

NOAA Technical Memorandum OAR GSD-46

doi:10.7289/V5SJ1HMD



---

## **COMMUNITY HWRF USERS GUIDE V3.7A**

**May 2015**

### **THE DEVELOPMENTAL TESTBED CENTER**

Mrinal K. Biswas  
Laurie Carson  
Christina Holt  
Ligia Bernardet  
Timothy Brown  
Shaowu Bao  
Richard Yablonsky

Earth System Research Laboratory  
Global System Division  
Boulder, Colorado  
May 2015

---



## **COMMUNITY HWRF USERS GUIDE V3.7A**

Mrinal K. Biswas<sup>1</sup>

Laurie Carson<sup>1</sup>

Christina Holt<sup>2</sup>

Ligia Bernardet<sup>2</sup>

<sup>1</sup> National Center for Atmospheric Research and Developmental Testbed Center

<sup>2</sup> Cooperative Institute for Research in Environmental Sciences (CIRES), Developmental Testbed Center and NOAA/ESRL/GSD

### Acknowledgements

Timothy P. Brown  
Research Computing/CU

Shaowu Bao  
Coastal Carolina University

Richard Yablonsky  
AIR Worldwide



**UNITED STATES  
DEPARTMENT OF COMMERCE**

**Penny Pritzker  
Secretary**

NATIONAL OCEANIC AND  
ATMOSPHERIC ADMINISTRATION

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

Office of Oceanic and  
Atmospheric Research

Craig N. McLean  
Assistant Administrator

# Community HWRF Users' Guide v3.7a

May 2015

Mrinal K. Biswas, Laurie Carson

*National Center for Atmospheric Research and Developmental Testbed Center*

Christina Holt, Ligia Bernardet

*NOAA/ESRL Global Systems Division , Developmental Testbed Center and CIRES/CU*

Acknowledgements

Timothy P. Brown

*Research Computing/CU*

Shaowu Bao

*Coastal Carolina University*

Richard Yablonsky

*AIR Worldwide*



# Acknowledgement

If significant help was provided via the HWRF helpdesk for work resulting in a publication, please acknowledge the Developmental Testbed Center Hurricane Team.

For referencing this document please use:

Biswas, M. K., L. Carson, C. Holt, L. Bernardet, 2015: Community HWRF Users' Guide V3.7a, 152 pp. doi:10.7289/V5SJ1HMD [available online at [http://www.dtcenter.org/HurrWRF/users/docs/users\\_guide/HWRF\\_v3.7a\\_UG.pdf](http://www.dtcenter.org/HurrWRF/users/docs/users_guide/HWRF_v3.7a_UG.pdf)]

# Contents

<b>Preface</b>	<b>vii</b>
<b>1. Introduction</b>	<b>1</b>
1.1. HWRF System Overview . . . . .	1
1.2. HWRF Development and Support . . . . .	4
1.3. HWRF Source Code Directory Structure . . . . .	5
1.3.1. HWRF Utilities Programs and Scripts . . . . .	6
1.3.2. MPIPOM-TC Ocean Model . . . . .	7
1.3.3. NCEP Coupler . . . . .	7
1.3.4. GFDL Vortex Tracker . . . . .	8
1.3.5. WRFV3 – Atmospheric Model . . . . .	8
1.3.6. WPSV3 – WRF Preprocessor . . . . .	9
1.3.7. UPP – Unified Post-Processor . . . . .	9
1.3.8. GSI – Gridpoint Statistical Interpolation . . . . .	9
1.3.9. HWRF Run . . . . .	9
<b>2. Software Installation</b>	<b>11</b>
2.1. Introduction . . . . .	11
2.2. Obtaining the HWRF Source Code . . . . .	12
2.3. Setting up HWRF . . . . .	12
2.4. System Requirements, Libraries, and Tools . . . . .	13
2.4.1. Compilers . . . . .	13
2.4.2. netCDF, pnetCDF, and MPI . . . . .	14
2.4.3. LAPACK and BLAS . . . . .	15
2.5. Included Libraries . . . . .	15
2.5.1. Component Dependencies . . . . .	16
2.6. Building WRF-NMM . . . . .	17
2.6.1. Set Environment Variables . . . . .	17
2.6.2. Configure and Compile WRF-NMM . . . . .	18
2.6.3. Configure and Compile: Idealized Tropical Cyclone WRF-NMM . . . . .	19
2.7. Building HWRF-utilities . . . . .	21
2.7.1. Set Environment Variables . . . . .	21
2.7.2. Configure and Compile . . . . .	22

## Contents

2.8.	Building MPIPOM-TC . . . . .	24
2.8.1.	Set Environment Variables . . . . .	24
2.8.2.	Configure and Compile . . . . .	25
2.9.	Building GFDL Vortex Tracker . . . . .	26
2.9.1.	Set Environment Variables . . . . .	26
2.9.2.	Configure and Compile . . . . .	27
2.10.	Building the NCEP Coupler . . . . .	28
2.10.1.	Configure and Compile . . . . .	28
2.11.	Building WPS . . . . .	29
2.11.1.	Set Environment Variables . . . . .	29
2.11.2.	Configure and Compile . . . . .	29
2.12.	Building UPP . . . . .	31
2.12.1.	Set Environment Variables . . . . .	31
2.12.2.	Configure and Compile . . . . .	32
2.13.	Building GSI . . . . .	33
2.13.1.	Set Environment Variables . . . . .	34
2.13.2.	Configure and Compile . . . . .	34
<b>3.</b>	<b>Running HWRF</b>	<b>36</b>
3.1.	HWRF Scripts Overview . . . . .	36
3.2.	Defining an Experiment . . . . .	37
3.2.1.	Standard Configuration Files . . . . .	37
3.3.	Input Data and Fix Directory Structure . . . . .	41
3.4.	Production Directory Structure . . . . .	47
3.5.	Scripts for Running HWRF . . . . .	49
3.5.1.	Submitting a Job . . . . .	49
3.6.	Running HWRF End-to-End . . . . .	50
3.6.1.	Editing <code>global_vars.sh</code> . . . . .	50
3.6.2.	Using Wrapper Scripts . . . . .	51
3.7.	Operational HWRF in Each Ocean Basin . . . . .	52
3.7.1.	Atlantic Basin . . . . .	52
3.7.2.	Eastern Pacific Basin . . . . .	53
3.7.3.	All Other Basins . . . . .	54
3.8.	Running HWRF in Non-operational Configurations . . . . .	55
3.8.1.	Running Coupled/Uncoupled Forecast . . . . .	55
3.8.2.	Running with Optional GSI . . . . .	57
3.8.3.	Running without Vortex Initialization . . . . .	58
3.8.4.	Running without Spectral Files (GRIB Only) . . . . .	59
3.8.5.	Running with 27/9/3-km Configuration . . . . .	59
3.8.6.	Running with 43 Vertical Levels . . . . .	61
<b>4.</b>	<b>HWRF Preprocessing System</b>	<b>62</b>
4.1.	Introduction . . . . .	62
4.2.	Scripts . . . . .	65
4.2.1.	Overview of <code>exhwrflaunch.py</code> . . . . .	68
4.2.2.	Overview of the Init Scripts: <code>exhwrflaunch.py</code> and Wrappers . . . . .	70
4.2.3.	Overview of Initialization Modules . . . . .	70

<b>5. Vortex Relocation</b>	<b>80</b>
5.1. Introduction	80
5.2. Scripts	84
5.2.1. Overview of <code>exhwrf_relocate.py</code>	84
5.2.2. Overview of the Relocate Modules	84
<b>6. Data Assimilation</b>	<b>92</b>
6.1. Introduction	92
6.2. Scripts	93
6.2.1. Overview of <code>exhwrf_bufrprep.py</code>	94
6.2.2. Overview of the Bufrprep Module	94
6.2.3. Overview of <code>exhwrf_gsi.py</code>	95
6.2.4. Overview of the GSI Module	95
<b>7. Merge</b>	<b>97</b>
7.1. Introduction	97
7.2. Scripts	97
7.2.1. Overview of <code>exhwrf_merge.py</code>	97
7.2.2. Overview of Merge Module	98
<b>8. Ocean Initialization for MIPOM-TC</b>	<b>100</b>
8.1. Introduction	100
8.2. Scripts	100
8.2.1. Overview of <code>exhwrf_ocean_init.py</code>	100
8.2.2. Overview of Ocean Init Modules	101
8.3. MIPOM-TC Diagnostics	105
8.4. User-provided Datasets for MIPOM-TC Initialization	105
<b>9. Forecast Model</b>	<b>107</b>
9.1. Introduction	107
9.2. Scripts	107
9.2.1. Overview of <code>exhwrf_forecast.py</code>	108
9.2.2. Overview of the Forecast Module	108
<b>10. HWRF Post-Processor</b>	<b>112</b>
10.1. Introduction	112
10.2. Scripts	112
10.2.1. Overview of <code>exhwrf_unpost.py</code>	113
10.2.2. Overview of <code>exhwrf_post.py</code>	113
10.2.3. Overview of UPP Python Modules	113
<b>11. Forecast Products</b>	<b>115</b>
11.1. Introduction	115
11.2. Scripts	116
11.2.1. Overview of <code>exhwrf_products.py</code>	117
11.2.2. Additional Tracking Utilities	123
11.3. How to Plot the Tracker Output Using <code>ATCF_PLOT</code>	125

*Contents*

<b>12. HWRF Idealized Tropical Cyclone Simulation</b>	<b>127</b>
12.1. Introduction . . . . .	127
12.2. How to Use HWRF for Idealized Tropical Cyclone Simulations . . . . .	128
12.2.1. Source Code . . . . .	128
12.2.2. Input Files and Datasets . . . . .	128
12.2.3. General Instructions for Running the Executables . . . . .	129
12.2.4. Running WPS to Create the ICs and LBCs . . . . .	129
12.2.5. Running <code>ideal.exe</code> and <code>wrf.exe</code> . . . . .	131
<b>A. Example of Computational Resources</b>	<b>133</b>
<b>B. Example HWRF Namelist</b>	<b>135</b>
<b>C. Additional GFDL Tracker Information</b>	<b>139</b>

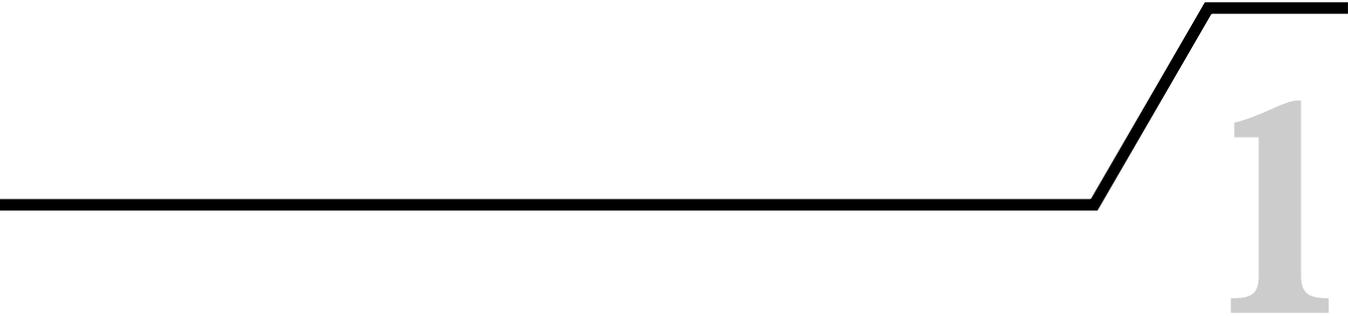
# Preface

## Meaning of typographic changes and symbols

Table 1 describes the type changes and symbols used in this book.

<b>Typeface or Symbol</b>	<b>Meaning</b>	<b>Example</b>
<i>AaBbCc123</i>	The names of commands, files, and directories;	Edit your <code>.bashrc</code>
<b>AaBbCc123</b>	on-screen computer output What you type, contrasted with on-screen computer output	Use <code>ls -a</code> to list all files. <code>host\$ You have mail!.</code> <code>host\$ <b>su</b></code>
<i>AaBbCc123</i>	Command line placeholder: replace with a real name or value	To delete a file, type <code>rm <i>filename</i></code>

Table 1: Typographic Conventions



# 1

## 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 v3.7a and this Users' Guide contain the capabilities of, and information regarding, the operational 2015 implementation of HWRF.

The HWRF model is a primitive-equation, non-hydrostatic, coupled atmosphere-ocean model with an atmospheric component that employs 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  $80^{\circ} \times 80^{\circ}$  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)/Joint Typhoon Warning Center (JTWC) forecast of the 72-h position, if available. The middle nest domain, of about  $12^{\circ} \times 12^{\circ}$ , and the inner nest domain, of about  $7.1^{\circ} \times 7.1^{\circ}$ , move along with the storm using two-way interactive nesting. The stationary parent domain has a grid spacing of  $0.135^{\circ}$  (about 18 km) while the middle nest grid spacing is  $0.045^{\circ}$  (about 6 km) and the inner nest grid spacing is  $0.015^{\circ}$  (about 2 km). The dynamic time steps are 38, 12.66, and 4.22 s, respectively, for the parent, middle nest, and inner nest domains. The model top and number of vertical levels used in operations is dependent on the basin. In the north Atlantic (AL), north Eastern Pacific (EP), and north Central Pacific (CP) basins, model top is 2 hPa and 61 levels are used. In all other basins, model top is 50 hPa and 43 levels are used. The system is flexible so that different model tops and number of vertical levels can be used. Additionally, backwards

## 1. Introduction

compatibility has been maintained such that lower-resolution configurations (i.e., 27/9/3 km) are still supported.

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-Aligo 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 employs an improved air-sea momentum flux parameterization in strong wind conditions, and the Noah land surface model. Radiation effects are evaluated by the RRTMG scheme, which includes diurnal variations and interactive effects of clouds. The HWRF physics operational suite also includes parameterization of dissipative heating.

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 is used. Horizontal diffusion is based on a 2nd order Smagorinsky-type, for more details see the HWRF Scientific Documentation at <http://dtcenter.org/HurrWRF/users/>.

HWRF is now running for all global basins in operations at NCEP. This has led NCEP to choose different configurations of HWRF for different global basins. The default configuration of HWRF is based on the full capabilities of HWRF to run for the AL and EP basins, with reduced complexity in all other basins. The following discussion outlines the Atlantic configuration, while section 3.8 describes alternate configurations for other basins.

Model initialization is comprised of both a vortex improvement procedure and data assimilation. The NCEP Global Forecast System (GFS) analysis is used to generate the initial conditions (ICs) for the hurricane model parent domain in the operational configuration. On the inner 6-km and 2-km nests, the NCEP Global Data Assimilation System (GDAS) 6-hour forecast initialized 6 hours prior to the HWRF analysis is interpolated to the appropriate grid and is used as the first guess.

The analysis is modified by first separating the vortex and environment fields of the respective first guess on each domain, i.e. the GFS vortex is separated from the environment on the parent domain, the GDAS vortex is removed from the environment on the inner domains. A new vortex is then incorporated onto the environment field. The new vortex that is added to the environment depends on the observed intensity of the cyclone, and on the existence of a previous HWRF forecast. The new vortex may derive from a bogus procedure, from the 6-h forecast of the HWRF model initialized 6-h previously (referred to as cycling), or from the GDAS 6-h forecast initialized 6 hours previously. In any case, the vortex is modified so that the initial storm position, structure, and intensity conform to the storm message. Cycling the vortex from NHC areas of investigation (invests) to numbered storms is supported.

The first guess with an improved vortex is modified using the HWRF Data Assimilation System (HDAS) by ingesting observations in a three-dimensional (3D) hybrid ensemble-variational (VAR) data assimilation system called Gridpoint Statistical Interpolation (GSI). HWRF assimilates conventional observations, reconnaissance dropsondes, Tail Doppler Radar (TDR), and satellite observations. The information for the flow-dependent back-

## 1. Introduction

ground error covariance is obtained from the GFS ensemble. HWRF assimilates conventional observations, reconnaissance dropsondes, TDR, and satellite observations. Starting in 2015, when TDR data are present, the operational HWRF uses a 40-member HWRF ensemble instead of the GFS ensemble. Due to the complexity and high use of computational resources to generate the HWRF ensemble, the capability is not supported as part of the HWRF v3.7a public release. The assimilation of TDR data using HWRF v3.7a can be performed using the GFS ensemble. Satellite observations utilized by HWRF include radiances from infrared instruments (HIRS, AIRS, IASI, and GOES sounders) and microwave instruments (AMSU-A, MHS, and ATMS), satellite-derived wind, and Global Positioning System (GPS) radio occultation bending angle. First Guess at Appropriate Time (FGAT) is used to ensure that GSI uses innovations calculated by comparing observations with corresponding model analysis fields valid at the time when the observations were collected. The GFS forecast fields are used to provide lateral boundary conditions every 6 hours during the forecast.

The Community HWRF model can be used in atmosphere-ocean coupled mode for all oceanic basins. The MPIPOM-TC ocean model implements Message Passing Interface (MPI) to run a parallel version of 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 all basins, MPIPOM-TC is run in three dimensions with  $1/12^\circ$  (approximately 9-km) horizontal grid spacing. The MPIPOM-TC is configured with 23 vertical levels. The default HWRF is configured to run in coupled mode only in the AL and EP basins. However, coupling can be enabled with HWRF v3.7a in any basin.

The MPIPOM-TC is initialized in any basin 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.

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 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 nine minutes.

HWRF is suitable for use in tropical applications including real-time NWP, forecast research, physics parameterization research, air-sea coupling research, and teaching. Additionally, the capability to perform idealized tropical cyclone simulations is included. The HWRF system support to the community by the Developmental Testbed Center (DTC) includes the following four main modules:

- HWRF atmospheric components
  - WRF-NMM (which has tropical physics schemes and a vortex-following moving nest)
  - WRF Preprocessing System (WPS)

## 1. Introduction

- Vortex initialization
- GSI
- Unified Post-Processor (UPP)
- GFDL vortex tracker
- HWRF oceanic components
  - MIPOM-TC model
  - Ocean initialization
- Atmosphere-Ocean Coupler
- HWRF Run Module

The run module includes scripts in the Python language. The NCEP 2015 operational implementation of HWRF and the HWRF v3.7a are almost identical with the exception that starting in 2015, NCEP runs an HWRF ensemble for the assimilation of TDR data – a capability that is not currently supported to the community. Due to the novel nature of these scripts, users should expect some additions and changes in the upcoming months. Those will be posted in the Frequently-Asked-Questions (FAQ) or Known Issues sections of the DTC website at <http://dtcenter.org/HurrWRF/users>.

## 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). An 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. Advanced users interested in contributing to the HWRF system are encouraged to visit the HWRF Developers page at <http://www.dtcenter.org/HurrWRF/developers/index.php> to get more information on the protocols of HWRF development.

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

The HWRF modeling system software is in the public domain and is freely available for community use. Information about obtaining the codes, datasets, documentation, and tutorials can be found at <http://www.dtcenter.org/HurrWRF/users> and in the following chapters of this Users' Guide. Please refer to [http://www.dtcenter.org/HurrWRF/users/support/index\\_help.php](http://www.dtcenter.org/HurrWRF/users/support/index_help.php) on how to contact HWRF helpdesk.

## 1.3 HWRF Source Code Directory Structure

The HWRF system source code has the following nine components:

- WRF Atmospheric Model
- WPS
- UPP
- GSI
- HWRF Utilities
- MPIPOM-TC
- GFDL Vortex Tracker
- NCEP Atmosphere-Ocean Coupler
- HWRF Run Component

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

- `HWRF_v3.7a_WRFV3.tar.gz`
- `HWRF_v3.7a_WPSV3.tar.gz`
- `HWRF_v3.7a_UPP.tar.gz`
- `HWRF_v3.7a_GSI.tar.gz`
- `HWRF_v3.7a_hwrf-utilities.tar.gz`
- `HWRF_v3.7a_gfdl-vortextracker.tar.gz`
- `HWRF_v3.7a_ncep-coupler.tar.gz`
- `HWRF_v3.7a_pomtc.tar.gz`
- `HWRF_v3.7a_hwrfrun.tar.gz`

First expand `HWRF_v3.7a_hwrfrun.tar.gz` in a user-defined HWRF top directory. Once completed, change directory to `${SCRATCH}/hwrfrun/sorc` to expand the remaining tar files. The following directories should be present once all files are expanded:

- `WRFV3` - Weather Research and Forecasting model
- `WPSV3` - WRF Preprocessing System
- `UPP` - Unified Post-Processor
- `GSI` - Gridpoint Statistical Interpolation hybrid ensemble 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 MPIPOM

For the remainder of this document, we assume that the tar files have been expanded under `${SCRATCH}/hwrfrun/sorc`, where `${SCRATCH}` is an environment variable that describes the location of the directory in which you installed the HWRF components.

The directory trees for these nine 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.

### 1.3.1 HWRF Utilities Programs and Scripts

```

hwrf-utilities/
├── arch/ ..... architecture compiling options
├── clean..... script to clean created files and executables
├── compile..... script to compile component
├── configure ..... script to create the configure file
├── exec/..... executables
├── graphics/
├── libs/ ..... libraries: blas, sp, sfcio, bacio, w3, and bufr
├── makefile..... top-level makefile
├── pure-openmp.inc
├── tools/..... source code for tools to run the HWRF system
│   ├── Makefile..... makefile for ocean model code
│   ├── bufr_remorest/
│   ├── grbindex/
│   ├── hwrf_afos/
│   ├── hwrf_atcf_to_stats/
│   ├── hwrf_aux_rw/
│   ├── hwrf_bdy_update/
│   ├── hwrf_binary_grads/
│   ├── hwrf_bin_io/
│   ├── hwrf_blend_gsi/
│   ├── hwrf_combinetrack/
│   ├── hwrf_gettrk/
│   ├── hwrf_gridgenfine/
│   ├── hwrf_htcfstats/
│   ├── hwrf_netcdf_grads/
│   ├── hwrf_nhc_products/
│   ├── hwrf_prep_hybrid/
│   ├── hwrf_read_indi/
│   ├── hwrf_readtdrinfo/
│   ├── hwrf_supvit/
│   ├── hwrf_swath/
│   ├── hwrf_wrfout_newtime/
│   ├── mdate/
│   ├── mpi_example/
│   ├── mpiserial/
│   ├── ndate/
│   ├── nhour/
│   ├── serpoe/
│   ├── ships/
│   ├── wave_sample/
│   └── wgrib/
├── vortex_init..... source code for the HWRF vortex initialization
│   ├── Makefile..... makefile for vortex_init code
│   ├── hwrf_anl_bogus/
│   └── hwrf_anl_cs/

```

## 1. Introduction

```
|_ hwrp_anl_step2/  
|_ hwrp_create_trak_fnl/  
|_ hwrp_create_trak_guess/  
|_ hwrp_diffwrf_3dvar/  
|_ hwrp_pert_ct/  
|_ hwrp_set_ijstart/  
|_ hwrp_split/  
|_ interpolate/  
|_ merge_nest/
```

### 1.3.2 MPIPOM-TC Ocean Model

```
pomtc  
|_ arch/ ..... architecture compiling options  
|_ clean..... script to clean created files and executables  
|_ compile..... script to compile component  
|_ configure ..... script to create the configure file  
|_ makefile..... makefile for tools code  
|_ ocean_diag/..... software to plot ocean output  
|   |_ fortran/  
|   |_ matlab/  
|   |_ Makefile/  
|_ ocean_exec/..... ocean model executables  
|_ ocean_init/..... source code for generating ocean model initial conditions  
|   |_ Makefile..... makefile for the ocean initialization code  
|   |_ date2day/  
|   |_ day2date/  
|   |_ fbtr/  
|   |_ gdm3/  
|   |_ getsst/  
|   |_ hycu/  
|   |_ ncda/  
|   |_ idel/  
|   |_ pom/  
|   |_ sharp_mcs_rf_l2m_rmy5/  
|   |_ tran/  
|_ ocean_main/..... source code for the ocean forecast model  
|   |_ Makefile..... makefile for the ocean model code  
|   |_ pom/
```

### 1.3.3 NCEP Coupler

```
ncep-coupler/  
|_ arch/ ..... architecture compiling options  
|_ clean..... script to clean created files and executables  
|_ compile..... script to compile component  
|_ configure ..... script to create the configure file
```

## 1. Introduction

```
| cpl_exec/.....coupler executable  
| hwrw_wm3c/.....source code for the coupler  
| makefile.....top-level makefile
```

### 1.3.4 GFDL Vortex Tracker

```
gfdl-vortextracker/  
| arch/ ..... architecture compiling options  
| clean.....script to clean created files and executables  
| compile.....script to compile component  
| configure.....script to create the configure file  
| makefile.....top-level makefile  
| trk_exec/ ..... executables  
| trk_plot/.....plot scripts and data  
| trk_src/.....source code for the vortex tracker
```

### 1.3.5 WRFV3 – Atmospheric Model

```
WRFV3/  
| Makefile ..... makefile used to compile WRFV3  
| arch/ ..... architecture compiling options  
| chem_exp/.....chemistry module, not used by HWRF  
| clean.....script to clean created files and executables  
| compile.....script to compile component  
| configure.....script to create the configure file  
| 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  
| 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  
| Registry/.....WRFV3 Registry files  
| 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, not used by HWRF
```

Refer to the WRF-NMM Users' Guide for more information. The WRF-NMM Users' Guide is available online at:

<http://www.dtcenter.org/HurrWRF/users/docs/index.php>.

### 1.3.6 WPSV3 – WRF Preprocessor

```
WPSV3/
├── arch/ ..... architecture compiling options
├── clean..... script to clean created files and executables
├── compile..... script to compile component
├── configure ..... script to create the configure file
├── geogrid/..... source code for geogrid.exe
├── link_grib.csh..... script to link input GRIB files, used in idealized simulations
├── metgrid/..... source code for metgrid.exe
├── ungrib/..... source code for ungrib.exe
└── util/..... utility programs for WPSV3
```

### 1.3.7 UPP – Unified Post-Processor

```
UPP/
├── arch/ ..... architecture compiling options
├── bin/. location of executables .2 clean script to clean created files and executables
├── compile..... script to compile component
├── configure ..... script to create the configure file
├── makefile..... top level makefile
├── include/. include files .2 lib/. library files .2 parm/ parameter files to control
  UPP performed, not used by HWRF
├── scripts/..... sample scripts running UPP, not used by HWRF
└── src/..... UPP and dependent libraries source codes
```

### 1.3.8 GSI – Gridpoint Statistical Interpolation

```
GSI/
├── arch/ ..... architecture compiling options
├── clean..... script to clean created files and executables
├── compile..... script to compile component
├── configure ..... script to create the configure file
├── fix/..... fix files for GSI
├── include/
├── lib/
├── makefile..... top level makefile
├── run/..... executables
├── src/..... source codes
│   ├── Makefile
│   ├── libs/ ..... dependent libraries
│   └── main/ ..... GSI source code
└── util/..... utilities, not used by HWRF
```

### 1.3.9 HWRF Run

```
hwrfrun/
```

## 1. Introduction

- |— nwport/..... extra utilities for HWRF used in operations.
- |— parm/ ..... files to configure HWRF experiment
- |— scripts/ ..... Python scripts to run the HWRF components
- |— src/..... empty directory where HWRF components' code should be placed
- |— ush/..... Python modules
  - |— hwrf/ ..... HWRF-specific Python modules
  - |— produtils/..... HWRF-independent Python modules for generalization of platforms and systems
- |— wrappers/..... sh wrapper scripts used to run the Python scripts



# 2

## Software Installation

### 2.1 Introduction

The DTC community HWRF system, which is based on the NOAA operational HWRF, consists of nine components:

- WRF Atmospheric Model
- WPS
- UPP
- GSI
- HWRF-utilities
- MPIPOM-TC
- GFDL Vortex Tracker
- NCEP Atmosphere-Ocean Coupler
- HWRF Run

The first three of these components are the traditional WRF components: Weather Research and Forecasting model (WRF), WRF Preprocessing System (WPS), and Unified Post-Processor (UPP). Gridpoint Statistical Interpolation (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 with a description of 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 explains how to build WRF-NMM for HWRF. The remaining sections are devoted to building each of the remaining components of the HWRF system.

## 2. Software Installation

### 2.2 Obtaining the HWRF Source Code

The HWRF hurricane system consists of nine components. All of these are available from the HWRF website,

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

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.

