARM.TBL VEGPARM.TBL*
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_tr49t67 tr49t67*
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_tr49t85 tr49t85*
        - *ln -fs ${SCRATCH}/HWRF/hwrf-utilities/parm/hwrf_tr67t85 tr67t85*
    b. Link the WPS files
        - *ln -fs $workdir/wpsprd/met_nmm* .*
        - *ln -fs $workdir/wpsprd/geo_nmm* .*
    c. Copy the idealized simulation input files
        - *cp ${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone/input.d .*
        - *cp ${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone/sigma.d .*
        - *cp ${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone/sound.d .*
        - *cp ${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone/storm.center .*
    d. Copy namelist input
        - *cp ${SCRATCH}/HWRF/WRFV3/test/nmm_tropical_cyclone/namelist.input .*
    e. Edit and modify files input.d, sound.d, if desired

- The sounding files provided have 30 vertical levels. In order to use a sounding with different number of levels, it is necessary to modify the source code in *${SCRATCH}/HWRF/WRFV3/dyn_nmm/module_initialize_tropical_cyclone.F*. In subroutine *tem*, parameter *nv* should be modified from 30 to the number of levels in the sounding.
- File *storm.center* should not be altered to make sure the storm is located in the center of the inner nest
- File *sigma.d* should not be modified as it does not pertain to the vertical levels of the sounding or of the simulation. Rather, it defines the vertical levels used to create the initial vortex.

f. Run executable *ideal.exe* on the command line or submitting it to a compute node or batch system
- *${SCRATCH}/HWRF/WRFV3/main/ideal.exe*

g. Verify that the output files were created
- ls -s *wrfinput_d01 wrfbdy_d01 fort.65*

5. Run WRF

a. Run executable *wrf.exe* on the command line or submitting it to a compute node or batch system
- ***${SCRATCH}****/HWRF/WRFV3/main/wrf.exe*
- Note that executable *wrf.exe* must have been created using the instructions for idealized simulations described in Chapter 2. The executable created for regular HWRF simulations that ingest real data should not be used to conduct idealized simulations.

b. Verify that the output files were created
- *ls -l wrfout_d01* wrfout_d02* wrfout_d03**

9.2.6 Using UPP to post-process the idealized tropical cyclone simulation output.

Once the WRF output (files *wrfout**) have been created in directory *$workdir/wrfprd*, the procedures outlined in Chapter 7 to run UPP using the wrappers and scripts can be used.

Appendix

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

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.

1. Storm Info (settings for SANDY 18L 2012 storm)		
Variable Name	**Description**	**Example**
START_TIME	HWRF initialization time	2012102806
START_TIME_MINUS6	Time 6 h before HWRF initialization	2012102800
FCST_LENGTH	Forecast length in hours	126
FCST_INTERVAL	Time between consecutive HWRF cycles	6
STORM_NAME	Storm name issued by NHC	SANDY
SID	Storm identification	18L
BASIN	Storm ocean basin (AL, EP, CP, WP, IO)	AL

2. Component options		
Variable Name	Description	Example
HWRF		
IO_FMT	IO format (1 for binary, 2 for NetCDF)	2
RUN_PREP_HYB	Logical variable to determine the use of prep_hybrid to process spectral global model data in hybrid native coordinates for HWRF initialization	T
GSI_USE_SAT	Not used	
UPP_PROD_SAT	Logical variable to determine if synthetic satellite images will be produced during postprocessing	F
GFS_DATA_MODE	Choice of variables present in global model input to HWRF. Using "REDUCED" speeds processing time	REDUCED
use_extended_eastatl	Logical variable to determine use of extended East Atlantic POM-TC domain	F
BKG_MODE	Background (global model) source	GDAS
Domain option sizes		
ATMOS_DOMAINS	Number of atmospheric domains	3
NX1	Number of grid points in meridional direction	216

	in parent domain	
NY1	Number of grid points in zonal direction in parent domain	432
DXX	Parent grid meridional spacing (deg)	0.18
DYY	Parent grid zonal spacing (deg)	0.18
DTT	Parent grid dynamic timestep (s)	45
VERT_LEV	Number of vertical levels	43
IO_SERVERS	Loginal variable to determine use of I/O servers	YES
IOSRV_PERGRP	Number of I/O servers per group	4
IOSRV_GROUPS	Number of I/O server groups	4
GSI Options		
RUN_GSI	Logical variable to determine if GSI will be run	T
RUN_GSI_WRFINPUT	Logical variable to determine if GSI will be run in the parent domain	T
RUN_GSI_WRFGHOST	Logical variable to determine if GSI will be run in the ghost domain	F
INNER_CORE_DA	Variable to determine inner core data assimilation will be performed (0=no; 1=assimilate TDR; 2=assimilate TDR and other inner core data – unsupported)	1
FGAT	Array containing number of hours, counting from the HWRF initialization time, for which global model data will be preprocessed for use FGAT	-3,0,3
GSI_USE_RAD	Logical variable to determine if satellite radiances will be assimilated	F
GSI_ENS_REG	Logical variable to determine if global ensemble data should be used	T
GSI_ENS_REG_SIZE	Number of members in ensemble used by GSI for hybrid data assimilation	80
GSI_ENS_REG_OPT	Variable to indicate to GSI the ensemble source type (1=GEFS)	1