To download HWRF, select the "Download" and "HWRF System" tabs on the left vertical menu of the HWRF Users' Website. New users must first register before downloading the source code. Returning users need only provide their registration email address. A successful download produces nine tar files:

- `HWRF_v3.7a_WRFV3.tar.gz`
- `HWRF_v3.7a_WPSV3.tar.gz`
- `HWRF_v3.7a_UPP.tar.gz`
- `HWRF_v3.7a_GSI.tar.gz`
- `HWRF_v3.7a_hwrf-utilities.tar.gz`
- `HWRF_v3.7a_gfdl-vortextracker.tar.gz`
- `HWRF_v3.7a_ncep-coupler.tar.gz`
- `HWRF_v3.7a_pomtc.tar.gz`
- `HWRF_v3.7a_hwrfmun.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 HWRF

Although the HWRF scripts may be modified for any directory structure, in this discussion, we assume 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 will typically need to be located on a computer's "scratch" or "work" partition.

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. The User should first unpack `hwrfmun`, which will create a directory `hwrfmun/` in `${SCRATCH}`. Then within `${SCRATCH}/hwrfmun/src/` directory, unpack the remaining tar files.

## 2. Software Installation

Once unpacked, there should be eight source directories in `src/`.

- `WRFV3` – Weather Research and Forecasting model
- `WPSV3` – WRF Preprocessing System
- `UPP` – Unified Post-Processor
- `GSI` – Gridpoint Statistical Interpolation
- `hwrf-utilities` – Vortex initialization, utilities, tools, and supplemental libraries
- `gfdl-vortextracker` – Vortex tracker
- `ncep-coupler` – Ocean/atmosphere coupler
- `pomtc` – Tropical cyclone version of MIPOM

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 Python modules, which runs 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 scripts, the build system relies on use of the Perl scripting language, along with 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+
  - if netCDF v4.1+ is used, HDF5 and SZIP libs may also be required
- LAPACK and BLAS
- Python
- Parallel-netCDF
- PNG
- JasPer
- zlib

Because these tools and libraries are typically the purview of system administrators to install and maintain, they are considered 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

## 2. Software Installation

compiler versions. Typically the best results come from using the most recent version of a compiler. Please note: A PGI version of at least 13.2 is required to build the HWRF system. 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, pnetCDF, 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, pnetCDF, 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. 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 variables `NETCDF` and `PNETCDF` can be set to point to the location of the library.

For `csh/tcsh`, the path can be set with the following command:

```
setenv NETCDF path_to_netcdf_library/  
setenv PNETCDF path_to_pnetcdf_library/
```

For `bash/ksh`, the equivalent command is as follows:

```
export NETCDF=path_to_netcdf_library/  
export PNETCDF=path_to_pnetcdf_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.

## 2. Software Installation

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 machine's documentation before you ask the local system administrator.

### 2.4.3 LAPACK and BLAS

The LAPACK and BLAS libraries are open source mathematics libraries for solving 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 contains **most** of the routines needed by HWRF. For PGI these libraries are loaded automatically. Since the vendor 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 MPIPOM-TC, and the GFDL Vortex Tracker components, the HWRF-utilities component includes a number of libraries in the **hwr-utilities/libs/src/** directory. The following libraries are built automatically when the HWRF-utilities component is built:

- BACIO

## 2. Software Installation

- BLAS
- BUFR
- SFCIO
- SIGIO
- SP
- W3
- G2

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 MPIPOM-TC component requires the SFCIO, SP and W3 libraries. In addition, the local copy of the BLAS library is required when the ESSL library is not included with the compiler installation. This is because the vendor-supplied versions of BLAS are typically incomplete, and the local version supplements the vendor version. Typically this is for any system other than 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 that contain source code 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 MPIPOM-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 main system component dependency is as follows:

- WRF
  - WPS
  - UPP
  - GSI
  - HWRF-utilities
    - \* MPIPOM-TC (BLAS on Linux, sfcio, sp, w3)
    - \* GFDL vortex tracker (w3, bacio, G2)
- 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, 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. See section 2.6.3 for compiling WRF for ideal runs.

#### 2.6.1 Set Environment Variables

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 following commands:

```
setenv HWRF 1
setenv WRF_NMM_CORE 1
setenv WRF_NMM_NEST 1
setenv JASPERLIB path_to_jasper_library/
setenv JASPERINC path_to_jasper_includes/
setenv PNETCDF_QUILT 1
setenv WRFIO_NCD_LARGE_FILE_SUPPORT 1
```

Add the following command for IBM AIX builds using C-Shell:

```
setenv IBM_REDUCE_BUG_WORKAROUND 1
```

In Bash shell, use the following commands:

```
export HWRF=1
export WRF_NMM_CORE=1
export WRF_NMM_NEST=1
export JASPERLIB=path_to_jasper_library/
export JASPERINC=path_to_jasper_includes/
export PNETCDF_QUILT=1
export WRFIO_NCD_LARGE_FILE_SUPPORT=1
```

Add the following command for IBM AIX builds using Bash:

```
export IBM_REDUCE_BUG_WORKAROUND=1
```

## 2. Software Installation

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=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.

### 2.6.2 Configure and Compile WRF-NMM

To configure WRF-NMM, go to the top of the WRF directory (`cd ${SCRATCH}/hwrfrun/sorc/WRFV3`) and use the following command:

```
./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 63 options. For the HWRF system, only the distributed memory (`dmpar`) builds are valid. Therefore as an example, the acceptable Intel options (15, 20, 24, and 28) are shown below:

```
13. (serial) 14. (smpar) 15. (dmpar) 16. (dm+sm) INTEL (ifort/icc)
18. (serial) 19. (smpar) 20. (dmpar) 21. (dm+sm) INTEL (ifort/icc): Xeon (S...
22. (serial) 23. (smpar) 24. (dmpar) 25. (dm+sm) INTEL (ifort/icc): SGI MPT
26. (serial) 27. (smpar) 28. (dmpar) 29. (dm+sm) INTEL (ifort/icc): IBM POE
```

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.

To build the WRF-NMM component enter the following command:

```
./compile nmm_real
```

In general, it is good practice to save the standard out and error to a log file for reference. In the `csh/tcsh` shell this can be done with the following command:

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

For the `ksh/bash` shell use the following command:

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

## 2. Software Installation

In both cases, the standard out and the standard error are sent 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.

A successful compilation produces two executables listed below in the directory `main/`.

<code>real_nmm.exe</code>	WRF initialization
<code>wrf.exe</code>	WRF model integration

If a recompilation is necessary, a clean to remove all object files (except those in `external/`) should be completed first:

```
./clean
```

A complete clean is strongly recommended if the compilation failed, the Registry has been changed, or the configuration file is changed. To conduct a complete clean that removes all built files in all directories, as well as the `configure.wrf` use the "-a" option:

```
./clean -a
```

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

Complete details on building and running the WRF-NMM model are available in the WRF-NMM Users' Guide, which is available here:

<http://www.dtcenter.org/HurrWRF/users/docs/index.php>.

### 2.6.3 Configure and Compile: 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 2.6.1. 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 typing:

## 2. Software Installation

```
./clean -a
```

which 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`. Several other variables must also be set.

In C-Shell use the following 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
```

The following commands should be used for bash/ksh:

```
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/frun/sorc/WRFV3`) before issuing the following command:

```
./configure
```

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

For Linux architectures, there are currently 51 options. For the HWRF system, only the distributed memory (`dmpar`) builds are valid. Therefore as an example, the acceptable Intel options (15, 20, 24, 28) are shown below:

```
13. (serial) 14. (smpar) 15. (dmpar) 16. (dm+sm) INTEL (ifort/icc)
18. (serial) 19. (smpar) 20. (dmpar) 21. (dm+sm) INTEL (ifort/icc): Xeon (S...
22. (serial) 23. (smpar) 24. (dmpar) 25. (dm+sm) INTEL (ifort/icc): SGI MPT
26. (serial) 27. (smpar) 28. (dmpar) 29. (dm+sm) INTEL (ifort/icc): IBM POE
```

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
```

## 2. Software Installation

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

<code>ideal.exe</code>	WRF initialization
<code>wrf.exe</code>	WRF model integration

Note: The only compilation requirements for the idealized capability are WPS and WRF. If wanted, UPP may also be used. The components MPIPOM-TC and coupler, GSI, GFDL vortex tracker, 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 MPIPOM-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 most 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-run/src/WRFV3`. In `csh/tcsh`, the variables may be set with two commands:

```
setenv NETCDF /absolute_path_to_appropriate_netCDF_library/  
setenv WRF_DIR ${SCRATCH}/hwrf-run/src/WRFV3
```

For the `ksh/bash` shells, use the following two commands:

```
export NETCDF=/absolute_path_to_appropriate_netCDF_library/  
export WRF_DIR=${SCRATCH}/hwrf-run/src/WRFV3
```

It is crucial that the Fortran compiler used to build the libraries (Intel, PGI, XLF, etc.) be the same as the compiler used to compile the source code. Typically, this is only an issue in two situations, on Linux systems having multiple compilers installed, and on systems where there is a choice between building 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 `hwrp-utilities` directory, type the following command:

```
./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.hwrp` in the `hwrp-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`, compile the HWRF-utilities and save the build output to a log file:

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

For the `ksh/bash` shell, use the following command:

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

If the compilation is successful, it will create the following executables in the directory `exec/`:

<code>bufr_remorest.exe</code>	<code>grp_inddiag.exe</code>
<code>diffwrf_3dvar.exe</code>	<code>grp_inddiagnull.exe</code>
<code>grbindex.exe</code>	<code>grp_nameparse.exe</code>
<code>gridparse.exe</code>	<code>grp_statsin_domain.exe</code>
<code>grp_atcf_to_stats.exe</code>	<code>grp_statsin_domain_TI.exe</code>
<code>grp_getcenter.exe</code>	<code>grp_totaldiag.exe</code>
<code>grp_gridparse.exe</code>	<code>hwrp_afos.exe</code>
<code>grp_hwrp_atcf_intensity.exe</code>	<code>hwrp_anl_4x_step2.exe</code>
<code>grp_hwrp_atcf_tracks.exe</code>	<code>hwrp_anl_bogus_10m.exe</code>

## 2. Software Installation

```
hwrp_anl_cs_10m.exe          hwrp_old_gettrk.exe
hwrp_atcf_to_stats.exe      hwrp_pert_ctl.exe
hwrp_aux_rw.exe             hwrp_prep.exe
hwrp_bdy_update.exe         hwrp_read_indi_write_all.exe
hwrp_binary_grads.exe       hwrp_readtdrstmid.exe
hwrp_bin_io.exe             hwrp_readtdrtime.exe
hwrp_blend_gsi.exe          hwrp_readtdrtrack.exe
hwrp_change_prepbufr_qm_in_circle.exe hwrp_readtdrtrigger.exe
hwrp_change_prepbufr_qm_typ.exe hwrp_rem_prepbufr_typ_in_circle.exe
hwrp_combinetrack.exe       hwrp_split1.exe
hwrp_create_trak_fnl.exe     hwrp_supvit.exe
hwrp_create_trak_guess.exe   hwrp_swath.exe
hwrp_gridgenfine.exe        hwrp_swcorner_dynamic.exe
hwrp_htcfcstats.exe         hwrp_wrfout_newtime.exe
hwrp_inter_2to1.exe         mdate.exe
hwrp_inter_2to2.exe         mpi_example.exe
hwrp_inter_2to6.exe         mpiserial.exe
hwrp_inter_4to2.exe         ndate.exe
hwrp_inter_4to6.exe         nhour.exe
hwrp_merge_nest_4x_step12_3n.exe serpoe.exe
hwrp_metgrid_levels.exe     wave_sample.exe
hwrp_netcdf_grads.exe       wgrib.exe
hwrp_nhc_products.exe
```

In addition, it will create twelve 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
libg2.a — GRIB2 library
libhwrputil_i4r4.a — Miscellaneous data manipulation utilities
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 MPIPOM-TC ocean model. The configuration step for these components will require setting a path variable to point to the `hwrp-utilities/libs/` directory in the `HWRP-utilities` directory.

If a recompilation is necessary, a clean to remove all object files (except those in `external/`) should be completed first:

```
./clean
```

## 2. Software Installation

A complete clean is strongly recommended if the compilation failed, the Registry has been changed, or the configuration file is changed. To conduct a complete clean that removes all built files in all directories, as well as the `configure.hwrf`, use the "-a" option:

```
./clean -a
```

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 MPIPOM-TC

### 2.8.1 Set Environment Variables

The Tropical Cyclone version of the MPIPOM-TC requires four external libraries: SFCIO, SP, W3, and PNETCDF. On platforms that lack the ESSL mathematical libraries, typically anything other than IBM AIX machines, a fifth library (BLAS) is required. The first three of these libraries are located in the `hwrf-utilities/libs/` directory and should be available if the HWRF-utilities component has been built successfully. You must first build them before building MPIPOM-TC.

Set the library paths (assuming the directory structure proposed in section 2.3) using C-Shell:

```
setenv LIB_W3_PATH ${SCRATCH}/hwrfun/sorc/hwrf-utilities/libs/  
setenv LIB_SP_PATH ${SCRATCH}/hwrfun/sorc/hwrf-utilities/libs/  
setenv LIB_SFCIO_PATH ${SCRATCH}/hwrfun/sorc/hwrf-utilities/libs/  
setenv PNETCDF PATH_TO_PNETCDF
```

Similarly, the libraries can be set using the ksh/bash shell:

```
export LIB_W3_PATH=${SCRATCH}/hwrfun/sorc/hwrf-utilities/libs/  
export LIB_SP_PATH=${SCRATCH}/hwrfun/sorc/hwrf-utilities/libs/  
export LIB_SFCIO_PATH=${SCRATCH}/hwrfun/sorc/hwrf-utilities/libs/  
export PNETCDF=PATH_TO_PNETCDF
```

In addition to these libraries, MPIPOM-TC requires linear algebra routines from the BLAS library. When building MPIPOM-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 MPIPOM-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 fifth and final path must be set:

```
setenv LIB_BLAS_PATH ${SCRATCH}/hwrfun/sorc/hwrf-utilities/libs/
```

## 2. Software Installation

For the ksh/bash shells the path can be set similarly:

```
export LIB_BLAS_PATH=${SCRATCH}/hwrfrun/sorc/hwrf-utilities/libs/
```

### 2.8.2 Configure and Compile

Configure MPIPOM-TC for compilation from within the `pomtc/` directory:

```
./configure
```

The configure script checks the system hardware, and if the path variables are not set, asks for software paths to the W3, SP, SFCIO, and PNETCDF, 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)

The following options exist for Linux:

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.

Compile the MPIPOM-TC and save the build output to a log file with `csh/tcsh`:

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

Similarly, for `ksh`, use the following syntax:

```
./compile 2>&1 | tee ocean.log
```

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

```
gfdl_date2day.exe  
gfdl_day2date.exe  
gfdl_getsst.exe  
gfdl_sharp_mcs_rf_l2m_rmy5.exe  
hwrf_ocean_fcst.exe  
hwrf_ocean_init.exe  
pomprep_fbtr.xc
```

## 2. Software Installation

```
pomprep_gdm3.xc  
pomprep_hycu.xc  
pomprep_idel.xc  
pomprep_ncda.xc  
readsstuvhflux.xc  
transatl06prep.xc
```

If a recompilation is necessary, a clean to remove all object files should be completed:

```
./clean
```

A complete clean is strongly recommended if the compilation failed, the configuration file has been changed, or the configuration file is changed. To conduct a complete clean that removes all built files in all directories, as well as the `configure.pom`, use the `"-a"` option.

```
./clean -a
```

## 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 `hwrfrun/sorc/hwrf-utilities/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, set the library paths:

```
setenv LIB_W3_PATH ${SCRATCH}/hwrfrun/sorc/hwrf-utilities/libs/  
setenv LIB_BACIO_PATH ${SCRATCH}/hwrfrun/sorc/hwrf-utilities/libs/  
setenv LIB_G2_PATH ${SCRATCH}/hwrfrun/sorc/hwrf-utilities/libs/  
setenv LIB_Z_PATH SYSTEM_LOCATION  
setenv LIB_PNG_PATH SYSTEM_LOCATION  
setenv LIB_JASPER_PATH SYSTEM_LOCATION
```

Similarly, the syntax for the ksh/bash shell can be used:

```
export LIB_W3_PATH=${SCRATCH}/hwrfrun/sorc/hwrf-utilities/libs/  
export LIB_BACIO_PATH=${SCRATCH}/hwrfrun/sorc/hwrf-utilities/libs/  
export LIB_G2_PATH=${SCRATCH}/hwrfrun/sorc/hwrf-utilities/libs/  
export LIB_Z_PATH=SYSTEM_LOCATION  
export LIB_PNG_PATH=SYSTEM_LOCATION  
export LIB_JASPER_PATH=SYSTEM_LOCATION
```

where `SYSTEM_LOCATION` should be replaced with the full path to the installed library. On

## 2. Software Installation

many systems, these libraries reside in `/usr/lib` or `/usr/lib64`.

### 2.9.2 Configure and Compile

Configure the Vortex Tracker for compilation from within the `gfdl-vortextracker` directory:

```
./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, there are six options:

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.

Compile the vortex tracker and save the build output to a log file:

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

The command for the `ksh`/`bash` shell follows:

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

If the compilation was successful, three executables are created in the directory `trk_exec/`.

```
hwrf_gettrk.exe  
hwrf_tave.exe  
hwrf_vint.exe
```

If a recompilation is necessary, a clean to remove all object files should be completed:

```
./clean
```

A complete clean is strongly recommended if the compilation failed, the configuration file has been changed, or the configuration file is changed. To conduct a complete clean that

## 2. Software Installation

removes all built files in all directories, as well as the `configure.trk`, use the "-a" option:

```
./clean -a
```

### 2.10 Building the NCEP Coupler

#### 2.10.1 Configure and Compile

Configure the NCEP Coupler for compilation from within the `ncep-coupler/` directory:

```
./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`.

There are five `dmpar` options for Linux:

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.

Compile the coupler and save the build output to a log file.

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

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

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

If the compilation is successful, it will create the single executable `hwrwf_wm3c.exe` in the `cpl_exec/` directory.

If a recompilation is necessary, a clean to remove all object files should be completed:

```
./clean
```

A complete clean is strongly recommended if the compilation failed, the configuration file has been changed, or the configuration file is changed. To conduct a complete clean that removes all built files in all directories, as well as the `configure.cpl`, use the "-a" option:

## 2. Software Installation

```
./clean -a
```

### 2.11 Building WPS

#### 2.11.1 Set Environment Variables

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:

```
setenv WRF_DIR ${SCRATCH}/hwrfrun/sorc/WRFV3/
```

For bash/ksh, use the command that follows:

```
export WRF_DIR=${SCRATCH}/hwrfrun/sorc/WRFV3/
```

Further details on using the WPS to create HWRF input data can be found in Chapter 4 of the HWRF Users' Guide.

Complete details on building and running the WPS are available from the WRF-NMM Users' Guide, and can be downloaded from:

<http://www.dtcenter.org/HurrWRF/users/docs/index.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 40 listed options. The first 28 are the most relevant to HWRF, and are listed below. Select a "NO\_GRIB2" option if you do not want GRIB2 support.

1. Linux x86\_64, gfortran (serial)
2. Linux x86\_64, gfortran (serial\_NO\_GRIB2)
3. Linux x86\_64, gfortran (dmpar)

## 2. Software Installation

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.

Compile the coupler and save the build output to a log file:

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

For the `ksh`/`bash` shell, use the equivalent command:

```
./compile 2>&1 | tee wps.log
```

After issuing the compile command, a successful compilation of WPS produces the three symbolic links: `geogrid.exe`, `ungrib.exe`, and `metgrid.exe` in the `WPSV3/` directory, and several symbolic links in the `util/` directory:

```
avg_tsfc.exe  
calc_ecmwf_p.exe  
glprint.exe  
g2print.exe  
height_ukmo.exe  
int2nc.exe  
mod_levs.exe  
rd_intermediate.exe
```

## 2. Software Installation

If any of these links do not exist, check the compilation log file to determine what went wrong.

A complete clean is strongly recommended if the compilation failed or if the configuration file is changed. To conduct a complete clean that removes ALL build files, including the executables, libraries, and the `configure.wps`, use the "-a" option to clean:

```
./clean -a
```

For full details on the operation of WPS, see the WPS chapter of the WRF-NMM Users' 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 destaggers 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 `hwr-f-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 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.

Set up the environment for UPP:

## 2. Software Installation

```
setenv HWRF 1
setenv WRF_DIR ${SCRATCH}/hwrfrun/sorc/WRFV3/
setenv JASPERLIB SYSTEM_LOCATION_SO_FILE
setenv JASPERINC SYSTEM_LOCATION_HEADER_FILE
```

The syntax for bash/ksh is as follows:

```
export HWRF=1
export WRF_DIR=${SCRATCH}/hwrfrun/sorc/WRFV3/
export JASPERLIB=SYSTEM_LOCATION_SO_FILE
export JASPERINC=SYSTEM_LOCATION_HEADER_FILE
```

### 2.12.2 Configure and Compile

UPP uses a build mechanism similar to that used by the WRF model. Within the UPP directory type:

```
./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 10 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, Intel compiler (serial)
4. Linux x86\_64, Intel compiler (dmpar)
5. Linux x86\_64, Intel compiler, SGI MPT (serial)
6. Linux x86\_64, Intel compiler, SGI MPT (dmpar)
7. Linux x86\_64, gfortran compiler (serial)
8. Linux x86\_64, gfortran compiler (dmpar)
9. Linux x86\_64, Intel compiler, IBM POE (serial)
10. Linux x86\_64, Intel compiler, IBM POE (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. NOTE: When using PGI compilers, please edit `configure.upp` and remove the `-Ktrap` flags from `FFLAGS`.

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

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

Alternatively, the ksh/bash command can be used:

## 2. Software Installation

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

This command should create 13 UPP libraries in `lib/`:

```
libbacio.a          libsfcio.a
libCRTM.a          libsigio.a
libg2.a            libsp.a
libg2tmpl.a        libw3emc.a
libgfsio.a         libw3nco.a
libip.a            libxmlparse.a
libnemsio.a
```

Four UPP executables are produced in `bin/`:

```
cnvgrib.exe
copygb.exe
ndate.exe
unipost.exe
```

Once again, these libraries are for the UPP only, and should not be used by the other components.

A complete clean is strongly recommended if the compilation failed, or if the configuration file or source code is changed. Conduct a complete clean that removes ALL build files, including the executables, libraries, and the `configure.upp`:

```
./clean
```

For full details on the operation of UPP, see the HWRF Post-Processor chapter of the HWRF Users' Guide, and for complete details on building and running the UPP, see the UPP Users' Guide, which can be downloaded at:

[http://www.dtcenter.org/upp/users/docs/user\\_guide/V3/upp\\_users\\_guide.pdf](http://www.dtcenter.org/upp/users/docs/user_guide/V3/upp_users_guide.pdf).

### 2.13 Building GSI

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.

### 2.13.1 Set Environment Variables

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.

Set up the environment for GSI:

```
setenv HWRF 1  
setenv WRF_DIR ${SCRATCH}/hwrfrun/sorc/WRFV3/
```

You may use bash/ksh instead:

```
export HWRF=1  
export WRF_DIR=${SCRATCH}/hwrfrun/sorc/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 vendor-provided version of LAPACK. IBM systems usually have the ESSL library installed and therefore do not need the LAPACK. Likewise, the PGI compiler often comes with a vendor-provided version of LAPACK that links automatically with the compiler. Problems with the vendor-supplied LAPACK library are more likely to occur with the Intel compiler. While the Intel compilers typically have the MKL libraries installed, the ifort compiler does not automatically load the library. It is therefore necessary to set the `LAPACK_PATH` variable to the location of the MKL libraries when using the Intel compiler.

Supposing that the MKL library path is set to the environment variable `MKL`, then the LAPACK environment may be set in terms of `MKL`:

```
setenv LAPACK_PATH $MKL
```

Alternatively, the bash/ksh option is as follows:

```
export LAPACK_PATH=$MKL
```

### 2.13.2 Configure and Compile

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. Software Installation

2. Linux x86\_64, PGI compilers (pgf90 & gcc) (dmpar,optimize)
3. Linux x86\_64, GNU compilers (gfortran & gcc) (dmpar,optimize)
4. Linux x86\_64, Intel compiler (ifort & gcc) (dmpar,optimize)
5. Linux x86\_64, Intel compiler (ifort & icc) (dmpar,optimize)
6. Linux x86\_64, Intel compiler (ifort & icc), IBM POE (EXPERIMENTAL) (dmpar,optimize)
7. Linux x86\_64, Intel compiler (ifort & icc), SGI MPT (EXPERIMENTAL) (dmpar,optimize)

Select the appropriate `dmpar` option for your platform and compiler. For a generic Linux machine, choose option (1) or (2) for a PGI build, or option (4) or (5) for an Intel build.

After selecting the proper option, run the compile script:

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

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

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

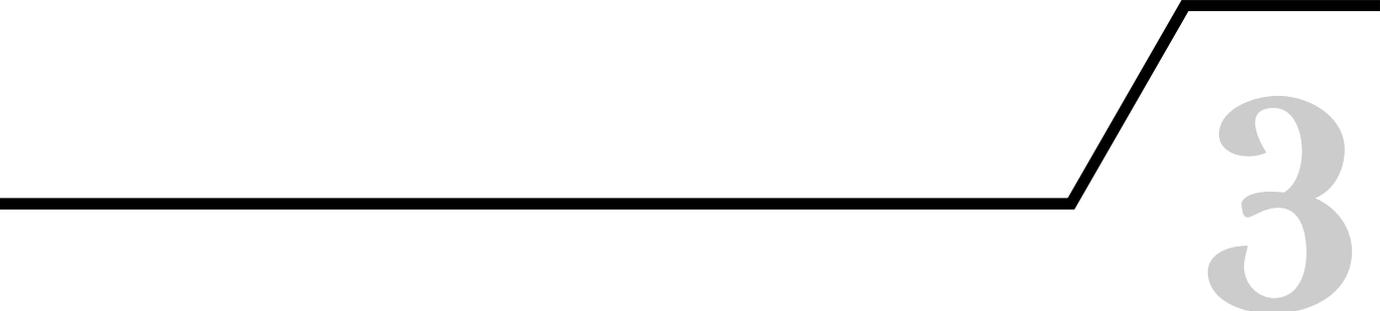
The successful completion of the compile will place the GSI executable `gsi.exe` in the `run/` directory. If the executable is not found, check the compilation log file to determine what went wrong.

A complete clean is strongly recommended if the compilation failed or if the configuration file is changed. To conduct a complete clean that removes ALL build files, including the executables, libraries, and the `configure.gsi`, use the "-a" option with clean:

```
./clean -a
```

For details on using GSI with HWRP, see the GSI chapter in the HWRP Users' Guide. For full details on the operation of GSI, see the DTC Community GSI Users' Guide:

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



# 3

## Running HWRF

### 3.1 HWRF Scripts Overview

HWRF v3.7a is run by a series of high-level shell scripts in the `wrappers/` directory, whose primary duty is to set environment variables and call the mid-level Python scripts in the `scripts/` directory. The mid-level Python scripts call HWRF-specific Python modules in the `ush/` directory and subdirectories. Many of the experiment configuration parameters and variables are set by files in the `parm/` directory:

```
hwrfrun/
├── confdoc.py
├── doc/
├── exec/
├── graphics/
├── nwport/
├── parm/
├── README.md
├── scripts/
├── src/
├── ush/
│   ├── hwrf/
│   ├── pom/
│   ├── produtil/
│   ├── hwrf_*.py
│   ├── hwrf_*.sh
│   ├── rocoto_*.sh
│   └── rocoto_*.py
```

### 3. Running HWRF

```
├── setup_hurricane.py
├── setup_hurricane
├── ush.dox
├── confdoc.py
└── wrappers/
```

The `doc` directory contains Doxygen based documentation about the scripting system. Please refer to <http://www.dtcenter.org/HurrWRF/users/docs/index.php> to access the documentation.

## 3.2 Defining an Experiment

### 3.2.1 Standard Configuration Files

The configuration of an HWRF experiment is controlled through the use of the configuration (`*.conf`) files in the `parm/` directory. Each of these configuration files has sections with headers contained in square brackets, e.g., `[config]`, `[dir]`, etc. Within a section, related variables are set.

There are four configuration files that are required by HWRF, and set all of the required variables needed to run every component of the workflow in the default configuration, which is the AL operational configuration. These four files are listed below:

```
hwrp.conf
hwrp_basic.conf
hwrp_input.conf
system.conf
```

The HWRF launcher, which is the first step in running HWRF, gathers all the options for a particular user configuration into the single file `storm1.conf` in the `/${COMhwrp}` directory. It does this by first reading in the configuration variables in the four files listed above, then accepting as command line arguments to the script `exhwrp_launch.py` additional configuration files and variables. The last option passed to the launcher overrides any previous settings, except for when `basin_overrides=yes` (more on that later). When the user passes a variable on the command line, the section and variable name are required, e.g., `config.diskproject=dtc-hwrp`.

We recommend that users not edit the four main configuration files, but instead pass in their own configuration files and variables when running the HWRF launcher, which will override the default settings. Most of the variables are documented within the four main configuration files.

## Overview of `hwrfl_v3.7release.conf`

---

For the HWRF v3.7a public release, there is a fifth configuration file, `hwrfl_v3.7release.conf`, which sets configuration options that require changes by the user, or that are required to be set based on the capabilities of HWRF v3.7a. The relevant variables are explained below.

[config]

<code>disk_project</code>	User's project on the computational platform.
<code>input_catalog</code>	Section within any configuration file that includes information about the locations and names of the input files. Community users should set it to <code>comm_hist</code> , which is defined below:
<code>run_ensemble_da</code>	Generates regional ensemble for data assimilation. This must be set to <code>no</code> because this capability is not supported in HWRFv3.7a.
<code>scrub</code>	Sets the automatic removal of temporary files.

[dir]

<code>inputroot</code>	Sets the path to the top-level data directory.
<code>syndat</code>	Sets the path to the input TC Vitals. Community users should set it to the location of staged SYNDAT-PLUS dataset, as explained later in this chapter.
<code>outputroot</code>	Sets the desired location of parent output directory.
<code>CDNOSCRUB</code>	Sets the path to the saved output.
<code>CDSCRUB</code>	Sets the path to the top-level output. Files and directories contained within would be scrubbed in the operational workflow by default to save disk space.
<code>CDSAVE</code>	Sets the path to the parent of the <code>hwrflrun</code> directory.

#### Other Variables

<code>scrub</code>	Sets the automatic removal of temporary files in <code>[relocate]</code> , <code>[gsi_d02]</code> , and <code>[gsi_d03]</code> working directories.
--------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------

To turn off scrubbing for any component of HWRF, set `scrub=no` for that section of configuration, e.g. pass `relocate.scrub=no` to the `exhwrfl_launch.py` script.

The `[comm_hist]` section provides the directory structure and naming conventions for the input data, where `inputroot` defines the parent data directory path.

The `[exe]` section provides the paths to the compiled executables.

### 3. Running HWRF

#### Overview of `hwrfl.conf`

---

The variables contained in `hwrfl.conf` primarily define namelist options for the components of HWRF. Please refer to the descriptions of the variables within the `hwrfl.conf`, along with the WRF NMM Users' Guide. Also included in this file is the mapping to the executables used in operations, which differ only by location from the executables used in the public release (overridden by `hwrfl_v3.7release.conf`).

#### Overview of `hwrfl_basic.conf`

---

The variables that are found in `hwrfl_basic.conf` define the HWRF workflow configuration. Users should be familiar with a few of the relevant sections and variables within this file, as follows. Further discussion about the use of these options is saved for section 3.6.

[config]	
<code>forecast_length</code>	Length of atmospheric forecast. Default is 126 hours.
<code>run_gsi</code>	Option to run GSI and FGAT initialization.
<code>run_ocean</code>	Option to run POM coupling.
<code>run_relocation</code>	Option to run vortex relocation/initialization.
<code>run_ensemble_da</code>	Option to run the DA ensemble (not supported in v3.7a).
<code>run_satpost</code>	Option to make synthetic satellite products.
<code>use_spectral</code>	Option to use spectral global files as input to <code>prep_hybrid</code> . If <code>no</code> , GRIB input files are used. Note that the GSI still needs spectral input from GDAS and the GFS Ensemble to run in hybrid mode.
<code>spectral_bdy</code>	Option to use spectral global files as boundary conditions if <code>use_spectral=yes</code> . If <code>no</code> , GRIB input files are used
<code>extra_trackers</code>	Option to run the GFDL vortex tracker on d02 and d03 (not supported in v3.7a).

Another important variable in this configuration file is `allow_fallbacks`. Most community users will have it set to `no`, which is the default. In operations, this variable is set to `yes`, which enables alternate paths in the HWRF run in case a component fails. For instance, when `allow_fallbacks` is set to `yes` and GSI fails, the run does not stop but instead the initial conditions for the main forecast are obtained directly from the output of the vortex relocation.

#### [prelaunch]

The variable `basin_overrides` is perhaps the most powerful of all the variables in the configuration files. This variable, when set to `yes`, will cause HWRF to run with operational settings for each basin by automatically loading a configuration file for a particular basin. Command line arguments to `exhwrfl_launch.py` will not override the settings within the basin-specific configuration files. Operational settings for each basin, along with the corresponding configuration files, are described in table 3.1.

#### [sanity]

### 3. Running HWRF

Basin	Ocean	Data assimilation	Vertical levels	Model top (hPa)	Extra configuration file
<b>AL</b>	POM	Always	61	2	None
<b>EP</b>	POM	Only if TDR available	61	2	hwrf_EP.conf
<b>NC Pac</b>	None	No	61	2	hwrf_CP.conf
<b>NW Pac</b>	None	No	43	50	hwrf_other_basins.conf
<b>N Ind</b>	None	No	43	50	hwrf_other_basins.conf
<b>S Pac</b>	None	No	43	50	hwrf_other_basins.conf
<b>S Ind</b>	None	No	43	50	hwrf_other_basins.conf
<b>S Atl</b>	None	No	43	50	hwrf_other_basins.conf

Table 3.1: Default settings for each basins.

Variables in this section control which checks run during the execution of `launcher_wrapper`.

[dir]

Contains variables that build paths based on the paths provided by `CDSAVE`, `CDSCRUB`, `CDNOSCRUB`, and `EXPT`. These variables should be changed with extreme caution. Following is a description of three of the most important [dir] variables:

`WORKhwrf`

The working directory for all jobs. These directories are scrubbed in the operational workflow to save disk space.

`HOMEhwrf`  
`com`

HWRF installation location.

Location of output files for delivery to operational centers or for use by the next HWRF cycle.

#### Overview of `hwrf_input.conf`

---

This file defines the input data directory structure. It currently contains input data locations for users on the NOAA research and development machines Zeus or Jet with access to the EMC input data directory, and for community users that stage the input data using the directory structure described in section 3.3. These three sets of input data are described in the sections labeled `[zeus_hist]`, `[jet_hist]`, and `[comm_hist]`, respectively. Settings for `[comm_hist]` for this release can be found in `hwrf_v3.7release.conf`.

Users who wish to adopt a different input data directory structure may define it within this

### 3. Running HWRF

file by adding an additional section, or by editing the existing `[comm_hist]` section. While the input data can be placed anywhere that is locally available to the compute nodes, users are not advised to change the input file naming convention.

The choice of which set of input data will actually be used in an experiment is determined by variable `input_catalog` in file `system.conf`. To use the test datasets provided by DTC, users should set this variable to `comm_hist`. The user must also set the path to the location of the input data within the `[comm_hist]` section of `hwrp_input.py` by editing the variable `inputroot`.

#### Overview of `system.conf`

---

This file defines the top-level output directory structure and a handful of other variables used for running HWRF. Community users need to copy or link the example file `system.conf.jet` to `system.conf`. The following variables are user-dependent and originate from this file, but should be set in `hwrp_v3.7arelease.conf`.

<code>disk_project</code>	Project on the computational platform (used to compose the output directory name).
<code>input_sources</code>	Variable that sets the configuration section that defines the local and remote staging areas of input data. This configuration section must exist in one of the configuration files used by HWRF (i.e., <code>hwrp_input.conf</code> , <code>hwrp_v3.7arelease.conf</code> , etc.). Community users should set it to <code>comm_hist</code> .
<code>input_catalog</code>	Variable that sets the configuration section that defines the input file location. This section must exist in one of the configuration files used by HWRF. Community users should set it to <code>comm_hist</code> .
<code>[dir] section</code>	<code>CDNOSCRUB</code> , <code>CDSCRUB</code> , and <code>CDSAVE</code> set the location of the HWRF install and output files.
<code>syndat</code>	The path to input TC Vitals. Community users should set it to the path for the staged SYNDAT-PLUS dataset, as explained later in this chapter.

The sections `[wrfexe]` and `[runwrf]` describe the mapping of the processor distribution for the WRF runs and should not be altered.

### 3.3 Input Data and Fix Directory Structure

Users will need the datasets below as input to HWRF components.

The following lists outline the files needed to initialize forecast using the operational configuration. Analysis times are indicated by capital letters, such as `YYYYMMDDHH` or `HH`, and forecast hours are indicated by lowercase letters, i.e., `hhh`. For example,

### 3. Running HWRF

gfs.2012102806.pgrbf024 is a GFS 24-h forecast in GRIB format whose initial time is October 28 06Z 2012 and would be indicated as gfs.YYYYMMDDHH.pgrbfhhh. The storm identifier, *sid* is the storm number and a lower-case letter corresponding to the basin (e.g., 081).

```

./
├── DATA/..... Top level data directory
├── gfs.YYYYMMDDHH/
├── enkf.YYYYMMDDHH/
├── loop/
├── recon.YYYYMMDDHH/
├── gdas1.YYYYMMDD/
├── SYNDAT-PLUS/
│   ├── tcvitals.txt.prev.YYYY
│   ├── syndat_tcvitals.YYYY
│   ├── investnamed/
│   ├── old/
│   ├── renumbered/
│   └── work/
├── tdr.YYYYMMDDHH/ ..... TDR data, if available
├── fix/
│   ├── bogus/
│   ├── hwrf_cpat.ice
│   ├── hwrf_cpat.moddef
│   ├── hwrf_cpat.wind
│   ├── hwrf-crtm-2.0.6/ ..... Updated Coefficients for Radiative Transfer
│   ├── hwrf-crtm-2.2.0/ ..... Updated Coefficients for Radiative Transfer
│   ├── hwrf-crtm-2.2.1/ ..... Updated Coefficients for Radiative Transfer
│   ├── hwrf_disclaimer.txt
│   ├── hwrf_eta_micro_lookup.dat
│   ├── hwrf_fix_datestamp
│   ├── hwrf-pom/ ..... Ocean init climate data
│   ├── hwrf_storm_20
│   ├── hwrf_storm_25
│   ├── hwrf_storm_cyn_axisy_47
│   ├── hwrf_storm_cyn_axisy_50_ep
│   ├── hwrf_storm_radius
│   ├── hwrf_track
│   ├── hwrf_wps_geo/ ..... Land use fixed files
│   ├── hwrf-wrf/ ..... Fixed files for WRF
│   └── loop_curr/ ..... Loop current initialization data

```

The loop/ directory contains loop current and warm core ring initialization data in the following two files.

gfdl_loop_current_rmy5.dat.YYYYMMDD	Loop current data
gfdl_loop_current_wc_ring_rmy5.dat.YYYYMMDD	Warm core ring data

The gfs.YYYYMMDDHH/ directory contains spectral atmospheric analyses.

### 3. Running HWRP

<code>gfs.tHH.sanl</code>	GFS analysis for ocean initialization
<code>gfs.tHH.sfcanl</code>	GFS sfc analysis for ocean initialization
<code>gfs.tHH.sfhhh</code>	GFS analysis for atmospheric initialization (0 to 126h in 3 h increments)

The `gfs.YYYYMMDDHH/` directory also contains data for the initialization of the atmosphere. These fall into three categories: observations, gridded data, and spectral data. The prefix for each file denotes the numerical weather prediction (NWP) system from which the data originate.

The following files are considered observations, and are in either BUFR or prepBURF formats where *HH* is the analysis time:

<code>gfs.tHH.prepbufr.nr</code>	Conventional obs
<code>gfs.tHH.SATELLITE.tm00.bufr_d</code>	Satellite obs

In the list above, *SATELLITE* can be any of the following, but this is not an all-inclusive list:

<code>1bamua</code>	<code>ascatt</code>	<code>esamua</code>	<code>rassda</code>
<code>1bamub</code>	<code>ascatw</code>	<code>esamub</code>	<code>sevcsr</code>
<code>1bhrs3</code>	<code>atms</code>	<code>goesfv</code>	<code>satwind</code>
<code>1bhrs4</code>	<code>avcsam</code>	<code>gome</code>	
<code>1bmhs</code>	<code>avcspm</code>	<code>gpsro</code>	
<code>airsev</code>	<code>cris</code>	<code>mtiasi</code>	

The following file is in gridded binary (GRIB) format:

`gfs.tHH.pgrb2.0p25.f00` GFS analysis

The `enkf.YYYYMMDDHH/` directory contains the GFS Ensemble spectral data used for data assimilation. There are eighty files named

`sfg_YYYYMMDDHH_fhrhhs_mem{mmm}`,

where *mmm* is the 3-digit ensemble member ID, which ranges from 001 to 080.

The `gdas1.YYYYMMDDHH/` directory contains the GDAS files to provide first guess information for data assimilation.

The following files are in gridded binary (GRIB) format:

<code>gdas1.tHH.pgrb2.1p00.f003</code>	3-h forecast from previous 6-h GDAS cycle
<code>gdas1.tHH.pgrb2.1p00.f006</code>	6-h forecast from previous 6-h GDAS cycle
<code>gdas1.tHH.pgrb2.1p00.f009</code>	9-h forecast from previous 6-h GDAS cycle

The following files are considered observations, and are in either BUFR or prepBURF formats. *HH* is the analysis time and *HH-6* is the analysis time six hours prior.

<code>gdas1.t{HH-6}.abias</code>	Time-dep. sat. bias correction
<code>gdas1.t{HH-6}.abias_pc</code>	Satellite-bias correction coefficients

### 3. Running HWRP

The remainder of the files are spectral files in binary format.

<code>gdas1.tHH-6.sf03</code>	3-h forecast from previous 6-h GDAS analysis
<code>gdas1.tHH-6.sf06</code>	6-h forecast from previous 6-h GDAS analysis
<code>gdas1.tHH-6.sf09</code>	9-h forecast from previous 6-h GDAS analysis

The `SYNDAT-PLUS` directory contains the TC Vitals files for several years.

The `recon.YYYYMMDD` directory contains aircraft reconnaissance data in BUFR format and some additional files with metadata. Files are named by the following conventions:

```
gdas1.tHH.SID.hdob.tm00.bufr_d
```

The `tdr.SID/YYYY/YYYYMMDDHH/SID` directory contains TDR data in BUFR format in a file named as follows:

```
gdas1.tHHz.tldplr.tm00.bufr_d
```

The fix files are time-independent and are included in the following directories:

- `bogus/`

```
hwrp_ofs_atl.A12grid.dat
hwrp_ofs_atl.intrp_pars.dat
hwrp_ofs_atl.ismus_msk1152x576.dat
hwrp_ofs_atl.ismus_msk384x190.dat
hwrp_ofs_atl.ismus_msk512x256.dat
hwrp_ofs_atl.ismus_msk768x384.dat
hwrp_ofs_atl.ncep1_12.regional.depth.a
hwrp_ofs_atl.ncep1_12.regional.depth.b
hwrp_ofs_atl.ncep1_12.regional.grid.a
hwrp_ofs_atl.ncep1_12.regional.grid.b
hwrp_ofs_atl.ncep1_12.regional.mask.a
hwrp_ofs_atl.ncep1_12.regional.mask.b
```

- `hwrp-crtm-2.2.1/`

```
AerosolCoeff/
CloudCoeff/
EmisCoeff/
SpcCoeff/
TauCoeff/
```

- `hwrp-gsi/` NOTE: starting with v3.7a, the GSI fixed files are included in the `src-c/GSI/fix` directory

<code>anavinfo_hwrp_L60</code>	<code>bufrtab.012</code>
<code>anavinfo_hwrp__L60_nooz</code>	<code>global_ozinfo.txt</code>
<code>anavinfo_hwrp_L75</code>	<code>global_scaninfo.txt</code>
<code>atms_beamwidth.txt</code>	<code>hwrp_convinfo.txt</code>

### 3. Running HWRF

```
hwrp_nam_errtable.r3dv          nam_global_satangbias.txt
hwrp_satinfo.txt                prepobs_prep.bufrtable
nam_glb_berror.f77.gcv
nam_global_pcpinfo.txt
```

- hwrp-pom/ - [#-#] represents multiple files numbered consecutively, e.g. [01-12] means there are twelve files with a two-digit number ranging from 01-12 replacing the string within the brackets.

```
depths_sfc_000000_000000_1o2161x  gfdl_ocean_topo_and_mask.eastatl_e
 1051_datafld                        xtn
gfdl_albedo.fall                    gfdl_ocean_topo_and_mask.eastpac
gfdl_albedo.spring                  lores
gfdl_albedo.summer                  gfdl_ocean_topo_and_mask.eastpac_x.
gfdl_albedo.winter                  lores
gfdl_datainp1                       gfdl_ocean_topo_and_mask.northind.
gfdl_datainp1.142                   lores
gfdl_datainp2                       gfdl_ocean_topo_and_mask.sepac.
gfdl_disclaimer.txt                 lores
gfdl_fildef.afos                    gfdl_ocean_topo_and_mask.southatl.
gfdl_fildef.sdm                     lores
gfdl_fort.7                          gfdl_ocean_topo_and_mask.southind.
gfdl_fort.7.142                     lores
gfdl_gdem.[00-13].ascii             gfdl_ocean_topo_and_mask.swpac.
gfdl_Hdeepgsu.eastatl               lores
gfdl_Hdeepgsu.united                gfdl_ocean_topo_and_mask.transatl.
gfdl_height                          lores
gfdl_huranal.data                  gfdl_ocean_topo_and_mask.united
gfdl_initdata.eastatl.[01-12]        gfdl_ocean_topo_and_mask.united_
gfdl_initdata.gdem3.united.[05-12]  12th
gfdl_initdata.gdem.united.[01-12]   gfdl_ocean_topo_and_mask.united.
gfdl_initdata.levit.united.[05-12]  lores
gfdl_initdata.united.[01-12]        gfdl_ocean_topo_and_mask.westpac.
gfdl_limit_2nest_dat_x.1             lores
gfdl_limit_2nest_dat_x.5             gfdl_pctwat
gfdl_limit_2nest_dat_x.6             gfdl_raw_temp_salin.eastpac.
gfdl_limit_2nest_dat_y.1             [04-12]
gfdl_limit_2nest_dat_y.12            gfdl_wetness
gfdl_limit_2nest_dat_y.15            gfdl_znot
gfdl_limit_2nest_dat_y.16            glb_ocn
gfdl_limit_2nest_dat_y.6             grdlat_sfc_000000_000000_1o2161x105
gfdl_ocean_dat                       newline 1_datafld
gfdl_ocean_readu.dat.[01-12]         grdlon_sfc_000000_000000_1o2161x105
gfdl_ocean_spinup.BAYuf              newline 1_datafld
gfdl_ocean_spinup.FSGsuf             maskls_sfc_000000_000000_1o2161x105
gfdl_ocean_spinup_gdem3.dat.         newline 1_datafld
  [01-12]                             sgdemv3s[01-12].nc
gfdl_ocean_spinup_gspath.[01-12]    tgdemv3s[01-12].nc
gfdl_ocean_spinup.SGYREuf           the.diff
gfdl_ocean_topo_and_mask.eastatl_1   there.diff
 2th
```

- hwrp\_wps\_geo/

```
albedo_ncep/                        hanis/
greenfrac/                           hasynw/
hangl/                                hasys/
```

### 3. Running HWRF

- hasysw/
  - hasyw/
  - hcnvx/
  - hlennw/
  - hlens/
  - hlensw/
  - hlenw/
  - hslop/
  - hstdv/
  - hzmax/
  - islope/
  - landuse\_10m/
  - landuse\_2m/
  - landuse\_30s/
  - landuse\_30s\_with\_lakes/
  - landuse\_5m/
  - maxsnowalb/
  - modis\_landuse\_20class\_30s/
  - modis\_landuse\_21class\_30s/
  - orogwd\_10m/
  - orogwd\_1deg/
  - orogwd\_20m/
  - orogwd\_2deg/
  - orogwd\_30m/
  - soiltemp\_1deg/
  - soiltype\_bot\_10m/
  - soiltype\_bot\_2m/
  - soiltype\_bot\_30s/
  - soiltype\_bot\_5m/
  - soiltype\_top\_10m/
  - soiltype\_top\_2m/
  - soiltype\_top\_30s/
  - soiltype\_top\_5m/
  - ssib\_landuse\_10m/
  - ssib\_landuse\_5m/
  - topo\_10m/
  - topo\_2m/
  - topo\_30s/
  - topo\_5m/
  - varssso/
- hwrfr-wrf/
    - aerosol.formatted
    - aerosol\_lat.formatted
    - aerosol\_lon.formatted
    - aerosol\_plev.formatted
    - bulkdens.asc\_s\_0\_03\_0\_9
    - bulkradii.asc\_s\_0\_03\_0\_9
    - CAM\_ABS\_DATA
    - CAM\_AEROPT\_DATA
    - CAMtr\_volume\_mixing\_ratio.A1B
    - CAMtr\_volume\_mixing\_ratio.A2
    - CAMtr\_volume\_mixing\_ratio.RCP4.5
    - CAMtr\_volume\_mixing\_ratio.RCP6
    - CAMtr\_volume\_mixing\_ratio.RCP8.5
    - capacity.asc
    - CCN\_ACTIVATE.BIN
    - CLM\_ALB\_ICE\_DFS\_DATA
    - CLM\_ALB\_ICE\_DRC\_DATA
    - CLM\_ASM\_ICE\_DFS\_DATA
    - CLM\_ASM\_ICE\_DRC\_DATA
    - CLM\_DRSDT0\_DATA
    - CLM\_EXT\_ICE\_DFS\_DATA
    - CLM\_EXT\_ICE\_DRC\_DATA
    - CLM\_KAPPA\_DATA
    - CLM\_TAU\_DATA
    - co2\_trans
    - coeff\_p.asc
    - coeff\_q.asc
    - constants.asc
    - ETAMPNEW\_DATA
    - ETAMPNEW\_DATA\_DBL
    - ETAMPNEW\_DATA.expanded\_rain
    - ETAMPNEW\_DATA.expanded\_rain\_DBL
    - GENPARM.TBL
    - grib2map.tbl
    - gribmap.txt
    - kernels.asc\_s\_0\_03\_0\_9
    - kernels\_z.asc
    - LANDUSE.TBL
    - masses.asc
    - MPTABLE.TBL
    - ozone.formatted
    - ozone\_lat.formatted
    - ozone\_plev.formatted
    - README.fix
    - README.namelist
    - README.tslist
    - RRTM\_DATA
    - RRTM\_DATA\_DBL
    - RRTMG\_LW\_DATA
    - RRTMG\_LW\_DATA\_DBL
    - RRTMG\_SW\_DATA
    - RRTMG\_SW\_DATA\_DBL
    - SOILPARM.TBL
    - termvels.asc
    - tr49t67
    - tr49t85
    - tr67t85
    - URBPARM.TBL
    - URBPARM\_UZE.TBL
    - VEGPARM.TBL
    - wind-turbine-1.tbl
  - loop\_curr/
    - hwrfr\_gfdl\_loop\_current\_rmy5.dat.YYYYMMDD

### 3. Running HWRf

hwr\_f\_gfdl\_loop\_current\_wc\_ring\_rmy5.dat.YYYYMMDD

Sample fix files and datasets for running two consecutive forecasts of Hurricane Gonzalo (October 14, 2014 12 and 18 UTC) can be obtained from the DTC website: <http://www.dtcenter.org/HurrWRF/users>. To use the DTC-supported scripts for running HWRf, these datasets should be stored following the directory structure described above, and must be on a disk accessible by the HWRf scripts and executables.

The following files are available for download:

HWRf\_v3.7a\_datasets\_enkf.2014101406.tar.gz  
HWRf\_v3.7a\_datasets\_enkf.2014101412.tar.gz  
HWRf\_v3.7a\_datasets\_gfs.2014101412.tar.gz  
HWRf\_v3.7a\_datasets\_gfs.2014101418.tar.gz  
HWRf\_v3.7a\_datasets\_gdas.2014101406.tar.gz  
HWRf\_v3.7a\_datasets\_gdas.2014101412.tar.gz  
HWRf\_v3.7a\_datasets\_recon.tar.gz  
HWRf\_v3.7a\_datasets\_SYNDAT-PLUS.tar.gz  
HWRf\_v3.7a\_datasets\_loop.tar.gz  
HWRf\_v3.7a\_fix.tar.gz

### 3.4 Production Directory Structure

The top production directory is  $\${WORKhwr_f}/YYYYMMDDHH/SID$  (where  $\${WORKhwr_f}$  is an environment variable defined by the scripts,  $SID$  is storm ID (e.g., 09L), and  $YYYYMMDDHH$  is the forecast initial time. The following subdirectories will be present for the default AL configuration.

```
 $\${WORKhwr_f}/YYYYMMDDHH/SID$ 
├── cpl.out ..... Coupler std out
├── fgat.tYYYYMMDDHHHH/..... Production dir for processing GDAS at each of the fgat hours: -3, 0 +3
├── gfsinit/ ..... Production dir for processing GFS input
│   ├── ghost/
│   ├── prep_hybrid/
│   ├── realfcst/
│   ├── realinit/
│   ├── relocate/
│   ├── regribber/
│   ├── tracker/
│   ├── wps/
│   └── wrfanl/
├── gsi_d02/ ..... Production dir for data assimilation on 6 km domain
├── gsi_d03/ ..... Production dir for data assimilation on 2 km domain
├── hwr_f_state.sqlite3 ..... Datastore file
└── hwr_f_state.sqlite3.lock
```

### 3. Running HWRF

	intercom/	.....	Dir containing files needed for subsequent processes
	lock/	.....	Datastore directory
	oldvit	.....	Previous 6 h Tcvitals
	PDY	.....	Text file with date information
	pom/	.....	Production dir for the ocean initialization
	regribber/	.....	Working dir for regribbing process. All files are delivered to other locations
	regribber-[0003-0010].out/	.....	Standard output from regribber
	regribber-[0003-0010].err/	.....	Standard error from regribber
	runwrf/	.....	Production dir for coupled or uncoupled forecast
	storm1.vitals	.....	Text file containing TC Vitals for forecast storm, including only the current storm label, i.e., 08L for Gonzalo
	storm1.vitals.allids	.....	Text file containing TC Vitals for forecast storm, including its Invest labels, i.e., 99L and 08L for Gonzalo
	storm1.vitals.oldid	.....	Text file containing TC Vitals for forecast storm, including only its Invest labels, i.e., 99L for Gonzalo
	storm1.vitals.renumberlog	.....	Log file for vitals messages processed
	tmpvit	.....	Text file containing only the analysis TC Vital message
	tracker/	.....	Production dir for the GFDL Vortex Tracker
	trackerd01/	.....	Production dir for the GFDL Vortex Tracker from parent domain
	trackerd02/	.....	Production dir for the GFDL Vortex Tracker from d02 domain
	tracker-[0000-0003].out/	.....	Standard output from tracker

The purpose of `intercom/` is to store the files that are used for subsequent processes, separating them from the working directory. Within the `intercom/` directory, the structure is similar to that in the `WORKhwrfl` directory, except each subdirectory contains only the files that will be used in subsequent steps. The following outlines the structure of `intercom/` when running all components of HWRF as in the default AL configuration.

```
intercom/
|_ bufrrprep/
|_ fgat.tYYYYMMDDHHHH/
|_ gdas_merge/
|_ gfsinit/
|_ {STORMNAME}{SID}.YYYYMMDDHH.hwrfrprs.GRID.res.grbfhh
|_ {STORMNAME}{SID}.YYYYMMDDHH.hwrfsat.GRID.res.grbfhh
|_ {STORMNAME}{SID}.YYYYMMDDHH.hwrfrtk.grbfhh.grbindex
|_ gsi_d02/
|_ gsi_d03/
|_ nonsatpost-f{hh}h00m/
|   |_ nonsatpost-fhhh00m-moad.egrb
|   |_ nonsatpost-fhhh00m-storm1inner.egrb
|   |_ nonsatpost-fhhh00m-storm1outer.egrb
|_ pom/
|_ regribber/
|_ satpost-f{hh}h00m/
|   |_ satpost-fhhh00m
|   |_ satpost-fhhh00m-moad.egrb
|   |_ satpost-fhhh00m-storm1inner.egrb
|   |_ satpost-fhhh00m-storm1outer.egrb
```

### 3. Running HWRP

Additionally, some output files are transferred to the `com/` directory, which is reserved for transfer of files between cycles and for delivery of final products (in an operational setting). This is discussed in Chapter 11.