3. Paths for source, script, and output directories		
HWRF_SRC_DIR	Path to the HWRF source code	${SCRATCH}/HWRF/sorc/
HWRF_SCRIPTS	Path to low-level scripts	${HWRF_SRC_DIR}/hwrf-utilities/scripts
HWRF_OUTPUT_DIR	Path to the HWRF output	${HWRF_SRC_DIR}/../results

4. HWRF data and fixed file paths		
DATA		
HWRF_DATA_DIR	Path to the HWRF input data sets	/tmp/datasets
CYCLE_DATA	Path to the previous cycle output	${HWRF_OUTPUT_DIR}/${SID}/${START_TIME_MINUS6}
DOMAIN_DATA	Path to the previous cycle output	${HWRF_OUTPUT_DIR}/${SID}/${START_TIME}
GFS_DIR	Path to the toplevel GFS input data	${HWRF_DATA_DIR}/GFS
GFS_SPECTRAL_DIR	Path to the GFS spectral input data	${GFS_DIR}/spectral
GFS_GRIDDED_DIR	Path to the GFS gridded input data	${GFS_DIR}/gridded
GFS_OBS_DIR	Path to the GFS observational input data	${GFS_DIR}/obs
GEFS_DIR	Path to the toplevel GEFS input data	${HWRF_DATA_DIR}/GEFS
GEFS_ENS_FCST_DIR	Path to the GEFS input data	${GEFS_DIR}/spectral
GDAS_DIR	Path to the toplevel GDAS input data	${HWRF_DATA_DIR}/GDAS
GDAS_SPECTRAL_DIR	Path to the GDAS spectral input data	${GDAS_DIR}/spectral
GDAS_GRIDDED_DIR	Path to the GDAS gridded input data	${GDAS_DIR}/gridded
GDAS_OBS_DIR	Path to the GDAS observational input data	${GDAS_DIR}/obs
TDR_OBS_DIR	Path to the TDR observations	${HWRF_DATA_DIR}/TDR
TCVITALS	Path to the TCVitals files	${HWRF_DATA_DIR}/Tcvitals
FIXED		
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
CRTM_FIXED_DIR	Path to the CRTM fix files used by UPP	${HWRF_DATA_DIR}/fix/upp
GEOG_DATA_PATH	Path to geographical fix data	${HWRF_DATA_DIR}/wps_geog
GSI		
GSI_FIXED_DIR	Path to the GSI fix	${HWRF_DATA_DIR}/fix/gsi
GSI_CRTM_FIXED_DIR	Path to the CRTM fix files used by GSI	${HWRF_DATA_DIR}/fix/gsi/CRTM_Coefficients

5. Runtime specific settings		
MPIRUN	Command used to run parallel code	mpiexec
WRF_ANAL_CORES	Number of cores to run WRF analysis	12
WRF_GHOST_CORES	Number of cores to run WRF ghost	12
HWRF_FCST_CORES	Number of cores to run the HWRF	202

	forecast	
GEOGRID_CORES	Number of cores to run geogrid	12
METGRID_CORES	Number of cores to run metgrid	1
REAL_CORES	Number of cores to run real	1
GSI_CORES	Number of cores to run GSI	100
UNI_CORES	Number of cores to run unipost	12
PREP_HYB_CORES	Number of cores to run prep_hybrid	1
ATCFNAME	ATCF identifier for HWRF forecast	HCOM

6. Path to GrADS tools		
GRADS_BIN	Path to GrADS	/apps/grads/2.0.1/bin
GADDIR	Path to GrADS binaries	/apps/grads/lib/

7. Component paths		
WRF_ROOT	Path to WRF source code	${HWRF_SRC_DIR}/WRFV3
WPS_ROOT	Path to WPS source code	${HWRF_SRC_DIR}/WPSV3
UPP_ROOT	Path to UPP source code	${HWRF_SRC_DIR}/UPP
HWRF_UTILITIES_ROOT	Path to HWRF Utilities source code	${HWRF_SRC_DIR}/hwrf-utilities
GSI_ROOT	Path to GSI source code	${HWRF_SRC_DIR}/GSI
POMTC_ROOT	Path to POM-TC source code	${HWRF_SRC_DIR}/pomtc
TRACKER_ROOT	Path to GFDL vortex tracker source code	${HWRF_SRC_DIR}/gfdl-vortextracker
COUPLER_ROOT	Path to coupler source code	${HWRF_SRC_DIR}/ncep-coupler

NOAA Technical Memorandum OAR GSD-43

COMMUNITY HWRF USERS GUIDE V3.5A

Ligia Bernardet[1]
Shaowu Bao[2]
Richard Yablonsky[3]
Don Stark[4]
Timothy Brown[1]

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

[2] Cooperative Institute for Research in Environmental Sciences (CIRES) and currently affiliated with S. Carolina Coastal University and NOAA/ESRL/GSD

[3] University of Rhode Island

[4] Developmental Testbed Center and NCAR/RAL/JNT

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