In the list above, date strings will be substituted for each forecast hour or fgat time. `GRID` denotes grids of various domains, e.g., `d1`, `d123`, `d2`, `d23`, `d3`, and grid spacings used for regridding the HWRP forecast for delivery and product generation. In this context, `moad` stands for *Mother Of All Domains*, or the HWRP parent grid. Conversely, `outer` and `inner` refer to the intermediate and innermost nests. More information about each grid can be found in section 11.2.1.

## 3.5 Scripts for Running HWRP

We recommend that HWRP v3.7a be run using the wrapper and Python scripts provided with the HWRP v3.7a release. In `scripts/`, users can find mid-level Python scripts that call HWRP-specific Python utilities located in `ush/`. Users are encouraged to run the scripts in the `scripts/` directory using the wrapper scripts located in `wrappers/`. The wrapper scripts set the proper environment variables to run each Python script, as well as execute multiple iterations as needed.

### 3.5.1 Submitting a Job

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 NOAA's 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 HWRP.

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 beginning of each wrapper script should be edited to add the LSF options listed below:

### 3. Running HWRF

```
#BSUB -P 99999999          # Project name
#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      # Standard output file name
#BSUB -e hwrf.%J.out      # Standard error file name
#BSUB -W 2:30             # Wallclock time
#BSUB -q debug            # Queue name
#BSUB -K                  # Don't return prompt until the job is finished
```

For a platform that uses the MOAB/Torque batch system, the beginning of each wrapper script should be edited to add the PBS options listed:

```
#PBS -A project          # Project name
#PBS -l procs=202        # Number of total (MPI) tasks
#PBS -o stdout.txt       # Standard output file name
#PBS -e stderr.txt       # Standard error file name
#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 using the command:

- On machines with LSF:  
**bsub** sample\_wrapper
- On machines with MOAB/Torque:  
**qsub** sample\_wrapper

The wrapper script `sample_wrapper` will be submitted to the computation nodes and, once it starts, will call the low-level script from the `scripts/` directory. Appendix A contains the guidelines for resources used to run HWRF at near operational efficiency.

## 3.6 Running HWRF End-to-End

### 3.6.1 Editing `global_vars.sh`

The file `global_vars.ksh` is used for setting a few environment variables to define the cycle(s) to be run by the wrapper scripts. The user should set the first four variables. Variables are set for the Gonzalo example case provided by the DTC:

### 3. Running HWRF

<code>export START_TIME=YYYYMMDDHH</code>	Initial time of the forecast
<code>export SID=08L</code>	Storm ID, e.g. 08L is Gonzalo – the 08th storm of the Atlantic season; Invest 99 in the East Pacific would be 99E
<code>export CASE=HISTORY</code>	For most users, this will be <code>HISTORY.FORECAST</code> is reserved for real-time mode.
<code>export HOMEhwrflauncher=</code>	Full path to the directory where HWRF has been installed, including <code>/hwrflauncher</code>

#### 3.6.2 Using Wrapper Scripts

Once all configure files and `global_vars.sh` have been edited to define an experiment, the wrapper scripts should be submitted to the batch system. Some of the wrappers have dependencies on previous wrappers, while others can be run simultaneously. The following list contains all of the wrappers to be run for a default Atlantic Basin case. The items under the same number can be submitted together in any order, but only after the previous numbered item(s) runs to completion:

1. `launcher_wrapper`
2. `init_gdas_wrapper`  
`init_gfs_wrapper`  
`init_ocean_wrapper`  
`bufprep_wrapper`
3. `relocate_wrapper`
4. `gsi_d02_wrapper`  
`gsi_d03_wrapper`
5. `merge_wrapper`
6. `unpost_wrapper`
7. `forecast_wrapper`  
`post_wrapper`<sup>†</sup>  
`products_wrapper`<sup>†</sup>

<sup>†</sup>Wrapper can run at the same time as `forecast_wrapper`, but should only be submitted after the forecast job has started, i.e., no longer in queue.

Wrappers with the same number may be run sequentially or simultaneously. Because the forecast job often waits in the queue before it starts, a post job submitted at the same time as the forecast job will have nothing to do for quite a while and will use wallclock time waiting on output from the forecast. Therefore, we suggest submitting the post and products wrappers after the forecast job has started running.

By default, the `launcher_wrapper` reads in the configure files `system.conf`, `hwrflauncher.conf`, `hwrflauncher_input.conf`, `hwrflauncher_v3.7release.conf` and `hwrflauncher_basic.conf`. Additional variables and configure files can be passed to `exhwrflauncher.py` within the `launcher_wrapper` by following the syntax documented within the Python script. For variables defined more than once, the variable will take the value that was last passed to

### 3. Running HWRF

exhwrflauncher.py.

The configuration set by the user will determine the wrapper scripts that are required. Keep in mind, however, that when the variable `basin_overrides=yes`, the configuration changes based on the basin for which you are running, so wrappers should be submitted accordingly.

An example of the command used in the `launcher_wrapper` for running AL storm Gonzalo is as follows. Please refer to the documentation block of `scripts/exhwrflaunch.py` for complete argument list. The environment variables `$HOMEhwrflaunch`, `$EXPT`, `$CASE_ROOT`, and `$startfile` are defined in `global_vars.sh`.

```
$HOMEhwrflaunch/scripts/exhwrflaunch.py 2014101412 08L HISTORY  
"$HOMEhwrflaunch/parm" "config.EXPT=$EXPT" "config.startfile=$startfile"  
"config.HOMEhwrflaunch=$HOMEhwrflaunch" "config.case_root=$CASE_ROOT"  
"$HOMEhwrflaunch/parm/hwrf_v3.7release.conf"
```

To run the full default AL configuration for any other basin, simply set `prelaunch.basin_overrides=no` as an additional argument in the previous example and choose the date and storm ID for a storm in a different basin.

In addition to editing the `launcher_wrapper`, the `forecast_wrapper` will also require the user to edit the number of processors, depending on the status of ocean coupling and resolution used for the forecast. Please refer to the comments within the `forecast_wrapper` for the appropriate choice of processors.

## 3.7 Operational HWRF in Each Ocean Basin

In the 2015 Operational implementation of HWRF, the configuration of the system varies by basin as described in table 3.1. The following sections describe the workflow of each of those configurations, and the appropriate wrappers to go with them.

### 3.7.1 Atlantic Basin

In Figure 3.1, each blue box corresponds to a component of HWRF that is run in the default configuration. Except for the forecast grouping, each box has its own corresponding wrapper. The three components of the forecast are run simultaneously by the `forecast_wrapper`. The `bufrrprep_wrapper` and `unpost_wrappers` are preparation steps for the GSI and the post jobs, respectively, and are not shown as components in the figures.

### 3. Running HWRF

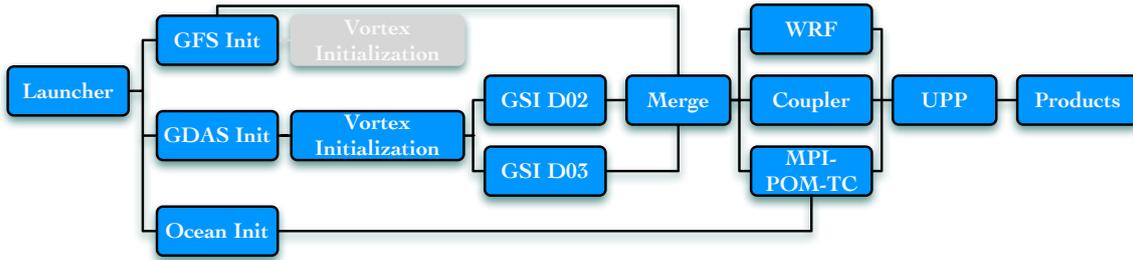


Figure 3.1: Components of HWRF that run for the operational AL configuration.

#### 3.7.2 Eastern Pacific Basin

In Figure 3.2, the workflow "With Inner Core Data" is used when TDR from a NOAA P3 flight is available. This configuration, which is quite rare given that flights do not often occur for EP storms, uses all of the same wrappers as the AL configuration. The bottom workflow of Fig. 3.2 describes the most common EP configuration.

Before running this workflow, users need to edit the `relocate_wrapper` to remove or comment the lines related to running vortex relocation on the GDAS files.

```
# Run the python relocate script for fgat times
# if test "$GSI" = "YES" ; then
# export INIT_MODEL=gdas1
# for fhr in 3 6 9 ; do
# export INIT_FHR=$fhr
# $EXhwrffexhwrf_relocate.py
# done
# fi
```

The wrappers below should be used when no TDR data is available for an EP storm.

1. `launcher_wrapper`
2. `init_gfs_wrapper`  
`init_ocean_wrapper`
3. `relocate_wrapper`
4. `unpost_wrapper`
5. `forecast_wrapper`  
`post_wrapper`<sup>†</sup>  
`products_wrapper`<sup>†</sup>

<sup>†</sup>Wrapper can run at the same time as `forecast_wrapper`, but should only be submitted after the forecast job has started, i.e., no longer in queue.

### 3. Running HWRF

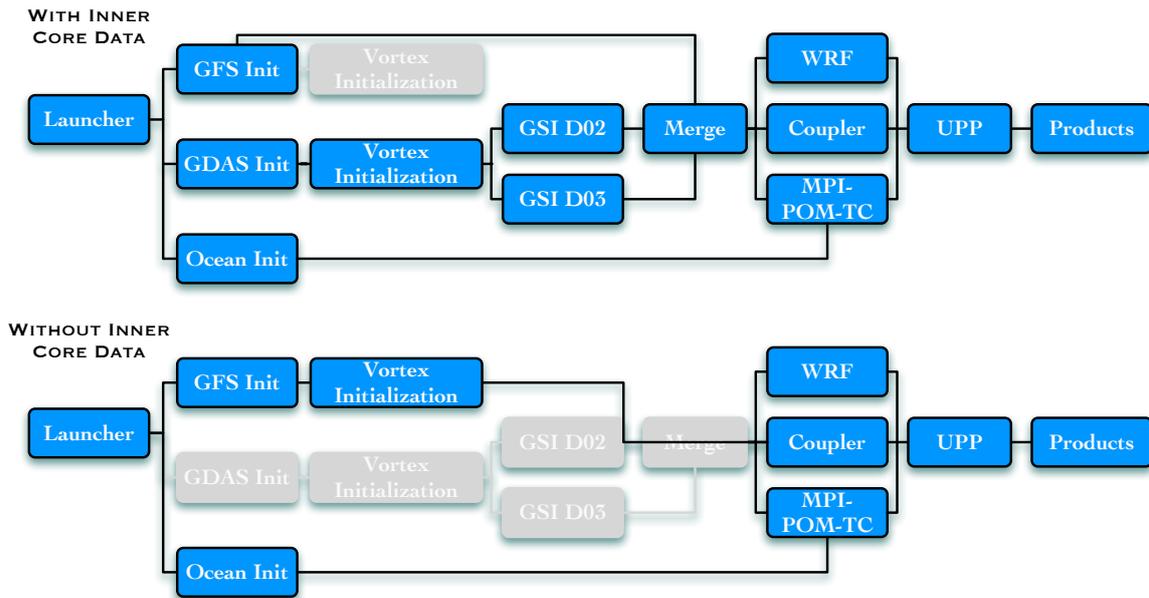


Figure 3.2: Components of HWRF that run for the operational EP configuration, which depends on the existence of inner core data.

#### 3.7.3 All Other Basins

Beginning in 2015, HWRF is run operationally for all ocean basins globally. These include the North Central Pacific, North West Pacific, Northern Indian Ocean, South Pacific, Southern Indian Ocean, and Southern Atlantic Basins. While the configuration of the vertical levels differs between these basins, the workflow configuration does not. Notice in Fig. 3.3 that the workflow has been simplified a great deal. The only wrappers required to run the default cases for these basins are as follows:

1. launcher\_wrapper
2. init\_gfs\_wrapper
3. relocate\_wrapper
4. unpost\_wrapper
5. forecast\_wrapper  
post\_wrapper<sup>†</sup>  
products\_wrapper<sup>†</sup>

<sup>†</sup>Wrapper can run at the same time as forecast\_wrapper, but should only be submitted after the forecast job has started, i.e., no longer in queue.

Note: Do not forget to edit the forecast\_wrapper for an uncoupled run.

### 3. Running HWRF

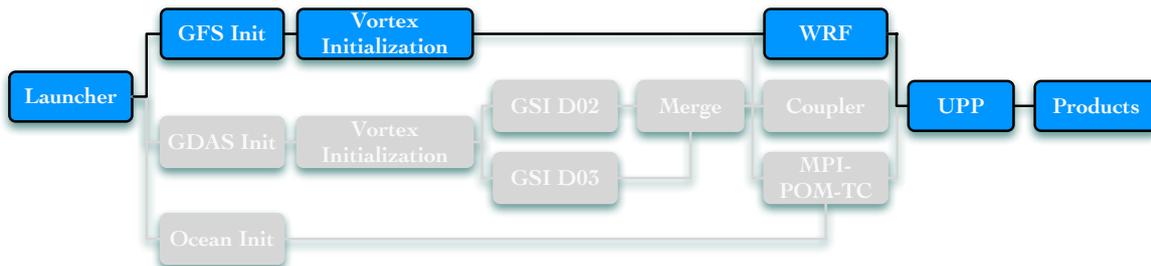


Figure 3.3: Components of HWRF that run for the operational configuration in all other basins.

## 3.8 Running HWRF in Non-operational Configurations

Users may wish to run HWRF with a different set of components than those used in operations. Most alternate configurations will require the user to set `basin_overrides=no`. If `basin_overrides=no` is the only additional configuration option set, all runs in all basins will be configured as the AL default.

The basin-specific configuration files listed in the "Extra conf" column of table 3.1 may still be passed through the launcher, but the variables within can be overridden by a subsequent argument to `exhwrflaunch.py`.

The remainder of this section describes a few alternate options that are supported when using HWRF v3.7a. These examples are also included in `launcher_wrapper`. It is left to the user to choose combinations of the following configuration options.

### 3.8.1 Running Coupled/Uncoupled Forecast

Coupling with the ocean is enabled by default for the AL and EP basins. To run an uncoupled forecast (no ocean) in the AL or EP basins, set `config.run_ocean=no`.

This configuration option also requires a change in the number of processors in the `forecast_wrapper`. Please see the explanation contained in the comments section of the wrapper to set the appropriate number of processors.

To enable ocean in any other basin, the two arguments below should be passed to `exhwrflaunch.py`.

```
prelaunch.basin_overrides=no
run_ocean=yes
```

If the ocean is the only component to be changed, the `hwrfotherbasins.conf` must also be used explicitly in the `launcher_wrapper`, otherwise the default AL configuration

### 3. Running HWRF

will be run.

An example of the arguments passed to `exhwrflaunch.py` within the `launcher_wrapper` to run an ocean-coupled forecast for a WP storm, retaining all other operational setting, follows:

```
$HOMEhwrflaunch.py 2014100906 19W HISTORY  
"$HOMEhwrflaunch/parm" "config.EXPT=$EXPT" "config.startfile=$startfile"  
"config.HOMEhwrflaunch=$HOMEhwrflaunch" "config.case_root=$CASE_ROOT"  
"$HOMEhwrflaunch/parm/hwrflaunch_v3.7release.conf" "prelaunch.basin_overrides=no"  
"$HOMEhwrflaunch/parm/hwrflaunch_other_basins.conf" "config.run_ocean=yes"
```

Wrappers to be run for this example are the same as for the EP configuration without inner core data.

Alternatively, to run an uncoupled forecast with full AL configuration for any basin, pass the following arguments:

```
prelaunch.basin_overrides=no  
run_ocean=no
```

Run this configuration with the following wrappers as shown in Fig 3.4:

1. `launcher_wrapper`
2. `init_gdas_wrapper`  
`init_gfs_wrapper`  
`bufrrprep_wrapper`
3. `relocate_wrapper`
4. `gsi_d02_wrapper`  
`gsi_d03_wrapper`
5. `merge_wrapper`
6. `unpost_wrapper`
7. `forecast_wrapper`  
`post_wrapper`<sup>†</sup>  
`products_wrapper`<sup>†</sup>

<sup>†</sup>Wrapper can run at the same time as `forecast_wrapper`, but should only be submitted after the forecast job has started, i.e., no longer in queue.

Note: Do not forget to edit the `forecast_wrapper` for the appropriate number of processors.

### 3. Running HWRF

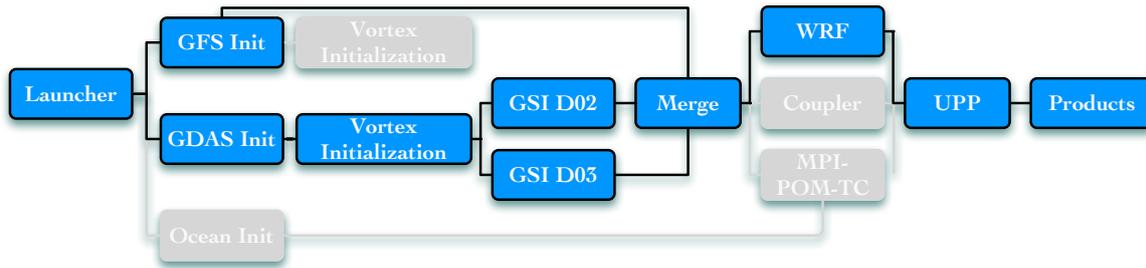


Figure 3.4: Components of HWRF that run for uncoupled configuration.

#### 3.8.2 Running with Optional GSI

A user may choose to run with or without data assimilation in any given basin. To force HWRF to run with GSI in any basin while retaining the other operational settings, set the following variables:

```
prelaunch.basin_overrides=no  
run_gsi=yes
```

Also pass the appropriate basin-specific configuration file listed in the "Extra conf" column of table 3.1 to `exhwrf_launch.py`.

To run without GSI in a full AL configuration for any basin, the following arguments should be passed to `exhwrf_launch.py`.

```
prelaunch.basin_overrides=no  
run_gsi=yes
```

These arguments will set up the configuration shown in Fig. 3.5, and the following wrappers should be run:

1. `launcher_wrapper`
2. `init_gfs_wrapper`  
`init_ocean_wrapper`
3. `relocate_wrapper`
4. `unpost_wrapper`
5. `forecast_wrapper`  
`post_wrapper`<sup>†</sup>  
`products_wrapper`<sup>†</sup>

<sup>†</sup>Wrapper can run at the same time as `forecast_wrapper`, but should only be submitted after the forecast job has started, i.e., no longer in queue.

### 3. Running HWRF

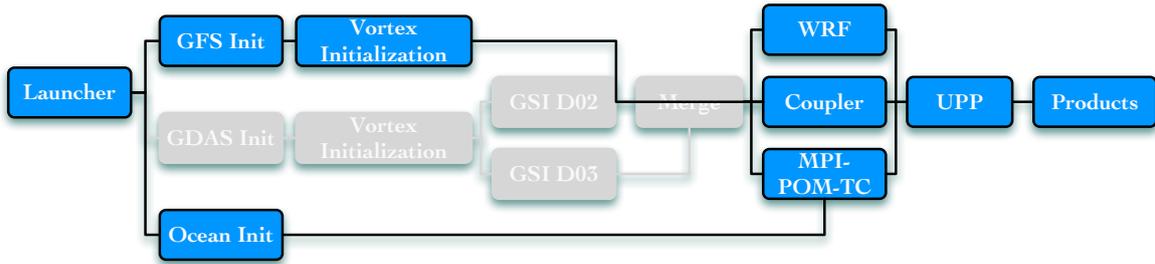


Figure 3.5: Components of HWRF that run with GSI turned off.

#### 3.8.3 Running without Vortex Initialization

The HWRF scripts require vortex relocation to be run before data assimilation can be performed. A feature introduced in HWRF v3.7a allows users to turn off the vortex adjustment component of the vortex initialization, and allow for only vortex relocation. Both options require `config.run_relocate=yes`. The configuration variable `relocate.initopt` controls the initialization procedure and should be set to "0" for full vortex initialization, or "1" for relocation only. Pass the `relocate.initopt` option to `exhwrflaunch.py` to change the relocation behavior.

To turn vortex initialization off completely from the full AL configuration for any basin, data assimilation must also be turned off, and the following arguments must be passed to `exhwrflaunch.py`:

```
run_gsi=no
run_relocate=no
```

There is no need to turn off `prelaunch.basin_overrides` for this case.

This configuration corresponds to that shown in Fig. 3.6 and requires the following wrappers:

1. `launcher_wrapper`
2. `init_gfs_wrapper`  
`init_ocean_wrapper`
3. `unpost_wrapper`
4. `forecast_wrapper`  
`post_wrapper`<sup>†</sup>  
`products_wrapper`<sup>†</sup>

<sup>†</sup>Wrapper can run at the same time as `forecast_wrapper`, but should only be submitted after the forecast job has started, i.e., no longer in queue.

### 3. Running HWRP

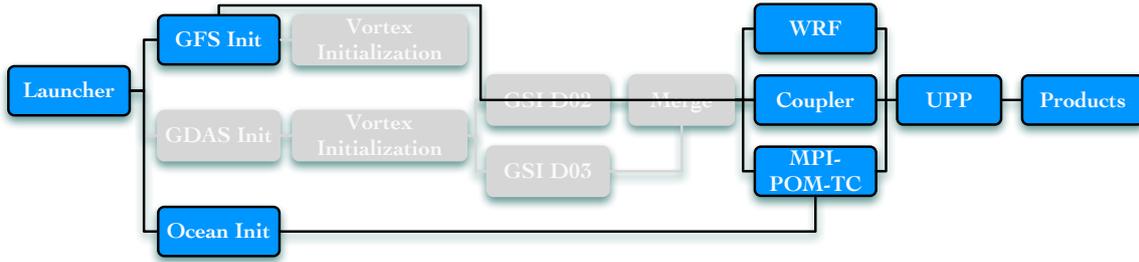


Figure 3.6: Components of HWRP that run without vortex initialization.

#### 3.8.4 Running without Spectral Files (GRIB Only)

To run with GRIB input files only, the following arguments should be passed to `exh-wrf_launch.py`.

```
use_spectral=no
```

When running with GRIB files only, data assimilation needs to be disabled or used in 3DVAR-only mode (hybrid data assimilation is not supported since it depends on the ingesting the spectral ensemble files). Therefore, one of the following two options must be adopted when using `use_spectral=no`:

##### **No data assimilation:**

```
config.run_gsi=no
```

See section 3.8.2 for details on running these configurations.

##### **3DVAR data assimilation using the default AL configuration in any basin:**

```
prelaunch.basin_overrides=no  
config.run_gsi=yes  
gsi_d02_nml.HYBENS_REGIONAL=F  
gsi_d03_nml.HYBENS_REGIONAL=F
```

Also set `gsi_d02.use_gfs_stratosphere=no` if you do not have GDAS forecast spectral files.

#### 3.8.5 Running with 27/9/3-km Configuration

It is possible to run v3.7a with 27/9/3-km resolution, the resolution used in v3.5a (2013) and v3.6a (2014). The rest of the system will be the same. Table 3.2 shows the settings for 27/9/3-km configuration for all the ocean basins. Add the relevant configuration file to the arguments in `launcher_wrapper` and set `prelaunch.basin_overrides=no`.

### 3. Running HWRF

<b>Basin</b>	<b>Ocean</b>	<b>Data assimilation</b>	<b>Vertical levels</b>	<b>Model top (hPa)</b>	<b>Extra configuration file</b>
<b>AL</b>	POM	Only if TDR available	61	2	hwrf_3km.conf
<b>EP</b>	POM	Only if TDR available	61	2	hwrf_3km_EP.conf
<b>NC Pac</b>	None	No	61	2	hwrf_3km_CP.conf
<b>NW Pac</b>	None	No	43	50	hwrf_3km_other_basins.conf
<b>N Ind</b>	None	No	43	50	hwrf_3km_other_basins.conf
<b>S Pac</b>	None	No	43	50	hwrf_3km_other_basins.conf
<b>S Ind</b>	None	No	43	50	hwrf_3km_other_basins.conf
<b>S Atl</b>	None	No	43	50	hwrf_3km_other_basins.conf

Table 3.2: Settings for all basins with 27/9/3 km configuration.

### *3. Running HWRf*

#### 3.8.6 Running with 43 Vertical Levels

HWRf v3.7a runs with 43 vertical levels by default for basins other than AL, EP, and CP. To use this option for for any configuration, pass `prelaunch.basin_overrides=no` and the `parm/hwrf_43lev.conf` as arguments to `exhwrf_launch.py`.

Note that if reduced horizontal resolution and vertical resolution to 27/9/3 km with 43 vertical levels is a desired configuration, `hwrf_3km.conf` must be passed after the `hwrf_43lev.conf` file.



# 4

## **HWRF Preprocessing System**

### 4.1 Introduction

HWRF needs data from the operational GFS and GDAS for its initialization procedures. Ultimately, the GFS dataset is used to create initial and boundary conditions for the 18-km outer domain, while the GDAS dataset is used to initialize the inner 6- and 2-km domains. However, as we explain later, the GDAS analysis and forecast are also used in the 18-km domain for an intermediate step.

The GFS analysis and forecast employed are from the same cycle as the HWRF initialization (e.g., to initialize a HWRF forecast at 12 UTC, the 12 UTC run of GFS is used). However, the GDAS data used by HWRF are forecasts from the previous 6-h cycle (e.g., to initialize a HWRF forecast at 12 UTC, the forecasts from 06 UTC run of GDAS are used). This differentiation is related to the data assimilation procedures (Chapter 6). First of all, data assimilation in HWRF is only conducted in the inner 6- and 2-km domains. A forecast from the GDAS initialized 6-h before the HWRF analysis provides a better first guess than the GDAS analyses at the same initialization time as HWRF, in which observations have already been included as part of the global data assimilation procedures. Second, the HDAS ingests observations in a 3-h time window centered in the HWRF analysis time. This requires the availability of three first-guess files, one at the HWRF analysis time, one before and one after. By using the GDAS forecast initialized 6-h before the HWRF analysis, HWRF can make use of the GDAS 3-, 6-, and 9-h forecast lead times, which are valid at 3 h before the HWRF initialization, at the time of the HWRF initialization, and 3 h after the HWRF initialization, respectively. This procedure is termed FGAT, or First Guess at Appropriate Time.

HWRF employs the WRF model to downscale the GDAS forecasts to the 6- and 2-km grids

#### 4. HWRF Preprocessing System

	<b>D01</b>	<b>D02</b>	<b>D03</b>
<b>Grid spacing (deg)</b>	0.135 (18 km)	0.045 (6 km)	0.015 (2 km)
<b>HWRF Forecast</b>	216 x 432 80°x 80°	106 x 204 12°x 12°	198 x 354 7.1°x 7.1°
<b>Analysis run</b>	216 x 432 80°x 80°	106 x 204 12°x 12°	198 x 354 7.1°x 7.1°
<b>Ghost run</b>	216 x 432 80°x 80°	166 x 336 26°x 26°	250 x 500 13°x 13°
<b>3X domain</b>			748 x 1504 30°x 30°

Table 4.1: Resolution (first row), number of grid points (top number in cell), and size (bottom number in cell) of the HWRF atmospheric grids.

for the vortex relocation and data assimilation procedures. This is done by using preprocessing utilities to interpolate the GDAS datasets to the HWRF 18-km domain, and then running two uncoupled 90-s WRF runs with three domains. These runs output "analysis" files, which are WRF restart files at  $t=0$  (analysis time), and are referred to as the WRF Analysis run and the WRF Ghost run.

In HWRF operations at the National Weather Service, it is important that safeguards are put in place to prevent model failure. To account for the possibility that the GDAS dataset may be unavailable, WRF Ghost and WRF Analysis runs using the GFS dataset as initial conditions are also performed and can be used as a backup. In general, GDAS data is available, and the WRF Ghost and WRF Analysis runs initialized from the GFS are not ultimately used in the forecast process.

The WRF Analysis run has the main purpose of downscaling the global information to the HWRF high-resolution grids for use in the vortex relocation procedure discussed in further detail in Chapter 5. The WRF Ghost run downscales the global information to high-resolution grids that are slightly larger than their WRF forecast domain counterparts to provide first-guesses for the data assimilation procedure discussed further in Chapter 6. Depending on the domain, the WRF Ghost run is referred to as Ghost domain 2 (ghost\_d02, the d02 counterpart) or Ghost domain 3 (ghost\_d03, the d03 counterpart). Table 4.1 describes the grid spacing and size of the domains used in the HWRF main forecast run, the WRF Analysis, and the WRF Ghost runs. These domains are shown in Figure 4.1.

The HWRF includes two preprocessing packages to generate input files for WRF, the WRF Preprocessing System (WPS) and prep\_hybrid. WPS consists of three programs to process input: geogrid interpolates static geographical data to the three HWRF domains; ungrib extracts meteorological fields from GRIB-formatted files and writes the fields to intermediate files; and metgrid horizontally interpolates the meteorological fields extracted by ungrib to the parent HWRF grid. The prep\_hybrid utility horizontally interpolates the atmospheric fields represented as spectral coefficients in the global model files in binary format and native sigma vertical levels to the parent HWRF grid. The output of prep\_hybrid and metgrid are utilized by the program real\_nmm to vertically interpolate meteorological information to the HWRF vertical levels, resulting in a full set of 3D initial conditions that constitute

#### 4. HWRF Preprocessing System

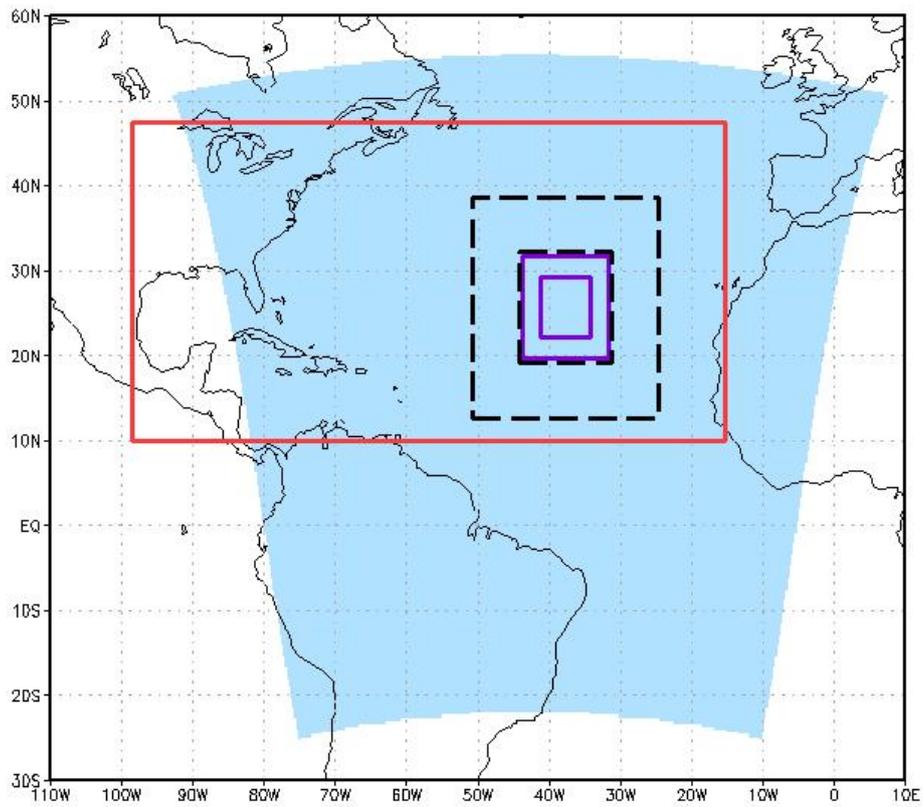


Figure 4.1: Example of the domains used by HWRF in the AL. The blue region is the outer 18-km domain. The purple solid boxes show the sizes of the vortex-following 6-km and 2-km domains, while the black dashed lines are the ghost domains for d02 and d03. The red box is the unified Atlantic MPIPOM-TC domain.

#### 4. HWRF Preprocessing System

the required WRF input. Both `prep_hybrid` and WPS are required because `prep_hybrid` only processes atmospheric data, so WPS is used to supply the initial conditions for soil temperature and moisture, as well as to supply the WRF model with the static information (topography, vegetation, etc.).

For general information about working with WPS, see the WRF-NMM documentation at:

<http://www.dtcenter.org/HurrWRF/users/docs/index.php>

As part of the vortex initialization procedure described in Chapter 5, the vortex from the processed global model analysis is removed and substituted with an improved vortex. To locate the vortex in the global model, the GFDL Vortex Tracker is run at the analysis time on the postprocessed 90-s output from the WRF Analysis run.

This chapter explains how to run the initialization procedure, including the launcher, WPS, `prep_hybrid`, `real_nmm`, WRF Analysis, and WRF Ghost to create the HWRF preliminary initial conditions.

## 4.2 Scripts

Three wrapper scripts are used to preprocess data for the atmospheric component of HWRF:

```
launcher_wrapper  
init_gdas_wrapper  
init_gfs_wrapper
```

The launcher wrapper calls `scripts/exhwrflaunch.py` to read the configuration files, set the output directory structures, and determine the location of the outer 18-km domain. The "init" wrapper scripts call the Python script `scripts/exhwrfinit.py`. For processing the GFS data, `exhwrfinit.py` is called once, and then three more times for processing GDAS data - once at each FGAT hour (3, 6, 9). Figures 4.2, 4.3, and 4.4 show the simplified outline of the processes that occur at each FGAT hour for both the GFS and GDAS initialization. The script `exhwrfinit.py` runs the three stages of WPS (geogrid, ungrib, and metgrid), `prep_hybrid`, `real_nmm` (to create initial and boundary conditions for the WRF Ghost and WRF Analysis runs), `wrfghost`, `wrfanalysis`, `post`, `gribber`, `tracker`, and `realcst` (to create LBCs for the main forecast run). All these steps are needed for WRF initialization, and some of these are used again at later stages of the run (for example, `post` and `tracker`).

## HWRP Initialization - 3 h Prior

exhwrp\_init.py for GDAS FGAT=3

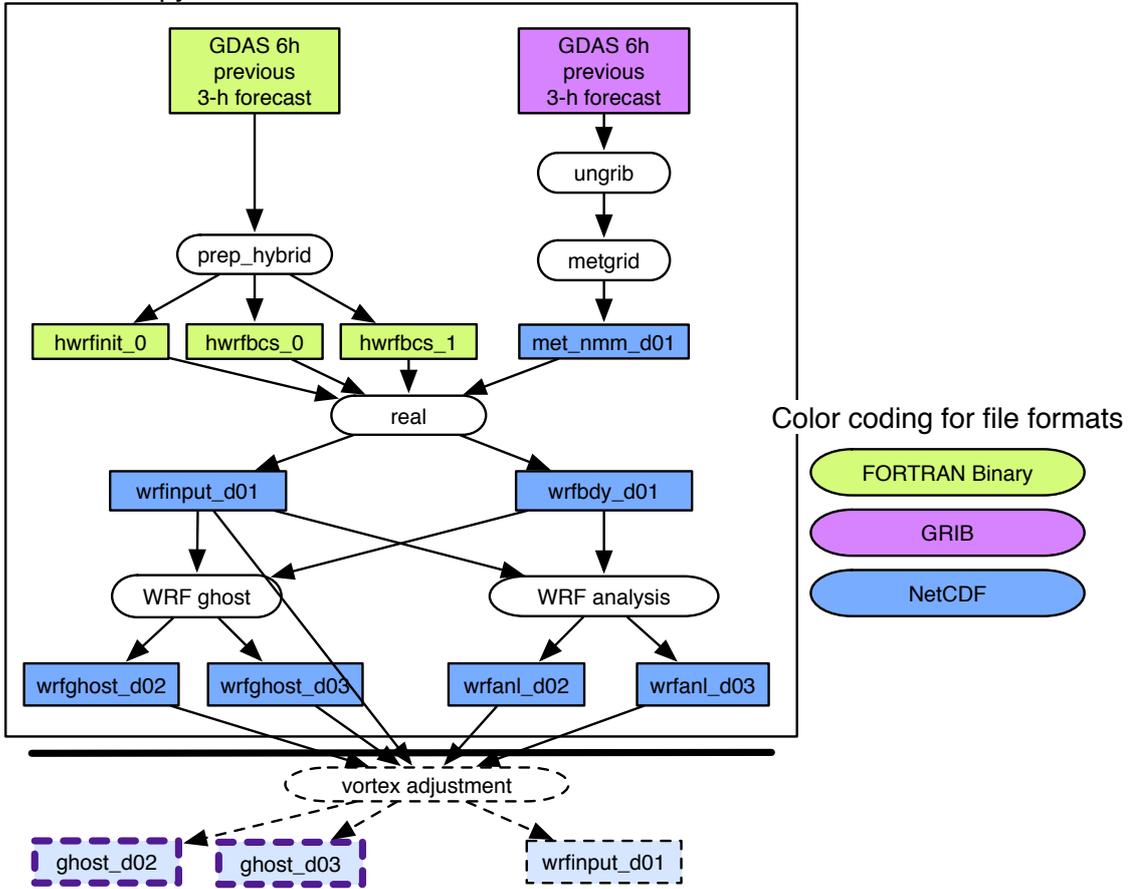


Figure 4.2: Simplified initialization procedures for the FGAT=3 valid 3 hours prior to HWRP initialization. All processes in the black box are run by calling `exhwrp_init.py` for GDAS at FGAT=3. Boxes with dashed outlines indicate modules and resulting files that are discussed in Chapter 5. Files that are outlined in heavy purple (dashed or solid) are used by subsequent processes described by Figure 6.1.

### HWRF Initialization - Analysis Time

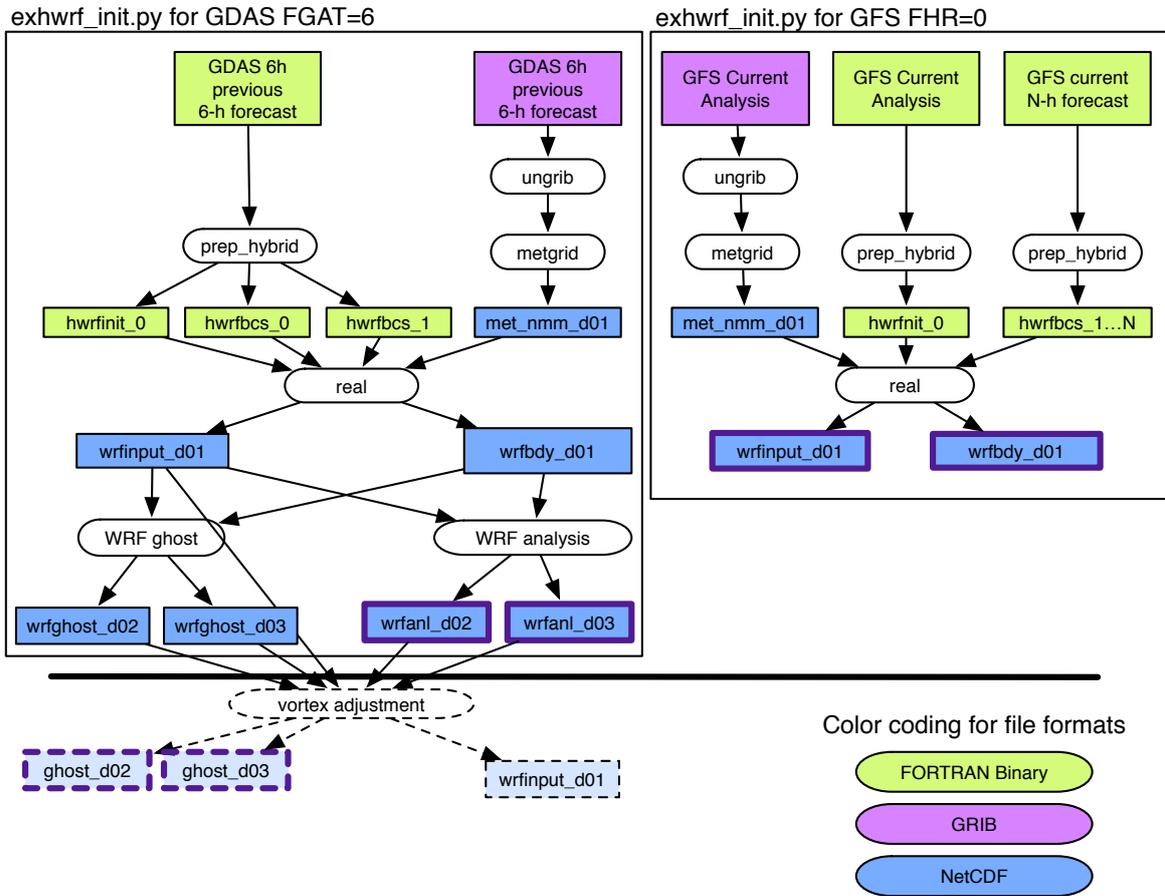


Figure 4.3: Simplified initialization procedures for the HWRF analysis time. All processes in the black box on the left are run by calling `exhwrfinit.py` for GDAS at FGAT=6, while the right black box are procedures from running `exhwrfinit.py` for GFS at analysis time. Boxes with dashed outlines indicate modules and resulting files that are discussed in Chapter 5. Files that are outlined in heavy purple (dashed or solid) are used by subsequent processes described by Figure 6.1.

## HWRF Initialization - 3 h After Analysis

exhwrflnit.py for GDAS FGAT= 9

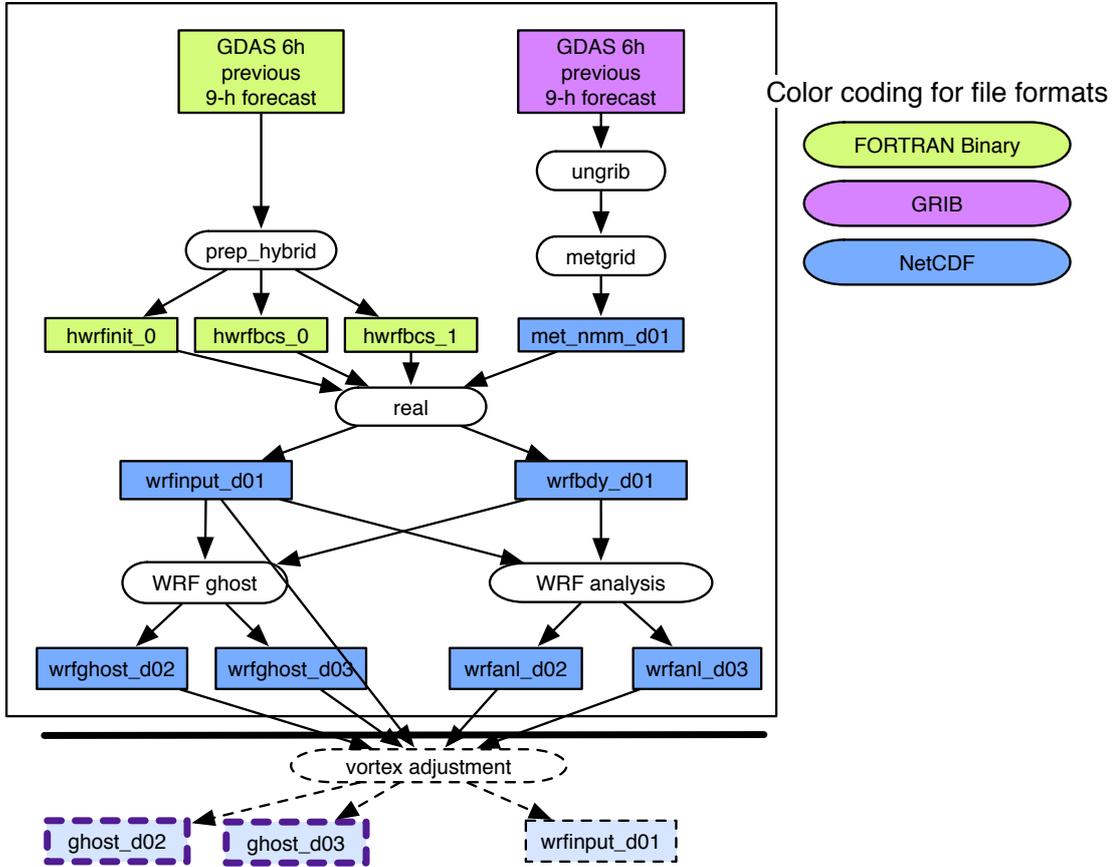


Figure 4.4: Simplified initialization procedures for the FGAT=9 valid 3 hours after HWRF initialization. All processes in the black box are run by calling `exhwrflnit.py` for GDAS at FGAT=9. Boxes with dashed outlines indicate modules and resulting files that are discussed in Chapter 5. Files that are outlined in heavy purple (dashed or solid) are used by subsequent processes described by Figure 6.1.

### 4.2.1 Overview of `exhwrflaunch.py`

1. Read in configure files from `parm/` directory
2. Set the paths to the directories containing HWRF source code and to the Python scripts (`$HOMEhwrfl` and `$USHhwrfl`, respectively)
3. Set the storm ID
4. Initialize the directory structure for the HWRF workflow
5. Locate and extract TC Vitals for the current storm and cycle, then write information to `$WORKhwrfl`
6. Using TC Vitals, determine the domain center and write output to file `storminfo.hwrfl_domain_center`

#### 4. HWRF Preprocessing System

7. Parse the configuration files and write configure and holdvars files to `com/`
8. Write a startfile to the launch directory (i.e., `wrappers/`)

##### Output files:

<code>\$startfile</code>	Name is defined in <code>global_vars.ksh</code> . Contains environment variables for the storm ID and paths to output data
<code>storm1.conf</code>	Located in <code>com/</code> directory. Contains configuration information compiled from all configure files
<code>storm1.holdvars.txt</code>	Located in <code>com/</code> directory. Only used in operational implementation of HWRF
<code>SID.YYYYMMDDHH.domain.center</code>	Coordinates of the domain center. <code>SID</code> is the storm ID, i.e., 08L for Gonzalo (2014), and <code>YYYYMMDDHH</code> is the analysis time

##### Status check:

The file `$startfile` defined in `global_vars.ksh` has been written in the `wrappers/` directory.

##### Usage:

To run `exhwrflaunch.py`, several arguments are required. Several arguments are provided within the wrapper, but it does not constitute an exhaustive list of options. Many additional arguments may be included with the submission of `exhwrflaunch.py` by editing the `launcher_wrapper`. The `launcher_wrapper` included with HWRF v3.7a submits `exhwrflaunch.py` with the following arguments:

```
${HOMEhwrfl}/scripts/exhwrflaunch.py YYYYMMDDHH STID HISTORY \  
${HOMEhwrfl}/parm config.EXPT=EXPT config.startfile=STARTFILE \  
config.HOMEhwrfl=${HOMEhwrfl} config.case_root=HISTORY \  
${HOMEhwrfl}/parm/hwrf_v3.7release.conf
```

In the previous command, `YYYYMMDDHH` should be replaced by the date to run, `STID` represents the storm ID, i.e., 08L for Gonzalo (2014), and `STARTFILE` is the launcher output file. The wrapper passes many of the values through environment variables, but values can be directly passed without the use of variables.

It is important to note that variables found in the configure files can also be passed in the command line, and do not need to be changed in the configure files. Yet another option is to use the configure files in the `parm/` directory as a template by which to create your own configuration, and pass the new file as an argument to the launch script by appending the following argument:

```
${HOMEhwrfl}/parm/my_hwrf_config.conf
```

Note that only those options meant to be changed need to be included in user-specific configuration files. The entire file should not be copied over.

## 4.2.2 Overview of the Init Scripts: `exhwrf_init.py` and Wrappers

In the following list, the top-level numbered items describe calls from the wrapper, either `init_gdas_wrapper` or `init_gfs_wrapper`, the alphabetic level represents calls from the script `exhwrf_init.py`, and the lowest level are calls to modules from within `init.py`.

1. Initialize the objects used to run all components of HWRF by calling `hwrf_expt.init_module()`
2. Call `exhwrf_init.py` to process the global input files using `INIT_PARTS=ALL` four times: once for GFS at analysis time from `init_gfs_wrapper`, then once for each FGAT hour for GDAS from `init_gdas_wrapper`
  - a) Run initialization steps leading up to the WRF Analysis (`run_through_anl()`):
    - i. `geogrid`
    - ii. `ungrib`
    - iii. `metgrid`
    - iv. `prep_hybrid`
    - v. `realinit`
    - vi. `runwrfanl`
  - b) Run additional initialization steps needed for data assimilation (`run_init_after_anl()`):
    - i. `ghost`
    - ii. `post`
    - iii. `gribber`
    - iv. `tracker`
  - c) Run steps necessary to generate the LBCs for the main forecast (`run_real_bdy()`):
    - i. `ungrib`
    - ii. `metgrid`
    - iii. `prep_hybrid`
    - iv. `realfcst`

## 4.2.3 Overview of Initialization Modules

### Geogrid ---

The run module for a Geogrid object resides in `ush/hwrf/wps.py`. The module performs the following tasks:

1. Link the `GEOGRID.TBL` and `geog_data/` fixed files
2. Create the namelist
3. Run `geogrid.exe`

#### **Output files:**

#### 4. HWRF Preprocessing System

<code>geo_nmm.d01.nc</code>	Static geographical data for the parent domain, with grid spacing of 0.135 degrees.
<code>geo_nmm_nest.101.nc</code>	Static geographical data that covers the parent domain, with grid spacing of 0.045 degrees.
<code>geo_nmm_nest.102.nc</code>	Static geographical data that covers the parent domain, with grid spacing of 0.015 degrees.

#### Status Check:

In the standard out file of either initialization task, you will find the line "INFO: WPS Geogrid completed.", once for gfsinit and once for each fgat hour of gdasinit.

#### Executables:

`geogrid.exe`

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

**INPUT:** Fix files from `${GEOG_DATA_PATH}`  
`GEOGRID.TBL` - defines parameters of input data sets  
`namelist.wps` - WPS namelist

**OUTPUT:** `geo_nmm.d01.nc`  
`geo_nmm_nest.101.nc`  
`geo_nmm_nest.102.nc`

**USAGE:** `geogrid.exe`

#### PrepHybrid ---

The run module for the PrepHybrid object is located in `ush/hwrf/prep.py`. The module performs the following tasks:

1. Copy the input files
2. Run `hwrf_prep.exe`
3. Link the output files

#### Output files for ICs:

<code>hwrfinit_00</code>	Global model spectral data preprocessed by <code>hwrf_prep.exe</code> and ready to be used by <code>real_nmm</code> to generate preliminary HWRF ICs.
--------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

#### Output files in for LBCs:

#### 4. HWRF Preprocessing System

hwrfbcs00\_\${bc\_index}      Global model spectral data preprocessed by hwrf\_prep.exe and ready to be used by real\_nmm to generate preliminary HWRF LBCs. The variable bc\_index=0,1,...,n corresponds to the forecast files that need to be preprocessed to create the LBCs for the HWRF forecast.

#### Status Check:

In the standard output file of each initialization task, you will find the line "INFO: - exit status 0" for the task "prep\_hybrid", once for gfsinit and once for each fgat hour in gdasinit.

#### Executables:

hwrf\_prep.exe

**FUNCTION:**      Preprocesses the GDAS or GFS spectral data on vertical sigma levels in binary format for use the by real\_nmm

**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 file  
gfsbc\${bc\_index} - link to the global spectral file [gdas|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 12-h LBCs for the HWRF forecast initialized at 2012102806, the GFS initialized at the HWRF analysis time is used, and these variables would be set to the following.

```
BKG_START_TIME = 2012102806  
BKG_FCST_TIME = 012
```

**OUTPUT:**            hwrfinit\_0  
hwrfbcs00\_\${bc\_index}

**USAGE:**            **\$PREP\_EXE \$NX1 \$NY1 \$VERT\_LEV \$DXX \$DYY**  
where \$NX1, \$NY1, and \$VERT\_LEV are the output file grid dimensions in the meridional, zonal and vertical directions, and \$DXX and \$DYY are the horizontal grid spacing.

Ungrib \_\_\_\_\_

The run module for an Ungrib object resides in ush/hwrf/wps.py. The module performs the following tasks:

## 4. HWRF Preprocessing System

1. Create the namelist
2. Link the `vtable` and input GRIB files
3. Run `ungrib.exe`

### Output files:

The intermediate files written by `ungrib.exe` will have names of the form `FILE:YYYY-MM-DD_HH` (unless the prefix variable in `hwrf.conf` was set to a prefix other than "FILE").

### Status Check:

In the standard out file of each initialization task, you will find the line "INFO: WPS Un-grib completed.", once for `gfsinit` and once for each `fgat` hour of `gdasinit`.

### Executables:

`ungrib.exe`

**FUNCTION:** Extracts meteorological fields from GRIB formatted files and writes the fields to intermediate files

**INPUT:** GRIB files  
`vtable` - codes to interpret GRIB files  
`namelist.wps` -WPS namelist

**OUTPUT:** `FILE:YYYY-MM-DD_HH`

**USAGE:** `ungrib.exe`

## Metgrid ---

The run module for a Metgrid object resides in `ush/hwrf/wps.py`. The module performs the following tasks:

1. Create the namelist
2. Link the metgrid table
3. Copy in the output from `geogrid` and `ungrib`
4. Run `metgrid.exe`

### Output files:

`met_nmm.d01.YYYY-MM-DD_HH:MM:SS.nc`

`YYYY-MM-DD_HH:MM:SS` refers to the valid date of the interpolated data in each file.

### Status Check:

#### 4. HWRF Preprocessing System

In the standard out file of each initialization task, you will find the line "INFO: WPS Met-grid completed.", once for gfsinit and once for each fgat hour of gdasinit.

##### Executables:

metgrid.exe

**FUNCTION:** Horizontally interpolates the meteorological fields extracted by ungrib to the model parent grid

**INPUT:** METGRID.TBL - parameters for interpolating each field  
geo\_nmm.d01.nc - output of geogrid  
namelist.wps - WPS namelist  
FILE: YYYY-MM-DD\_HH - output of ungrib

**OUTPUT:** met\_nmm.d01.YYYY-MM-DD\_HH:MM:SS.nc

**USAGE:** **metgrid.exe**

#### Realinit ---

Realinit is a WRFTask object and its run module resides in `ush/hwrf/wps.py`. The module performs the following tasks:

1. Link the input and fixed files
2. Run `hwrf_swcorner_dynamic.exe` to calculate the `istart` and `jstart` values for a nest
3. Generate the namelist
4. 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.

##### Output files:

<code>wrfinput_d01</code>	ICs created from GDAS
<code>wrfbdy_d01</code>	LBCs created from GFS for the ghost and analysis runs
<code>fort.65</code>	High-resolution sea mask data

##### Status Check:

In the standard output file for gfsinit, you will find the line "INFO: gfsinit/realinit: completed". Similarly for gdasinit standard output, you will find the line "INFO: fgat.tYYYYMMDDHH/realinit: completed" where YYYYMMDDHH is the date string for each fgat hour valid time.

#### 4. HWRF Preprocessing System

##### Executables:

`real_nmm.exe`

**FUNCTION:** Generates the initial and boundary conditions

**INPUT:** Fixed files<sup>†</sup>  
`hwrfbcs_[0-21]` - BC output of `prep_hybrid`  
`hwrfinit_0` - IC output of `prep_hybrid`  
`geo_nmm.d01.nc` - d01 output from `geogrid`  
`geo_nmm.l01.nc` - d02 output from `geogrid`  
`geo_nmm.l02.nc` - d03 output from `geogrid`  
`met_nmm.d01.YYYY-MM-DD_HH:00:00.nc` - d01 output from `metgrid`

**OUTPUT:** `fort.65`  
`wrfbdy_d01`  
`wrfinput_d01`

**USAGE:** **`real_nmm.exe`**

`hwrf_swcorner_dynamic.exe`

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

**INPUT:** `storm.center` - storm center location  
`domain.center` - domain center location  
`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_swcorner_dynamic.exe`**

Realfcst \_\_\_\_\_

The process for `realfcst` is the same as `realinit`, except that `realfcst` runs for the length of the HWRF forecast, instead of only at the analysis time.

## Runwrfanl ---

Runwrfanl is a WRFAnl4Trak object whose run module resides in `ush/hwrf/fcsttask.py`. The module performs the following tasks:

1. Link the input and fixed files
2. Run `hwrf_swcorner_dynamic.exe` to calculate the `istart` and `jstart` values for a nest
3. Generate the `namelist.input` for grids identical to the HWRF forecast grids
4. Run `wrf.exe` to make a 90-s run of WRF and generate two analysis output files

### Output files:

`trackin_d01` - storm location in the parent domain  
`wrfanl_d02_YYYY-MM-DD_HH_00_00` - "analysis" file for 9-km nest  
`wrfanl_d03_YYYY-MM-DD_HH_00_00` - "analysis" file for 3-km nest

### Status Check:

In the standard output file, you will find the line "INFO: gfsinit/wrfanl: completed".

### Executables:

`wrf.exe`

FUNCTION: Atmospheric model component of HWRF

INPUT: Fixed files<sup>†</sup>  
`geo_nmm.d01.nc` - d01 output from geogrid  
`geo_nmm.l01.nc` - d02 output from geogrid  
`geo_nmm.l02.nc` - d03 output from geogrid  
`wrfinput_d01` - ICs for parent domain from realinit  
`wrfbdy_d01` - LBCs for parent domain from realinit  
`namelist.input` - WRF namelist

OUTPUT: `trackin_d01`  
`wrfanl_d02_YYYY-MM-DD_HH_00_00`  
`wrfanl_d03_YYYY-MM-DD_HH_00_00`

USAGE: **wrf.exe**

## Runghost ---

Runghost is a WRFGhost object whose run module resides in `ush/hwrf/fcsttask.py`. The module performs the following tasks:

1. Link the input and fixed files

#### 4. HWRF Preprocessing System

2. Run `hwrf_swcorner_dynamic.exe` to calculate the `istart` and `jstart` values for a nest
3. Generate the `namelist.input` for the ghost domains
4. Run `wrf.exe` to make a 90-s run of WRF and generate two analysis output files

#### Output files:

`wrfanl_d02_YYYY-MM-DD_HH_00_00` - "ghost" analysis file for 9-km nest  
`wrfanl_d03_YYYY-MM-DD_HH_00_00` - "ghost" analysis file for 3-km nest

#### Status Check:

In the standard output file of `gfsinit`, you will find the line "INFO: gfsinit/ghost: completed". Similarly for `gdasinit`, you will find the line "INFO: fgat.tYYYYMMDD/ghost:", where `YYYYMMDDHH` is the date string of the FGAT hour valid time.

#### Executables

`wrf.exe`

**FUNCTION:** Atmospheric model component of HWRF

**INPUT:** Fixed files<sup>†</sup>  
`geo_nmm.d01.nc` - d01 output from geogrid  
`geo_nmm.l01.nc` - d02 output from geogrid  
`geo_nmm.l02.nc` - d03 output from geogrid  
`wrfinput_d01` - ICs for parent domain from `realinit`  
`wrfbody_d01` - LBCs for parent domain from `realinit`  
`namelist.input` - WRF `namelist`

**OUTPUT:** `wrfanl_d02_YYYY-MM-DD_HH_00_00`  
`wrfanl_d03_YYYY-MM-DD_HH_00_00`

**USAGE:** `wrf.exe`

#### <sup>†</sup>Fixed files:

<code>aerosol.formatted</code>	<code>CAMtr_volume_mixing_ratio.RCP8.5</code>
<code>aerosol_lat.formatted</code>	<code>capacity.asc</code>
<code>aerosol_lon.formatted</code>	<code>CCN_ACTIVATE.BIN</code>
<code>aerosol_plev.formatted</code>	<code>CLM_ALB_ICE_DFS_DATA</code>
<code>bulkdens.asc_s_0_03_0_9</code>	<code>CLM_ALB_ICE_DRC_DATA</code>
<code>bulkascii.asc_s_0_03_0_9</code>	<code>CLM_ASM_ICE_DFS_DATA</code>
<code>CAM_ABS_DATA</code>	<code>CLM_ASM_ICE_DRC_DATA</code>
<code>CAM_AEROPT_DATA</code>	<code>CLM_DRDSDT0_DATA</code>
<code>CAMtr_volume_mixing_ratio.A1B</code>	<code>CLM_EXT_ICE_DFS_DATA</code>
<code>CAMtr_volume_mixing_ratio.A2</code>	<code>CLM_EXT_ICE_DRC_DATA</code>
<code>CAMtr_volume_mixing_ratio.RCP4.5</code>	<code>CLM_KAPPA_DATA</code>
<code>CAMtr_volume_mixing_ratio.RCP6</code>	<code>CLM_TAU_DATA</code>

## 4. HWRF Preprocessing System

coeff_p.asc	ozone_lat.formatted
coeff_q.asc	ozone_plev.formatted
constants.asc	RRTM_DATA
eta_micro_lookup.dat	RRTM_DATA_DBL
hwrp_track	RRTMG_LW_DATA
co2_trans	RRTMG_LW_DATA_DBL
ETAMPNEW_DATA	RRTMG_SW_DATA
ETAMPNEW_DATA_DBL	RRTMG_SW_DATA_DBL
ETAMPNEW_DATA.expanded_rain	SOILPARM.TBL
ETAMPNEW_DATA.expanded_rain_DBL	termvels.asc
GENPARM.TBL	tr49t67
grib2map.tbl	tr49t85
gribmap.txt	tr67t85
kernels.asc_s_0_03_0_9	URBPARM.TBL
kernels_z.asc	URBPARM_UZE.TBL
LANDUSE.TBL	VEGPARM.TBL
MPTABLE.TBL	wind-turbine-1.tbl
masses.asc	
ozone.formatted	

### Post ---

Post is a `PostOneWRF` object whose run module resides in `ush/hwrp/post.py`. This instance of the module runs `unipost.exe` on the output of the 90-s WRF Analysis run. The purpose of this step is to destagger the HWRF native output, interpolate it vertically to pressure levels, compute derived variables, and output the result in GRIB format. Further details about the post objects, modules, and executables can be found in Chapter 10.

#### Output:

In the `intercom/fgat.tYYYYMMDDHH/post/fgat.tYYYYMMDDHH/post` directory, you will find the following file:

```
fgat.tYYYYMMDDHH_post-moad.egrb
```

#### Status check:

The line `"INFO: state=COMPLETED"` can be found in the standard output files for `gdasinit` and `gfsinit` for the `"post"` task. Performing a search for both of quoted strings should return one line for each FGAT hour for GDAS, and once for GFS.

### Gribber ---

Gribber is a `GRIBTask` object and serves to regrib the output of `post`, interpolating the 18-km parent domain GRIB file to a  $20^{\circ} \times 20^{\circ}$  grid. This file is used as input to the GFDL Vortex Tracker. See Chapter 11 for more details about `GRIBTask` objects, modules, and executables.

## 4. HWRF Preprocessing System

### Output:

In the `intercom/fgat.tYYYYMMDDHH/regribber` directory, you will find the files below, where `stormname` and `sid` are the name of the storm and the SID in lower case (e.g., `gonzalo 081` for Hurricane Gonzalo):

```
quarter_degree.grb
{stormname}{sid}.YYYYMMDDHH.hwrftmk.grbf[-3,00,03]
{stormname}{sid}.YYYYMMDDHH.hwrftmk.grbf[-3,00,03].grbindex
subset.grb
```

### Status check:

In the standard output files for `gdasinit` and `gfsinit`, you will find the following string on the same line as "regribber" for each FGAT hour: "WARNING: No subtasks incomplete. I think I am done running. Will exit regribber now."

## Tracker ---

The GFDL Vortex Tracker is run on the GRIB file resulting from the `gribber` step above. The tracker object resides in `ush/hwrf/tracker.py`. More information about the tracker can be found in section 11.

### Output:

In the `intercom/fgat.tYYYYMMDDHHHH` directory, you will find the file:

```
gfs.track0.atcfunix - contains the storm center at initial time in the WRF analysis run output
```

### Status Check:

In the standard output files for `gdasinit` and `gfsinit`, you should find the line "WARNING: Successful return status from gettrk.", once for `gfsinit` and once for each FGAT hour in `gdasinit`.



# 5

## Vortex Relocation

### 5.1 Introduction

The atmospheric component of HWRF, WRF-NMM, needs ICs and LBCs to produce forecasts. The GFS and GDAS fields are used to create the preliminary atmospheric fields, which are further improved through the vortex adjustment procedures and data assimilation to provide the final IC to WRF.

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 observed storm intensity, location, and structure.

Initial conditions for HWRF d02 and d03 are created by ingesting GDAS 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 configuration is detailed in Chapter 4. Within the vortex relocation code, the fields are interpolated to the 3X domain, a temporary domain with  $0.015^\circ$  grid spacing. For historical reasons, some file names and executables use 4X when referring to the 3X domain.

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 5.1 describes Stages 1 and 2, and Figure 5.2 describes Stage 3. If the previous cycle HWRF forecast exists, and if the observed storm intensity is at least  $14 \text{ ms}^{-1}$ , HWRF is run in cycled mode. In cycled mode, the the 6-h forecast vortex from the previous HWRF cycle, adjusted according to the TC Vitals, is used for initializing the

## 5. Vortex Relocation

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  $20 \text{ ms}^{-1}$ , the GDAS vortex is adjusted and then used. Conversely, for storms with observed intensity greater than or equal to  $20 \text{ ms}^{-1}$ , 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-h 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 6-h forecast from the previous cycle (for cycled runs), from the GDAS, or from the bogus vortex is adjusted to match the observed location, intensity, and structure provided by the NHC for the current time. Then the vortex and environment fields are combined.

## 5. Vortex Relocation

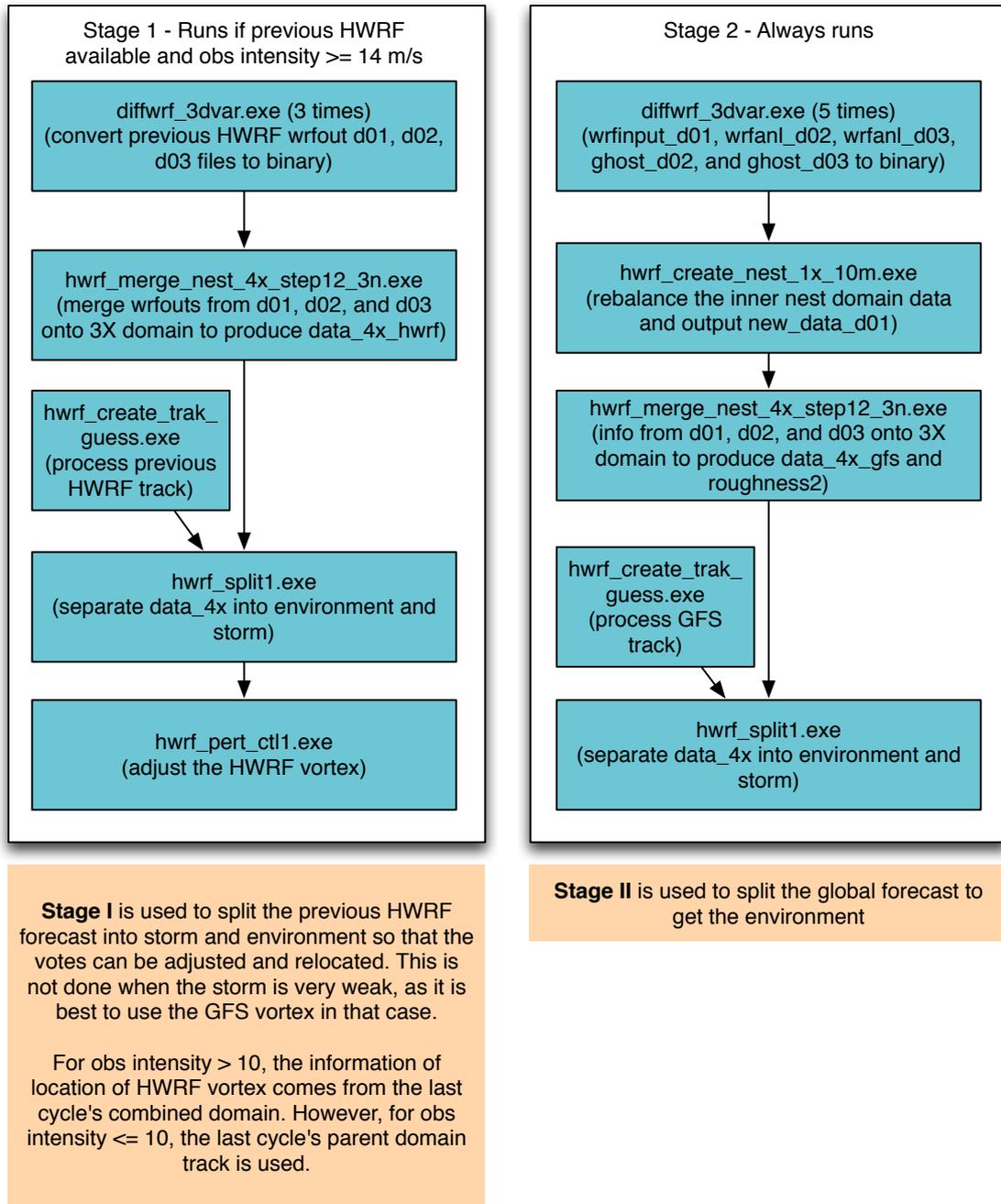


Figure 5.1: Simplified flow diagram of Stages 1 and 2 of the vortex relocation process.

## 5. Vortex Relocation

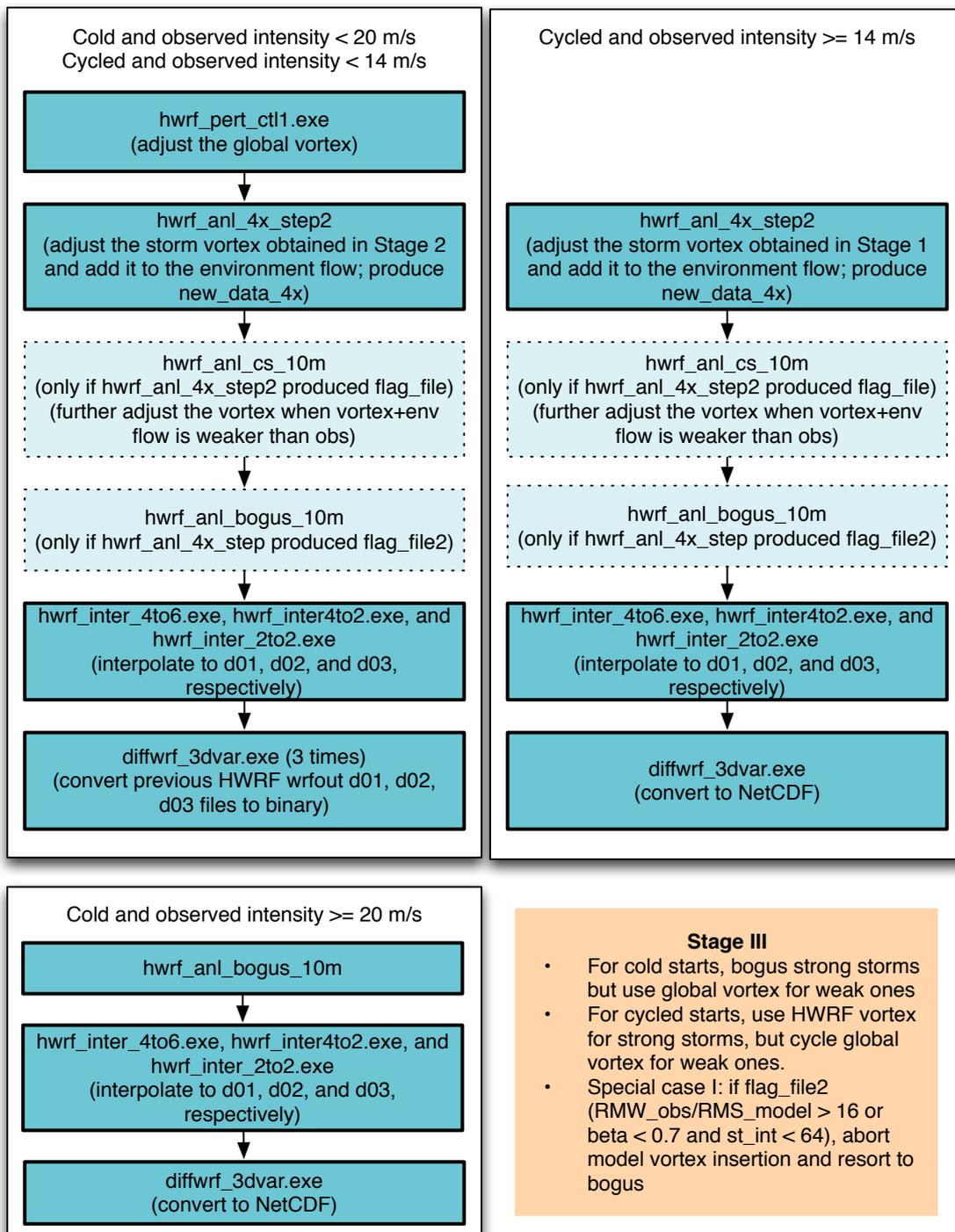


Figure 5.2: Simplified flow diagram of Stage 3 of the vortex relocation process.

## 5.2 Scripts

The vortex improvement procedure is entirely driven by the wrapper script `relocate_wrapper`, which calls 4 instances of `scripts/exhwrp_relocate.py`. The first instance runs `relocate` on the 90-s WRF Analysis run initialized from the GFS analysis. The other three instances run `relocate` on the 90-s WRF Analysis runs created at each FGAT time for the domains initialized by the GDAS forecasts. If the relocation procedure using the GDAS-derived input files is successful, the `relocate` results from GFS-derived fields are discarded.

### 5.2.1 Overview of `exhwrp_relocate.py`

The numbered items in the following list indicate calls made from the `exhwrp_relocate.py` script, while the lower level list items are calls made within the Python modules.

1. Initialize the objects used to run all components of HWRF by calling `hwrp_expt.init_module()`.
2. If GFS: run `relocate` at analysis time
  - a) Stage 1
  - b) Stage 2
  - c) Stage 3
3. If GDAS: run `relocate` for each FGAT time
  - a) Stage 1
  - b) Stage 2
  - c) Stage 3

### 5.2.2 Overview of the Relocate Modules

#### Stage 1 \_\_\_\_\_

1. Copy the fixed files and input files to the working directory.
2. Check if the HWRF forecast from the previous cycle exists, and if the storm intensity is greater than  $14 \text{ ms}^{-1}$ ; if not, continue to Stage 2.
3. Run `diffwrf_3dvar.exe` to convert the previous cycle forecast output `wrfout_d0[1-3]` into unformatted data files `old_hwrp_d0[1-3]` respectively.
4. Run `merge_nest_4x_step12_3n.exe` to merge `old_hwrp_d0[1-3]` onto 3X domain and produce a file containing the merged data: `data_4x_hwrp`.
5. Run `hwrp_create_trak_guess.exe` to produce a guess track (0,3,6,9 hour) for the current forecast using previous cycle forecast track.
6. Run `wrf_split1.exe` to separate `data_4x_hwrp` into two parts, an environment field (`wrf_env`) and a storm vortex (`storm_pert`). A storm radius data file

## 5. Vortex Relocation

(`storm_radius`) is also generated.

7. Run `hwrp_pert_ctl.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:

<code>storm_size_p</code>	Storm size information
<code>storm_pert_new</code>	New storm vortex after adjustments by <code>hwrp_pert_ctl.exe</code>
<code>storm_sym</code>	Symmetric part of the vortex
<code>storm_radius</code>	Storm radius information
<code>wrf_env</code>	Environment field

### Status Check:

If the line "INFO: Stage 1 completed" is found in the standard output, Stage 1 was successful.

### Executables:

`diffwrf_3dvar.exe`

This executable serves two functions, denoted by 1 or 2 below.

- FUNCTION:
1. Converts netCDF input to unformatted file (when first argument is "`storm_relocate`")
  2. Updates existing netCDF file with new unformatted file (when first argument is "`3dvar_update`")

- INPUT:
1. netCDF format input files or previous cycle 6-h forecast
  2. Unformatted file containing new vortex fields

- OUTPUT:
1. Unformatted data file
  2. Updated netCDF file

- USAGE:
1. `diffwrf_3dvar.exe storm_relocate input_file flnm3 \ output_file`  
The command above writes the WRF file `input_file` into an unformatted file, `output_file`, which will be used in the vortex relocation procedures.
  2. `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.

`hwrp_merge_nest_4x_step12_3n.exe`

- FUNCTION: Merges inner and outer domains onto a 3X domain

## 5. Vortex Relocation

**INPUT:** \$gesfhr (=6) - last digit of the input/output fort file, i.e., fort.26  
\$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)  
old\_hwrf\_d02 or new\_gfs\_d02 (fort.36)  
old\_hwrf\_d03 or new\_gfs\_d03 (fort.46)

**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**

hwrf\_create\_trak\_guess.exe

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

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

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

**USAGE:** **echo \$storm\_id \$ih \$BASIN | hwrf\_create\_trak\_guess.exe**

hwrf\_split1.exe

**FUNCTION:** Splits the vortex from the background (environmental) field

**INPUT:** \$gesfhr=6 - last digit of the input/output fort file, i.e., fort.26  
\$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.fn1.all (fort.30) - storm center guess  
old\_hwrf\_d01 (fort.46) - outer domain data

## 5. Vortex Relocation

**OUTPUT:** wrf\_env (fort.56) - environmental flow  
storm\_pert (fort.71) - separated 3D vortex field  
storm\_radius (fort.85) - average of model and observed storm radius  
rel\_inform.\$cdate (fort.52) - diagnostics file (obs-previous 6-h forecast)  
vital\_syn.\$cdate (fort.55) - information for generating bogus if storm not found in previous 6-h forecast

**USAGE:** **echo \$gesfhr \$ibgs \$st\_int \$BASIN | hwrp\_split.exe**

hwrp\_pert\_ct1.exe

**FUNCTION:** Adjusts storm vortex (storm\_pert)

**INPUT:** \$gesfhr (=6) - last digit of the input/output fort file, i.e., fort.26  
hdas\_atcfunix (fort.12) - storm track  
tcvitals.as (fort.11) - storm center obs  
wrf\_env (fort.26) - environmental flow (from hwrp\_split1.exe)  
storm\_pert (fort.71) - separated 3D vortex field (from hwrp\_split1.exe)  
roughness1 (fort.66) - sea-mask (1=sea, 0=land) and ZNT (roughness length) merged onto the 3X domain.

**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 | hwrp\_pert\_ct1.exe**

## Stage 2 \_\_\_\_\_

1. Copy the fix files and namelist.
2. Run `diffwrf_3dvar.exe` to convert `wrfinput_d0[1-3]` and `wrfghost_d0[2-3]` to binary files `new_gfs_d0[1-3]` and `new_ghst_d0[2-3]`, respectively.
3. Run `hwrp_create_trak_fnl.exe` to create `trak_fnl.all_gfs`, a guess track file from `atcfunix`.
4. Run `hwrp_merge_nest_4x_step12_3n.exe` to merge all three HWRF domains (`new_gfs_d0[1-3]`) 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`).
5. Run `hwrp_split1.exe` to separate the `data_4x_gfs` into environment data (`gfs_env`) and storm vortex (`storm_pert_gfs`). A file containing the storm radius information will be generated, too (`storm_radius_gfs`).

**Status Check:**

## 5. Vortex Relocation

In the standard output file, the line "INFO: Stage 2 completed" should exist.

### Output files:

<code>gfs_env</code>	environment fields from GFS data
<code>roughness2</code>	sea-mask and roughness length from GFS data
<code>storm_pert_gfs</code>	storm vortex from GFS data
<code>storm_radius_gfs</code>	storm radius information from GFS data

### Executables:

`diffwrf_3dvar.exe`

Refer to Stage 1 in section 5.2.2.

`hwrf_create_trak_fnl.exe`

Refer to Stage 1 in section 5.2.2.

`hwrf_merge_nest_4x_step12_3n.exe`

Refer to Stage 1 in section 5.2.2.

`hwrf_split1.exe`

Refer to Stage 1 in section 5.2.2.

## Stage 3 ---

**For a cold start or cycled start of a weak storm:** The vortex and environment are obtained from the global data.

1. Link the input and fixed files.
2. Run `hwrf_pert_ct1.exe` to adjust the GDAS (or GFS) vortex (`storm_pert_gfs` from Stage 2).
3. Run `hwrf_anl_4x_step2.exe` to adjust the storm vortex (`storm_pert_gfs1`) 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.
4. When the combined vortex and environment flow is weaker than observations, discard the new file (`new_data_4x`), and run `hwrf_anl_cs_10m.exe` to further adjust the analysis. This produces a new version of `new_data_4x` containing the combined environment flow and adjusted vortex.
5. Run `hwrf_inter_4to6.exe` to interpolate the `new_data_4x` from the 3X domain onto the outermost HWRf grid. This will produce the new `data_merge_d01`. Input file for storm radius is `storm_radius_gfs`.
6. Run `hwrf_inter_2to2.exe` to interpolate the `new_data_4x` from the 3X domain onto the `ghost_d02` grid. This will produce the new `data_merge_g02`.
7. Run `hwrf_inter_2to2.exe` to interpolate the `new_data_4x` from the 3X domain onto the `ghost_d03` grid. This will produce the new `data_merge_g03`.

## 5. Vortex Relocation

8. Run `diffwrf_3dvar.exe` to convert the merged data files (`data_merge_d01` and `data_merge_g0[2-3]`) to NetCDF files (`wrfinput_d01` and `wrfghost_d0[2-3]`).

**For a cycled start of a strong storm:** Performs all steps from cold/cycled weak storm except for Step 2. For a strong cycled storm, `hwrf_pert_ctl.exe` runs in Stage 1 and the vortex is taken from the previous HWRf forecast.

**For a cold start of a strong storm:** The vortex is an adjusted bogus vortex.

1. Link the input and fixed files.
2. Run `hwrf_anl_bogus_10m.exe` to create a bogus vortex and add it to the environment.
3. Perform Steps 5-8 of the weak storm procedure.

### Status Check:

The line "INFO: Stage 3 completed" exists in the standard output.

### Executables:

`hwrf_pert_ctl.exe`

Refer to Stage 1 in section 5.2.2.

`hwrf_anl_4x_step2.exe`

**FUNCTION:** Adjusts the storm vortex and adds the new storm vortex to the environment flow on the 3X domain grid

**INPUT:** `$gesfhr (=6)` - last digit of the input/output fort file, i.e., fort.26  
`storm_size_p (fort.14)` - from `hwrf_pert_ctl.exe`  
`tcvitals.as (fort.11)` - storm center obs  
`hdas_atcfunix (fort.12)` - input track file from previous 6-h forecast  
`storm_sym (fort.23)` - symmetric part of storm  
`gfs_env (fort.26)` - GFS environmental flow  
`roughness1 (fort.46)` - roughness output from Stage 2 executable  
`merge_nest_4x_step2.exe`  
`storm_pert_new (fort.71)` - adjusted storm perturbation from `hwrf_pert_ctl.exe`

**OUTPUT:** `wrf_env_new (fort.36)` - new environmental flow  
`new_data_4x (fort.56)` - adjusted vortex plus environment on 3X domain

**USAGE:** `echo $gesfhr $BASIN 0 1 | hwrf_anl_4x_step2.exe`

`hwrf_anl_cs_10m.exe`

## 5. Vortex Relocation

**FUNCTION:** Further adjusts the storm vortex when combined vortex plus environmental flow is less than the observed maximum wind speed

**INPUT:** \$gesfhr (=6) - last digit of the input/output fort file, i.e., fort.26  
tcvitals.as (fort.11) - observed storm center  
wrf\_env\_new (fort.26) - new environmental flow (from hwrf\_anl\_4x\_step2.exe)  
storm\_sym (fort.23) - symmetric part of storm (from hwrf\_pert\_ctl.exe)  
roughness (fort.46) - roughness info for boundary layer calculation (from hwrf\_merge\_nest\_4x\_step2.exe)  
storm\_radius (fort.85) - from wrf\_split.exe  
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**

hwrf\_inter\_4to6.exe

**FUNCTION:** Interpolates from 3X domain onto outer domain

**INPUT:** \$gesfhr (=6) - last digit of the input/output fort file, i.e., fort.26  
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**

hwrf\_inter\_2to2.exe

**FUNCTION:** Interpolates from 3X domain to ghost\_d02 or ghost\_d03

## 5. Vortex Relocation

**INPUT:**     \$gesfhr (=6) - last digit of the input/output fort file, i.e., fort.26  
          tcvitals.as (fort.11) - observed storm center  
          new\_data\_4x (fort.26) - adjusted vortex + environment  
          new\_ghst\_d0[2-3] (fort.36) - input ghost file in binary format from  
          either d02 or d03  
          new\_gfs\_d01 (fort.46) - outer domain adjusted GFS data

**OUTPUT:**    data\_merge\_g0[2-3] (fort.56) - merged data on respective do-  
          main

**USAGE:**     **echo \$gesfhr \$BASIN | hwrp\_inter\_2to2.exe**

diffwrf\_3dvar.exe

Refer to Stage 1 in section 5.2.2.

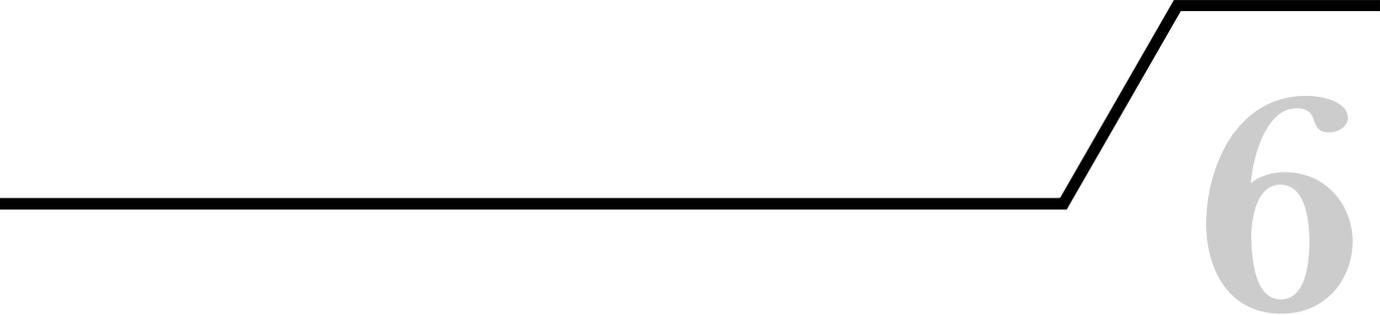
hwrp\_anl\_bogus\_10m.exe

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

**INPUT:**     \$gesfhr (=6) - last digit of the input/output fort file, i.e., fort.26  
          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 calcula-  
          tion  
          storm\_pert\_gfs (fort.61) - separated GFS 3D vortex field  
          storm\_radius\_gfs (fort.85)  
          hwrp\_storm\_cyn\_axisy\_47 (fort.71, 72, 73, 74, 75, 78)     input  
          static vortex data  
          hwrp\_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 | hwrp\_anl\_bogus\_10m.exe**



# 6

## Data Assimilation

### 6.1 Introduction

The preliminary initial conditions created by downscaling the global model data and performing the vortex relocation procedures are further modified with data assimilation using GSI on the 6- and 2-km WRF Ghost domains. No data assimilation is done in the 18-km parent domain. The term HDAS, or HWRF Data Assimilation System, refers to the process of running GSI for data assimilation in HWRF.

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, pregenerated matrix from the global model, and a flow-dependent matrix derived from an ensemble of 6-h forecasts. Because HWRF uses an ensemble, but does not feedback into it, this procedure is termed "one-way hybrid". The 2015 HWRF operational implementation uses a 40 member HWRF ensemble for the North Atlantic and Eastern Pacific domains when inner-core data from NOAA's P3 TDR are available. However, this procedure is not supported in the HWRF v3.7a public release. Users can still use the GFS ensemble for hybrid ensemble-variational data assimilation. For more information on the ensemble-variational method, refer to the HWRF v3.7a Scientific Documentation available from the DTC website ([www.dtcenter.org/HurrWRF/users](http://www.dtcenter.org/HurrWRF/users)).

The datasets assimilated in operations in the 6-km (d02) and 2-km (d03) domains are described in the HWRF Scientific Documentation. HWRF has the capability of assimilating tropical cyclone inner-core data such as the NOAA's P3 TDR observation. 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

## 6. Data Assimilation

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 span a range of times, the First Guess at Appropriate Time (FGAT) procedure 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 4.2); at the HWRF initial time (Figure 4.3); and 3 h after the HWRF initial time (Figure 4.4). To create the three first-guesses, the real\_nmm, short WRF forecasts, and vortex adjustment procedures are performed three times. This produces all of the ghost d03 and ghost d02 output files that are used by GSI in its FGAT operation (Figure 6.1).

After the data is assimilated in the ghost d02 and ghost d03 domains, the preliminary analysis for the parent domain, the middle and inner domain output from the WRF Analysis, and the ghost d02 and ghost d03 GSI analysis (which used FGAT) are merged to produce the final atmospheric IC for the 5-day forecast. To perform the data assimilation in the ghost domain, users should run GSI and then merge. For more details about GSI, please consult the GSI Users' Guide available from the DTC at [http://www.dtcenter.org/com-GSI/users/docs/users\\_guide/GSIUserGuide\\_v3.4.pdf](http://www.dtcenter.org/com-GSI/users/docs/users_guide/GSIUserGuide_v3.4.pdf).

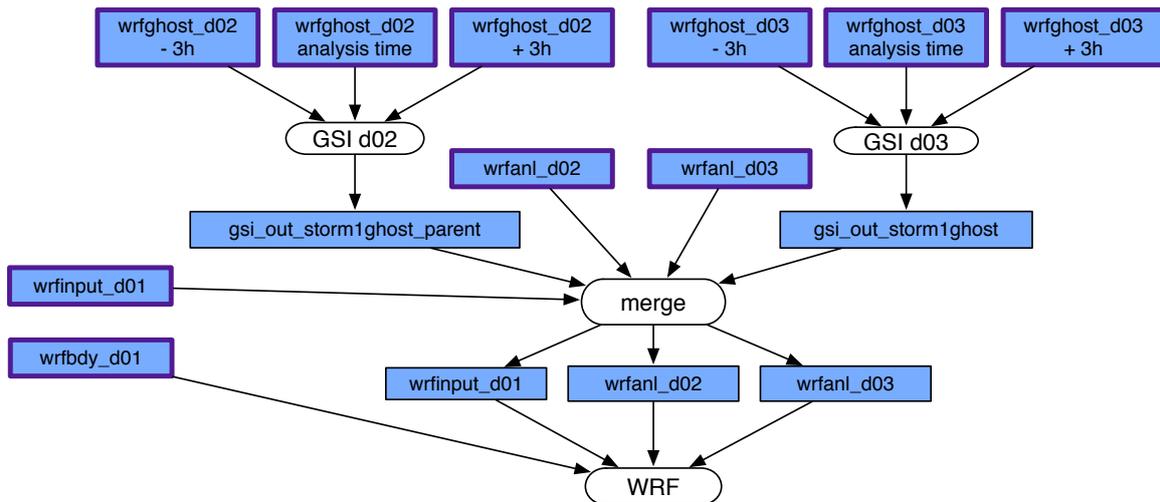


Figure 6.1: Simplified GSI and merge procedures. Purple outlined boxes correspond to the purple outlined boxes of the Figures in section 4.2. Blue boxes are NetCDF files.

## 6.2 Scripts

The HWRF data assimilation component begins by running a data preparation script, `bufr-prep-wrapper`, which calls `scripts/exhwrp_bufrprep.py`. Next two wrapper scripts, `gsi_d02-wrapper` and `gsi_d03-wrapper` are run. These wrappers are responsible for calling their respective instances of `scripts/exhwrp_gsi.py`.

### 6.2.1 Overview of `exhwrf_bufrprep.py`

1. Initialize the objects used to run all components of HWRF by calling `hwrf_expt.init_module()`.
2. If GSI is enabled, run the bufr data preparation script.

### 6.2.2 Overview of the Bufrprep Module

The run module for the bufr data preparation step (an FGATGSI class object) is located in `ush/hwrf/bufrprep.py`, and is responsible for the following tasks:

1. Copy the prepbuf files
2. Link the specified observations files
3. Run a prepbuf processing algorithm based on the choice of `bufrprep.prepbufprep` flag in `hwrf.conf`.
  - Case 0: Make no change. Do not run a program.
  - Case 1: Remove some inner-core data by running `hwrf_rem_prepbuf_typ_in_circle`.
  - Case 2: Change the flag for mass and dropsonde u, v data by running `hwrf_change_prepbuf_qm_in_circle`.
  - Case 3: Unflag HS3 dropsonde data by running `hwrf_change_prepbuf_qm_typ`. (Default value)

#### Output files:

<code>gsi_status.{STORMNAME}{SID}.YYYYMMDDHH</code>	GSI status file
<code>gfs.tHHz.prepbuf.nr</code>	prepbuf data

#### Status Check:

In the standard output file, you will find: `INFO: - exit status 0.`

#### Executables:

`hwrf_rem_prepbuf_typ_in_circle.exe`

FUNCTION: Removes some inner core data

INPUT: `prepbuf.ALL (fort.21)` - prepbuf file containing original observations  
\$RLATC - environment parameter for storm latitude  
\$RLONC - environment parameter for storm longitude  
\$RRADC - environment parameter for the radius of a circle centered at TC center

OUTPUT: `prepbuf (fort.51)` - edited prepbuf file

## 6. Data Assimilation

USAGE: **hwrf\_rem\_prepbufr\_typ\_in\_circle.exe**

hwrf\_rem\_prepbufr\_qm\_in\_circle.exe

FUNCTION: Flags or unflags observations of mass and dropsonde u, v data

INPUT: prepbufr.ALL (fort.21) - prepbufr file containing original observations  
\$RRADC - environment parameter for half the side length of a square centered at TC center  
\$RRADC - environment parameter for the radius of a circle centered at TC center

OUTPUT: prepbufr (fort.51 - edited prepbufr file

USAGE: **hwrf\_rem\_prepbufr\_qm\_in\_circle.exe**

hwrf\_change\_prepbufr\_qm\_typ.exe

FUNCTION: Unflags HS3 dropsonde data

INPUT: prepbufr.ALL (fort.21) - prepbufr file containing original observations

OUTPUT: prepbufr (fort.51 - edited prepbufr file

USAGE: **hwrf\_change\_prepbufr\_qm\_typ.exe**

### 6.2.3 Overview of `exhwrf_gsi.py`

1. Initialize the objects used to run all components of HWRF by calling `hwrf_expt.init_module()`.
2. Run GSI for the appropriate domain.

### 6.2.4 Overview of the GSI Module

The run module for GSI (an `FGATGSI` class object) is located in `ush/hwrf/gsi.py`, and is responsible for the following tasks:

1. Link the fixed files
2. Link the GFS ensemble files
3. Link the observation files
4. Link the bias correction files

## 6. Data Assimilation

5. Create a namelist for the GSI analysis
6. Copy the background file (`wrf_inout`) from the corresponding WRF Ghost run
7. Run `gsi.exe`

### Output files:

`stdout`            The 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; format is same as the input background file

### Status Check:

In the standard output file, you will find the line "INFO: GSI succeeded" followed by the domain for which the assimilation was run (`stormlghost_parent` is the intermediate nest and `stormlghost` is the innermost nest).

### Executables:

`gsi.exe`

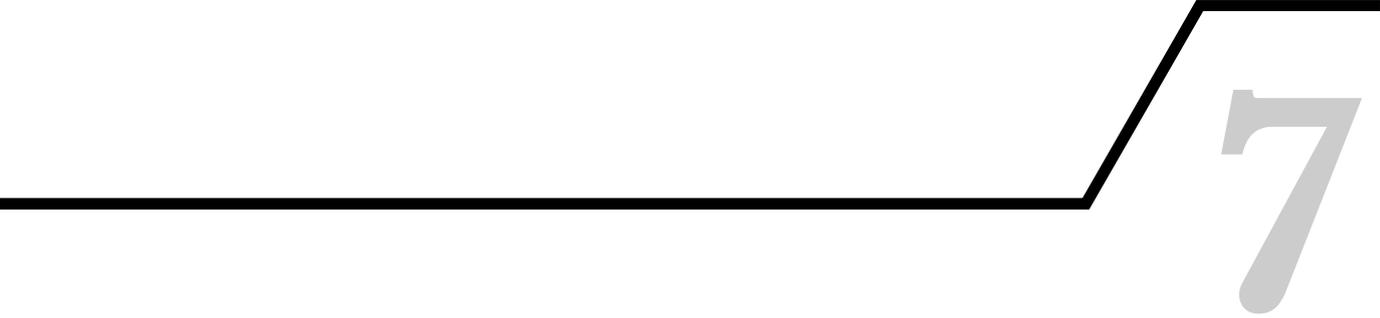
FUNCTION: Performs the GSI 3D-VAR data assimilation analysis

INPUT:     `gsiparm.anl` - GSI namelist, created by the script by modifying template `/parm/hwrf_gsi.nml`  
         `filelist` - ASCII file with 80 lines, each one containing a file name for a GFS ensemble member (used for ensemble-based background covariance)  
         `satbias_angle` - file containing information on satellite angle, from dataset directory  
         `satbias_in` - file containing information on satellite bias, from dataset directory  
         `wrf_inout` - background file, copied from WRF Ghost output  
         Various observations in BUFR and prepBUFR format

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

`satbias_out` - Newly computed satellite bias

USAGE:     `gsi.exe < gsiparm.anl`



# 7

## Merge

### 7.1 Introduction

Once the HWRf atmospheric initialization has been completed with the use of the vortex relocation and data assimilation, the adjusted ICs on all grids must be merged to provide the final ICs for the HWRf 5-day forecast. The origin of the files going into the merge procedure is shown in Figure 6.1. The merge is run by the wrapper script `merge_wrapper`. A description of the domains used in HWRf is included in section 4.1.

### 7.2 Scripts

Merge is run by submitting the `merge_wrapper`, which sets necessary environment variables before running Python script `exhwrf_merge.py`.

#### 7.2.1 Overview of `exhwrf_merge.py`

1. Initialize the objects used to run all components of HWRf by calling `hwrf_expt.init_module()`.
2. Run the `gdas_merge` Python module.

## 7.2.2 Overview of Merge Module

Merge is a RelocationTask object whose run module lives in `ush/hwrf/relocate.py` and is responsible for the following tasks:

1. Copy the input files.
2. Check to see whether `storm_radius` file exists from relocate process and contains information.
3. Run `diffwrf_3dvar.exe` to convert the netCDF format `wrfinput_d0[1-3]` and `wrfghost_d0[2-3]` to unformatted data files `new_hdas_d01`, `new_gfs_d0[2-3]`, and `new_ghd_d0[2-3]`.
4. Run `hwrf_inter_2to1.exe` to interpolate the data in file `new_ghd_d03` and `new_gfs_d03` to the inner-nest domain grid. This will produce the merged data on the inner-nest grid (`data_merge_d03`).
5. Run `hwrf_inter_2to1.exe` to interpolate the data in file `new_ghd_d02` and `new_gfs_d02` to the inner-nest domain grid. This will produce the merged data on the inner-nest grid (`data_merge_d02`).
6. Run `hwrf_inter_2to6.exe` to interpolate the files `new_hdas_d01`, `new_gfs_d02`, and `new_ghd_d02` to the outer domain grid. This will produce the merged data on the outer domain grid (`data_merge_d01`).
7. Run `diffwrf_3dvar.exe` to convert the unformatted files `data_merge_d0[1-3]` to the netCDF format files `wrfinput_d0[1-3]`.
8. Deliver the products.

### Output files:

<code>wrfinput_d01</code>	IC for the outer domain containing the new vortex
<code>wrfinput_d02</code>	IC for the middle-nest domain containing the new vortex
<code>wrfinput_d03</code>	IC for the inner-nest domain containing the new vortex

### Status Check:

Check that output files exist in the `com/` directory for the current cycle. The line "INFO: `exhwrf_merge` has completed" will also appear near the end of the standard output file.

### Executables:

`diffwrf_3dvar.exe`

Refer to Stage 1 in section 5.2.2.

## 7. Merge

hwrf\_inter\_2to1.exe

**FUNCTION:** Interpolates from ghost domains to nest domains (*DOMAIN* is "02" or "03")

**INPUT:**     \$gesfhr (=6) - last digit of the input/output fort file, i.e. fort.26  
          new\_ght\_d{*DOMAIN*} (fort.26) - data on ghost domain  
          new\_gfs\_d{*DOMAIN*} (fort.36) - data on nest domain

**OUTPUT:**    data\_merge\_d{*DOMAIN*} (fort.56) - interpolated data on inner domain

**USAGE:**     **echo \$gesfhr \$BASIN | hwrf\_inter\_2to1.exe**

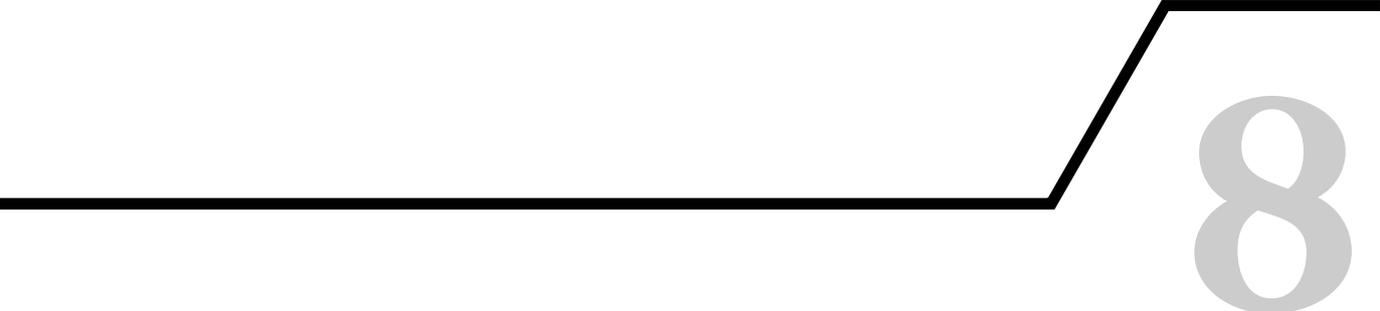
hwrf\_inter\_2to6.exe

**FUNCTION:** Interpolates data from ghost domain to outer domain.

**INPUT:**     \$gesfhr (=6) - last digit of the input/output fort file, i.e. fort.26  
          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\_splitl.exe  
          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**



# 8

## Ocean Initialization for MPIPOM-TC

### 8.1 Introduction

This chapter explains how to run the initialization of the MPIPOM-TC component of the HWRF model, available from the DTC. Users are also encouraged to read the HWRF v3.7a Scientific Documentation.

### 8.2 Scripts

The initialization of the HWRF ocean model, MPIPOM-TC, is accomplished by running the `init_ocean_wrapper`, which is responsible for linking the ocean executables to the `exec/` directory and running `exhwrf_ocean_init.py` to generate updated initial conditions for the ocean forecast component of HWRF.

#### 8.2.1 Overview of `exhwrf_ocean_init.py`

1. Initialize the objects used to run all components of HWRF by calling `hwrf_expt.init_module()`.

## 8. Ocean Initialization for MPIPOM-TC

2. Run `pominit`. If successful, write a status file to indicate that the forecast will be coupled, otherwise indicate that the forecast will be uncoupled.

### 8.2.2 Overview of Ocean Init Modules

1. Determine the region for which the ocean model will be run. Currently supported options are Transatlantic, East Pacific, West Pacific, North Indian, South Indian, Southwest Pacific, and Southeast Pacific domains, as described in the HWRF Scientific Documentation.
2. Determine the ocean initial condition module to be used. Currently supported options, which are consistent with the 2015 operational configuration of HWRF, include the Generalized Digital Environmental Model (GDEM) temperature and salinity climatology with feature-based modifications for the transatlantic domain and the unmodified GDEMv3 climatology for the East Pacific domain. If coupling with the ocean is configured for other basins, a non-operational procedure, GDEMv3 is also used.
3. Link the input and fix files.
4. Run `gfdl_getsst.exe` to obtain the sea surface temperature and land/sea mask from the GFS analysis.
5. Run `gfdl_sharp_mcs_rf_l2m_rmy5.exe` (in transatlantic domain only) to assimilate ocean features, including major fronts and eddies, and sharpen the frontal gradients.
6. Run `transatl06prep.xc` (in transatlantic domain only) to blend the sharpened GDEM and the unsharpened GDEM along 50 W longitude.
7. Prepare ocean initial conditions for MPIPOM-TC Phase 1. By default, `pomprep_fbtr.xc` is used for the transatlantic domain, and `pomprep_gdm3.xc` is used for the East Pacific domain. Also, by default, both executables are set to assimilate the GFS SST analysis into the upper ocean mixed layer, creating an ocean initial condition at the sea surface that is identical to the atmospheric initial condition at the sea surface.
8. Run `hwrfocean_init.exe` for Phase 1 to spin up the ocean currents. The SST is held constant during Phase 1. Historically, Phase 1 has also been known as Phase 3, so the terms *Phase 1* and *Phase 3* are sometimes used interchangeably.
9. Run `hwrfocean_init.exe` for Phase 2 to generate the cold wake at the sea surface prior to the start of the coupled model forecast. Historically, Phase 2 has also been known as Phase 4, so the terms *Phase 2* and *Phase 4* are sometimes used interchangeably.

#### Executables:

`gfdl_getsst.exe`

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

INPUT:     `for11 (gfs.YYYYMMDDHH.tHHz.sfcanl)`  
          `fort.11 (gfs.YYYYMMDDHH.tHHz.sfcanl)`  
          `fort.12 (gfs.YYYYMMDDHH.tHHz.sanl)`

## 8. Ocean Initialization for MPIPOM-TC

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

**USAGE:** **gfdl\_getsst.exe > getsst.out**

gfdl\_sharp\_mcs\_rf\_l2m\_rmy5.exe

**FUNCTION:** Run the feature-based sharpening program, which takes the GDEM T/S climatology, horizontally interpolates it onto the old POM-TC grid for the United domain, and employs the diagnostic, feature-based modeling procedure, as described in the HWRF Scientific Documentation. This executable is called for the transatlantic domain only.

**INPUT:** input\_sharp  
fort.66 (gfdl\_ocean\_topo\_and\_mask.REGION)  
fort.8 (gfdl\_gdem.MM.ascii)  
fort.90 (gfdl\_gdem.MM±1.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 (mmdl.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.united.MM)  
sharp\_mcs\_r\_l2b.out

**USAGE:** **gfdl\_sharp\_mcs\_rf\_l2m\_rmy5.exe < input\_sharp >**  
**sharp\_mcs\_r\_l2b.out**

transatl06prep.xc

**FUNCTION:** Blend T/S between sharpened GDEM and unsharpened GDEM along 50W. This executable is called for the transatlantic domain only.

**INPUT:** fort.8 (gfdl\_gdem.MM.ascii)  
fort.90 (gfdl\_gdem.MM±1.ascii)  
fort.91 (mmdl.dat)  
fort.13 (gfdl\_initdata.united.MM)

## 8. Ocean Initialization for MPIPOM-TC

**OUTPUT:** fort.113 (gfdl\_initdata.REGION.MM)  
transatl06prep.out

**USAGE:** **transatl06prep.xc > transatl06prep.out**

pomprep\_fbtr.xc

**FUNCTION:** Read sharpened and blended GDEM climatology, horizontally interpolate it onto the high-resolution MPIPOM-TC grid, incorporate the bathymetry and a land-sea mask, assimilate the GFS SST, and prepare the ICs for MPIPOM-TC. This executable is called for the transatlantic domain only.

**INPUT:** input  
fort.13 (gfdl\_initdata.transatl.MM)  
fort.66 (gfdl\_ocean\_topo\_and\_mask.REGION.lores)  
fort.21 (sst.gfs.dat)  
fort.22 (mask.gfs.dat)  
fort.23 (lonlat.gfs)

**OUTPUT:** STORM.el\_initial.nc  
STORM.grid.nc  
STORM.ts\_clim.nc  
STORM.ts\_initial.nc  
STORM.uv\_initial.nc  
ocean\_pomprep.out

**USAGE:** **pomprep\_fbtr.xc < input > ocean\_pomprep.out**

pomprep\_gdm3.xc

**FUNCTION:** Read GDEMv3 climatology, horizontally interpolate it onto the high resolution MPIPOM-TC grid, incorporate the bathymetry and a land/sea mask, assimilate the GFS SST, and prepare the ICs for MPIPOM-TC. This executable is currently operational for the East Pacific domain only, but it can be used worldwide.

**INPUT:** input  
tin.nc (tgdemv3sMM.nc)  
sin.nc (sgdemv3sMM.nc)  
fort.66 (gfdl\_ocean\_topo\_and\_mask.REGION.lores)  
fort.21 (sst.gfs.dat)  
fort.22 (mask.gfs.dat)  
fort.23 (lonlat.gfs)

## 8. Ocean Initialization for MPIPOM-TC

**OUTPUT:** *STORM.el\_initial.nc*  
*STORM.grid.nc*  
*STORM.ts\_clim.nc*  
*STORM.ts\_initial.nc*  
*STORM.uv\_initial.nc*  
ocean\_pomprep.out

**USAGE:** **pomprep\_gdm3.xc < input > ocean\_pomprep.out**

hwrf\_ocean\_init.exe

**FUNCTION:** Run MPIPOM-TC ocean Phase 1 or Phase 2 (also known historically as ocean Phase 3 and Phase 4, respectively, as in the model code).

**INPUT:** **For Phase 1**  
pom.nml  
*STORM.el\_initial.nc*  
*STORM.grid.nc*  
*STORM.ts\_clim.nc*  
*STORM.ts\_initial.nc*  
*STORM.uv\_initial.nc*  
**For Phase 2**  
pom.nml  
*STORM.el\_initial.nc*  
*STORM.grid.nc*  
*STORM.ts\_clim.nc*  
*STORM.ts\_initial.nc*  
*STORM.uv\_initial.nc*  
restart.phasel.nc

**OUTPUT:** **For Phase 1**  
restart.phasel.nc  
*STORM.0000.nc*  
*STORM.0001.nc*  
*STORM.0002.nc*  
**For Phase 2**  
restart.phase2.nc  
*STORM.0000.nc*  
*STORM.0001.nc*  
*STORM.0002.nc*  
*STORM.0003.nc*

**USAGE:** **hwrf\_ocean\_init.exe > ocean\_init.out**

### 8.3 MPIPOM-TC Diagnostics

MATLAB-based diagnostic tools are now available to plot the SST, subsurface temperature, and ocean current vectors at various times throughout the forecast and at various zoom levels. A README file in `pomtc/ocean_diag/matlab` explains how to use the mature tools. Additional community-based plotting tools are also provided for the creation of plots such as vertical cross sections.

### 8.4 User-provided Datasets for MPIPOM-TC Initialization

To initialize MPIPOM-TC with user-provided datasets, it is necessary to turn off SST assimilation by choosing `SSTASIM = False` in `master.py`.

The user-provided datasets should be in the same format as the Navy Coupled Ocean Data Assimilation (NCODA) datasets. Once the data is in this format, MPIPOM can be run with this data by selecting the NCODA option in `master.py` script.

Example: `oinit_d = {"transatl": "NCODA"}`

A FORTRAN program to generate data in this format is provided below:

```

INTEGER, PARAMETER                :: nx=2161,ny=1051,nl=34
INTEGER, PARAMETER                :: reclen=nx*ny*4
REAL, DIMENSION(nl)              :: zncda
REAL, DIMENSION(nx,ny)           :: alonin, alatin
REAL, DIMENSION(nx,ny,nl)       :: temp, salt

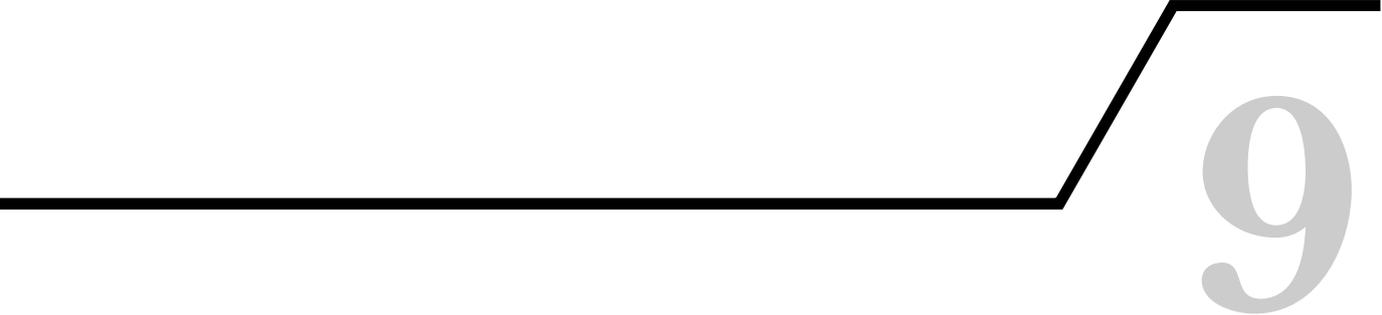
! Vertical depth levels where data need to be
zncda =      (0., 2.5, 7.5, 12.5, 17.5, 25., 32.5,
              40., 50., 62.5, 75., 100., 125., 150.,
              200., 300., 400., 500., 600., 700., 800.,
              900.,1000.,1100.,1200.,1300.,1400.,1500.,
              1750.,2000.,2500.,3000.,4000.,5000.)

fname = 'grdlon_sfc_000000_000000_1o2161x1051_datafld'
OPEN(78,FILE=fname,ACCESS='DIRECT',RECL=reclen,FORM='UNFORMATTED')
WRITE(78,REC=1) alonin ! Longitude values
CLOSE(78)
fname='grdlat_sfc_000000_000000_1o2161x1051_datafld'
OPEN(79, FILE=fname,ACCESS='DIRECT',RECL=reclen,FORM='UNFORMATTED')
WRITE(79,REC=1) alatin ! Latitude values
CLOSE(79)

```

### *8. Ocean Initialization for MPIPOM-TC*

```
fname='seatmp_pre_000000_005000_1o2161x1051_yyyymmdd00_00000000_analfld'  
OPEN(48, FILE=fname, ACCESS='DIRECT', RECL=reclen, FORM='UNFORMATTED')  
fname='salint_pre_000000_005000_1o2161x1051_yyyymmdd00_00000000_analfld'  
OPEN(49, FILE=fname, ACCESS='DIRECT', RECL=reclen, FORM='UNFORMATTED')  
DO k = 1, nl  
  READ(48, REC=k) temp(:, :, k) ! Temperature  
  READ(49, REC=k) salt(:, :, k) ! Salinity  
END DO  
CLOSE(48)  
CLOSE(49)
```



# 9

## Forecast Model

### 9.1 Introduction

The operational HWRF, which runs in all basins, is an atmosphere-ocean-coupled forecast system, which includes an atmospheric component (WRF-NMM), an ocean component (MPIPOM-TC), and the NCEP Coupler. Therefore, HWRF is a Multiple-Program Multiple-Data (MPMD) system which consists of three executables, WRF, MPIPOM-TC, and Coupler. After the ocean and atmosphere initializations are successfully completed, the coupled HWRF forecast can be submitted. In the Central, West and South Pacific, and Indian Ocean basins, HWRF is run operationally in atmosphere standalone mode, that is, uncoupled.

### 9.2 Scripts

The wrapper script `forecast_wrapper` is responsible for calling the Python script `ex-hwrf_forecast.py` in the `scripts/` directory. The wrapper script sets the number of tasks for the parallel forecast job. In operations, 477 tasks are used: 464 for the WRF forecast, nine for the MPIPOM-TC, and four for the NCEP Coupler. While this configuration is recommended, you may change the total number of tasks to reflect the following relationship,

$$TOTAL\_TASKS = np + 9 + 4, \tag{9.1}$$

where  $np$  is an integer multiple of 4. The total number of processors used should match `TOTAL_TASKS`.

## 9. Forecast Model

For uncoupled runs, you should change the variable `TOTAL_TASKS` in `forecast_wrapper` to reflect the reduction of tasks (i.e., subtract 13 for the MIPOM-TC and Coupler). The number of processors chosen should also be consistent with the product of the WRF namelist variables `runwrf.nproc_x` and `runwrf.nproc_y` that are originally set in `system.conf` and can be changed by passing arguments to `exhwrflaunch.py` inside the `launcher_wrapper`.

### 9.2.1 Overview of `exhwrflaunch.py`

1. Initialize all of the objects used to run HWRF
2. Run the HWRF main forecast, coupled or uncoupled (`runwrf.run`)

### 9.2.2 Overview of the Forecast Module

For coupled forecasts, `runwrf` is an object of the `WRFCoupledPOM` subclass of `fcst-task.WRFAtmos`. The run module is responsible for the following tasks:

1. Link the input files required by WRF (fix files, initial and boundary condition files, and geographical data files).
2. Make the Coupler namelist.
3. Make the POM namelist.
4. Copy POM inputs.
5. Run `hwrflaunch_dynamic.exe` to calculate the location of the middle nest.
6. Make the WRF namelist.
7. Submit the MPI forecast run (three executables: `wrf.exe`, `hwrflaunch_fcst.exe`, `hwrflaunch_wm3c.exe`).

#### Output files:

Primary output files, containing a large number of forecast variables, are produced every hour for the first nine hours, then 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, with hourly output in a single file for each domain.

```
wrfdiag_d01  
wrfdiag_d02  
wrfdiag_d03
```

## 9. Forecast Model

Text file with time series of storm properties.

hifreq\_d03.htcf

File hifreq\_d03.htcf has nine columns containing the following items:

1. Forecast lead time (s)
2. Minimum Sea Level Pressure (MSLP) in the inner nest (hPa)
3. Latitude of grid point with minimum sea level pressure
4. Longitude of grid point with minimum sea level pressure
5. Maximum wind in the inner nest at the lowest model level (kt)
6. Latitude of grid point with the maximum wind
7. Longitude of grid point 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:

<code>STORMNAME.00DD.nc</code>	Ocean forecast output for each day ( <i>DD</i> ) in NetCDF format
<code>flux.00DD</code>	Forecast ocean flux for each day
<code>sst.00DD</code>	Forecast sea surface temperature for each day
<code>sstuvhflux.00HH</code>	Additional fields used for diagnostics, per hour

### 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`). This check is also done in the code, and can be found in the standard output file.

### Executables:

`hwrf_swcorner_dynamic.exe`

Refer to section 4.2.3.

`wrf.exe`

FUNCTION: Atmospheric component of HWRF

INPUT: `geo_nmm.d01.nc` - Geogrid static files for d01  
`geo_nmm_nest.101.nc` - Geogrid static files for d02  
`geo_nmm_nest.102.nc` - Geogrid static files for d03  
`wrfbdy_d01` - LBCs for d01  
`wrfinput_d01` - ICs for d01  
`wrfanl_d02_YYYY-MM-DD_HH_00_00` - ICs for d02  
`wrfanl_d03_YYYY-MM-DD_HH_00_00` - ICs for d03  
`namelist.input` - Example in Appendix B  
`fort.65`  
WRF Fix files (Refer to section 4.2.3)

## 9. Forecast Model

**OUTPUT:** wrfout\_d01\_YYYY-MM-DD\_HH\_00\_00  
wrfout\_d02\_YYYY-MM-DD\_HH\_00\_00  
wrfout\_d03\_YYYY-MM-DD\_HH\_00\_00  
wrfdiag\_d01  
wrfdiag\_d02  
wrfdiag\_d03  
hifreq\_d03.htcf

**USAGE:** For a coupled HWRF forecast, `wrf.exe` must be submitted with the coupler and the ocean model. Refer to MPI Explanation below.  
For an uncoupled run, you only need to issue the executable.  
**wrf.exe**

`hwrf_ocean_fcst.exe`

**FUNCTION:** MPIPOM-TC ocean model for HWRF

**INPUT:** `STORM.el_initial.nc`  
`STORM.grid.nc`  
`STORM.ts_clim.nc`  
`STORM.ts_initial.nc`  
`STORM.uv_initial.nc`  
`restart.phase2.nc`

**OUTPUT:** `STORM.0000.nc`  
`STORM.0001.nc`  
`STORM.0002.nc`  
`STORM.0003.nc`  
`STORM.0004.nc`  
`STORM.0005.nc`

**USAGE:** For a coupled HWRF forecast, the ocean model `hwrf_ocean_fcst.exe` must be submitted to the computers with the atmosphere model `wrf.exe` and the coupler `hwrf_wm3c.exe`. Refer to the MPI explanation below.

`hwrf_wm3c.exe`

**FUNCTION:** Coupler that links the atmospheric component and oceanic component

**INPUT:** `cp1.nml` - coupler namelist

**OUTPUT:** None

**USAGE:** Refer to the MPI explanation below

**Explanation of the MPI command for the forecast model**

As mentioned in section 9.1, HWRF can be run as either a coupled or uncoupled model of the atmosphere and ocean. The operational HWRF runs are coupled in the AL and EP basins. By default, the scripting system submits coupled runs in the AL and EP basins when `config.run_ocean=yes` in `hwrp_basic.conf`, and uncoupled runs in other basins (`config.run_ocean=no`). If an uncoupled run in the AL or EP basins is desired, refer to section 3.8.1 for configuration details.

- 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 477 processors, four for the coupler (`hwrp_wm3c.exe`), nine for the ocean domain (`hwrp_ocean_fcst.exe`) and 464 for `wrf.exe`.

```
hwrp_wm3c.exe
hwrp_ocean_fcst.exe
wrf.exe
wrf.exe
wrf.exe
wrf.exe
```

With MOAB/Torque, using the command `mpiexec`

```
mpiexec -np 4 ./hwrp_wm3c.exe : -np 9 \  
./hwrp_ocean_fcst.exe : -np 464 ./wrf.exe
```

For example, the previous command will run the coupled model using 477 processors, four for the coupler (`hwrp_wm3c.exe`), nine for the ocean domain (`hwrp_ocean_fcst.exe`), and 464 for `wrf.exe`

- Uncoupled

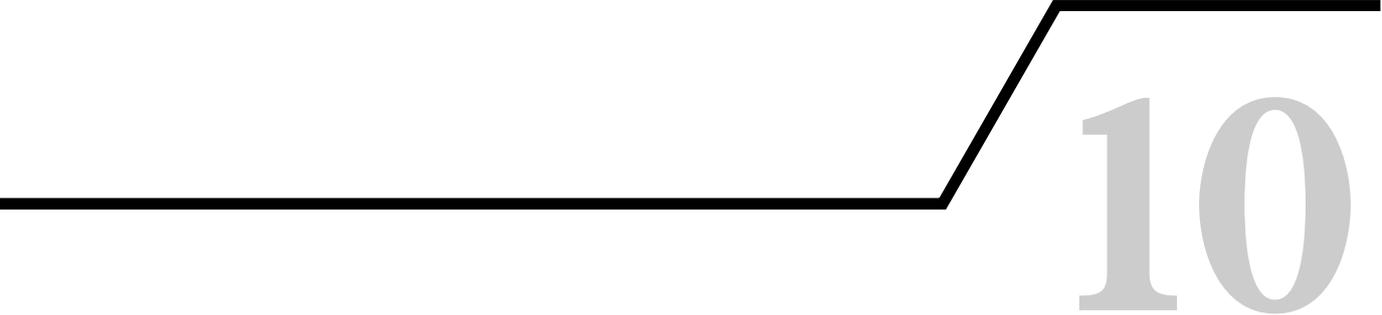
- With LSF, using the command `mpirun.lsf`

```
mpirun.lsf -procs 464 ${WRF_ROOT}/main/wrf.exe
```

- With MOAB/Torque, using the command `mpiexec`

```
mpiexec -np 464 ${WRF_ROOT}/main/wrf.exe
```

Note that the number of processors should be changed when running a reduced-resolution 27/9/3-km HWRF configuration. The coupled system in the 27/9/3-km configuration runs by default with a total of 210 processors: 200 for the atmosphere, one for the coupler, and nine for the ocean.



# 10

## **HWRF Post-Processor**

### 10.1 Introduction

The NCEP UPP is used to destagger 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 section 2. Please refer to the UPP Users' Guide ([http://www.dtcenter.org/upp/users/docs/user\\_guide/V3/upp\\_users\\_guide.pdf](http://www.dtcenter.org/upp/users/docs/user_guide/V3/upp_users_guide.pdf)) for more details.

There are two main executables in UPP, `unipost.exe` and `copygb.exe`. This chapter covers only the module that calls `unipost.exe`. The use of `copygb.exe` is covered in Chapter 11.

### 10.2 Scripts

The postprocessing using UPP is run using two wrappers, `unpost_wrapper` and `post_wrapper`. These wrappers call the `exhwrf_unpost.py` and `exhwrf_post.py` scripts, respectively.

### 10.2.1 Overview of `exhwrf_unpost.py`

The purpose of this script is to delete output from any previous attempt to run the same cycle.

1. Initialize the objects used to run all components of HWRF by calling `hwrf_expt.init_module()`.
2. Run the unrun modules for the following tasks:
  - `runwrf`
  - `wrfcopier`
  - `satpost`
  - `nonsatpost`
  - `gribber`

### 10.2.2 Overview of `exhwrf_post.py`

The Python script contains a loop that continually checks the status of the forecast, and postprocesses any output files that are available. As long as there are tasks remaining, it runs copies of `wrfcopier`, `nonsatpost`, and `satpost`. Note that `satpost` refers to the postprocessing to produce synthetic satellite brightness temperatures, while `nonsatpost` refers to the postprocessing of all other variables (temperature, winds, etc.).

### 10.2.3 Overview of UPP Python Modules

#### Wrfcopier ---

`Wrfcopier` is a `WRFCopyTask` class object that lives in `ush/hwrf/copywrf.py`. It serves the primary purpose of delivering files from the WRF run directory to the `com/` directory.

#### Nonsatpost and Satpost ---

`Nonsatpost` and `satpost` are `PostManyWRF` class objects that run the UPP on the WRF output files. They are used to produce general forecast products and synthetic satellite images, respectively. The run module for these tasks performs the following duties:

1. Link the input file (`wrfout` forecast or analysis file)
2. Make a control file that corresponds to the input file
3. Write `itag` file which contains the following four lines to be read by `unipost.exe`.
  - Name of the WRF output file to be postprocessed
  - Format of the WRF output (NetCDF or binary; choose NetCDF for HWRF)
  - Forecast valid time (not model start time) in WRF format

## 10. HWRF Post-Processor

- Model name (NMM or NCAR; choose NMM for HWRF)
4. Run `unipost.exe`

### Output files:

In the `intercom/` directory, there are directories for each forecast hour containing the `satpost` and `nonsatpost` output. The following list describes the naming convention for these directories and files. The forecast hour, *hh*, by default is hourly for the first nine hours, and three hourly after that for `nonsatpost` output and six-hourly for `satpost` output.

```
satpost-fhh00m/satpost-fhh00m/  
  satpost-fhhh00m-moad.egrb  
  satpost-fhhh00m-stormlinner.egrb  
  satpost-fhhh00m-stormlouter.egrb  
nonsatpost-fhh00m/nonsatpost-fhh00m/  
  nonsatpost-fhhh00m-moad.egrb  
  nonsatpost-fhhh00m-stormlinner.egrb  
  nonsatpost-fhhh00m-stormlouter.egrb
```

### Status check:

The string "INFO: completed post" will appear in the standard output file.

### Executables:

`unipost.exe`

**FUNCTION:** Destaggers the HWRF native output, interpolates it vertically to pressure levels, computes derived variables, and outputs results in GRIB format.

**INPUT:** `hwrf_eta_micro_lookup.dat`  
`wrfout_d01`, `wrfout_d02` or `wrfout_d03` - HWRF native output  
`itag - namelist`  
unipost control file:

for `nonsatpost`: `hwrf_cntrl.nonsat`; and,  
for `satpost`: `hwrf_cntrl.sat${BASIN}`, where *BASIN* can be A, B, L, E, C, P, Q, S, or W for Arabian Sea (Indian Ocean), Bay of Bengal (Indian Ocean), N. Atlantic, E. N. Pacific, Central N. Pacific, S. Indian Ocean, S. Atlantic, S. Pacific, or W. N. Pacific, respectively.

**OUTPUT:** HWRF postprocessed output in GRIB format

**USAGE:** `unipost.exe < itag`



# 11

## Forecast Products

### 11.1 Introduction

HWRF v3.7a will produce several types of forecast products, including processed GRIB1 and GRIB2 files (projected to lat-lon grids), track files, rainfall swath data, and wind products containing information about the tropical cyclone. The processed GRIB2 files are produced on several different grids outlined in Figure 11.1 for the general atmospheric fields and Figure 11.2 for satellite-derived products. Those GRIB1 files are used as input to the GFDL Vortex Tracker. Any of the GRIB output can also be used to create images with visualization packages such as GrADS, NCL, etc. Image generation is not covered in this Users' Guide. Additionally, several files produced in the previous steps of a HWRF run (such as data assimilation and vortex relocation) are copied to the `com` directory to compose a full set of HWRF products.

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. See section 11.2.1 for the HWRF-specific naming conventions of these files.

## 11. Forecast Products

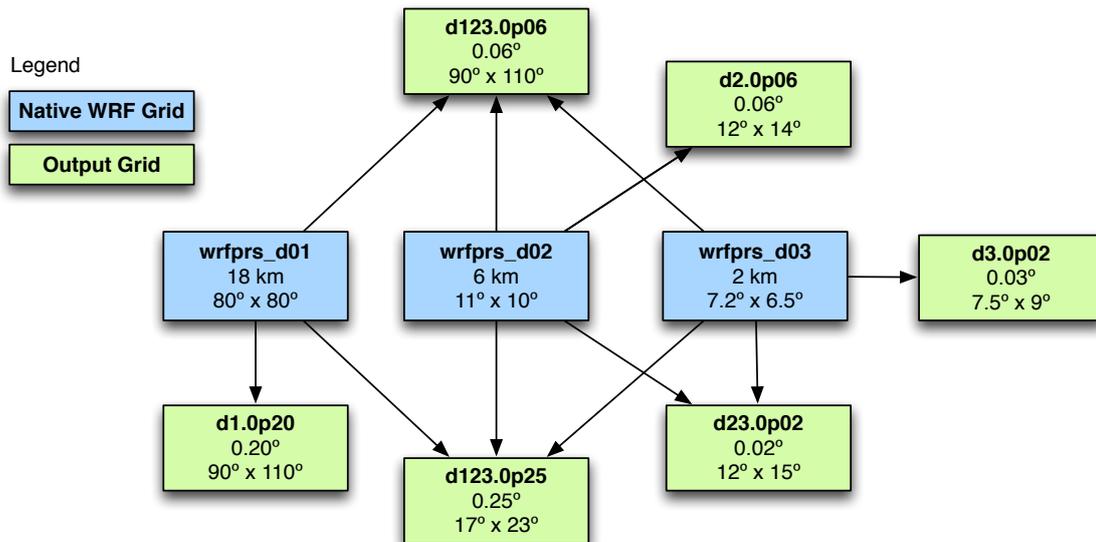


Figure 11.1: Naming convention, resolution, and size for the output grids that contain conventional atmospheric data. Blue boxes indicate the grids from the `wrfout` files. Green boxes are the grids in the final GRIB files.

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 hPa and 700 hPa, MSLP, and geopotential height at 850 and 700 hPa. Secondly, wind speed at 10 m, 850 hPa, and 700 hPa are used. Winds at 500 hPa 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 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 HWRP 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 hPa (for the Vitart scheme) or the geopotential height every 50 hPa from 300 to 900 hPa (for the Cyclone Phase Space scheme).

### 11.2 Scripts

The forecasts products are obtained by running the `products_wrapper`, which calls `scripts/exhwrf_products.py` after setting a few environment variables to redirect the

## 11. Forecast Products

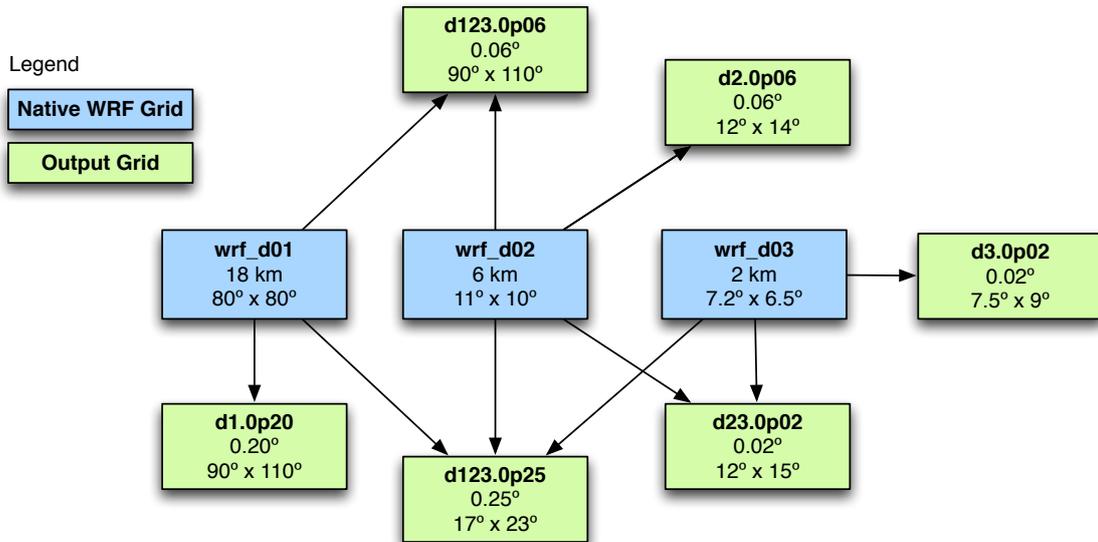


Figure 11.2: Naming convention, resolution, and size for the output grids that contain satellite-derived products. Blue boxes indicate the grids from the `wrfout` files. Green boxes are the grids in the final GRIB files.

standard output and standard error files. These files can be placed anywhere by changing the environment variables `REGTRIBBER_LOGS` and `TRACKER_LOGS` to the desired path.

The GFDL Vortex Tracker is driven by the wrapper script `tracker_wrapper`, which calls `scripts/exhwrf_products.py`. The Python script runs the tracker on the processed GRIB forecast files.

### 11.2.1 Overview of `exhwrf_products.py`

1. Initialize the objects used to run all components of HWRF by calling `hwrf_expt.init_module()`
2. Launch the four parallel tasks. The parent process launches `products`, while the subprocesses run copies of `gribber` and `tracker`. Each of these continually check the availability of files before running.
3. Deliver products to `com/` directory as they become available from the `products`, `gribber`, `tracker` and other processes.

**Files delivered to** `com/YYYYMMDDHH/STORMID` **directory:**

The following examples are for Hurricane Gonzalo (2014), which was AL storm number 08. The string `"gonzalo08"` would be replaced by the name and number of the storm in the given experiment. `SID` is the storm ID (i.e., 08L for Gonzalo). The "l" or "L" following `"gonzalo08"` is a product of an operational naming convention requiring some files to have identical counterparts, but different capitalization. In this case, the letter "L" denotes

## 11. Forecast Products

that the files are being delivered for a storm in the AL. Upper case *YYYYMMDDHH* denotes analysis time, while lower case *yyyymmddhh* denotes forecast time.

081.YYYYMMDDHH.domain.center	ASCII file containing the coordinates of the domain center
081.trak.hwrft.atcfunix.YYYYMMDDHH.combine	Forecast track
081.wrfout_d[01-03]_yyyy-mm-dd_hh:00:00	WRF fcst output for the first 9 hrs in 3-hr increments
aal082014_HWRF_hPYHW_YYYYMMDDHH.dat	Tracker output
gonzalo081.YYYYMMDDHH.fort.65	Land-sea mask for the coupler
gonzalo08L.YYYYMMDDHH.gsi_d02.diag _[instrument]_[satellite]_anl.gz	GSI diagnostic outputs from D02 domain
gonzalo08L.YYYYMMDDHH.gsi_d02.diag _[instrument]_[satellite]_ges.gz	GSI diagnostic outputs from D02 domain
gonzalo08L.YYYYMMDDHH.gsi_d02.stdout.anl	GSI std out from D02 domain
gonzalo08L.YYYYMMDDHH.gsi_d02.diag_conv _anl.gz	GSI diagnostics from D02 domain
gonzalo08L.YYYYMMDDHH.gsi_d02.diag_conv _ges.gz	GSI diagnostics from D02 domain
gonzalo08L.YYYYMMDDHH.gsi_d02.stdout.anl	GSI std out from D02 domain
gonzalo08L.YYYYMMDDHH.gsi_d03.diag_conv _anl.gz	GSI diagnostics from D03 domain
gonzalo08L.YYYYMMDDHH.gsi_d03.diag_conv _ges.gz	GSI diagnostics from D03 domain
gonzalo08L.YYYYMMDDHH.gsi_d03.stdout.anl	GSI std out from D03 domain
gonzalo081.YYYYMMDDHH.hwrft_d03.htcf	Storm info from d03 every 5 mins
GONZALO081.YYYYMMDDHH.hwrft_d03.htcf	Storm info from d03 every 5 mins
gonzalo081.YYYYMMDDHH.hwrftprs.d1.0p20. f[000-126].grb2	Non-satellite vars from D01 domain onto 0.20° d01-sized grid
gonzalo081.YYYYMMDDHH.hwrftprs.d123.0p06. f[000-126].grb2	Non-satellite vars from all domains merged onto 0.06° d01-sized grid
gonzalo081.YYYYMMDDHH.hwrftprs.d123.0p25. f[000-126].grb2	Non-satellite vars from all domains merged onto 0.25° d01-sized grid but in GFS global domain (NCEP GRID 193, 1440x721 lon/lat)
gonzalo081.YYYYMMDDHH.hwrftprs.d2.0p06. f[000-126].grb2	Non-satellite vars from D02 domain onto 0.06° d02-sized grid
gonzalo081.YYYYMMDDHH.hwrftprs.d23.0p02. f[000-126].grb2	Non-satellite vars from all domains merged onto 0.1° d01-sized grid
gonzalo081.YYYYMMDDHH.hwrftprs.d3.0p02. f[000-126].grb2	Non-satellite vars from D03 domain onto 0.02° d03-sized grid
gonzalo081.YYYYMMDDHH.hwrftsat.d1.0p20. f[000-126].grb2	Satellite vars from D01 domain onto 0.2° d01-sized grid
gonzalo081.YYYYMMDDHH.hwrftsat.d12.0p06. f[000-126].grb2	Satellite vars from D01 and D02 domains merged onto 0.06° d01-sized grid
gonzalo081.YYYYMMDDHH.hwrftsat.d123.0p25. f[000-126].grb2	Satellite vars from all domains merged onto 0.25° d01-sized grid but in GFS global domain (NCEP GRID 193, 1440x721 lon/lat)
gonzalo081.YYYYMMDDHH.hwrftsat.d2.0p06. f[000-126].grb2	Satellite vars from D02 domain onto 0.06° d02-sized grid
gonzalo081.YYYYMMDDHH.hwrftsat.d23.0p02. f[000-126].grb2	Satellite vars from D02 and D03 domains merged onto 0.02° d02-sized grid
gonzalo081.YYYYMMDDHH.hwrftsat.d3.0p02. f[000-126].grb2	Satellite vars from D03 domain onto 0.02° d03-sized grid
gonzalo081.YYYYMMDDHH.hwrfttrk. grb[000-126]	Tracker vars merged from all domains on a 0.03° grid, slightly larger than d02
gonzalo081.YYYYMMDDHH.hwrfttrk. grbf[000-126].grbindex	Index file for hwrfttrk
gonzalo081.YYYYMMDDHH.namelist.input	WRF namelist

## 11. Forecast Products

<code>gonzalo081.YYYYMMDDHH.rainfall.ascii</code>	Rainfall
<code>gonzalo081.YYYYMMDDHH.resolution</code>	Text file containing info about nest motion
<code>GONZALO08L.YYYYMMDDHH.resolution</code>	Same as above
<code>gonzalo081.YYYYMMDDHH.stats.short</code>	Storm info at each forecast hour
<code>gonzalo081.YYYYMMDDHH.stats.tpc</code>	Storm info at each forecast hour for delivery to NHC
<code>GONZALO08L.YYYYMMDDHH.stats.tpc</code>	Same as above
<code>gonzalo081.YYYYMMDDHH.storm_vit</code>	Storm info at each forecast hour for delivery to NHC
<code>gonzalo081.YYYYMMDDHH.swathctl</code>	GrADS control file for swath
<code>gonzalo081.YYYYMMDDHH.swath.dat</code>	Along-track wind and rain information
<code>gonzalo081.YYYYMMDDHH.track_d03.patcf</code>	Tracker info from d03
<code>gonzalo081.YYYYMMDDHH.trak.hwrf.3hourly</code>	Tracker output in ATCF format
<code>gonzalo081.YYYYMMDDHH.trak.hwrf.atcfunix</code>	Same as above
<code>gonzalo081.YYYYMMDDHH.trak.hwrfd01.atcfunix</code>	Tracker info from d01
<code>gonzalo081.YYYYMMDDHH.trak.hwrfd01.raw</code>	Same as above
<code>gonzalo081.YYYYMMDDHH.trak.hwrfd02.atcfunix</code>	Tracker info from d02
<code>gonzalo081.YYYYMMDDHH.trak.hwrfd02.raw</code>	Same as above
<code>gonzalo081.YYYYMMDDHH.trak.hwrf.raw</code>	Same as above
<code>gonzalo081.YYYYMMDDHH.trak.hwrf.short6hr</code>	Same as above
<code>gonzalo081.YYYYMMDDHH.wind10hrly.ascii</code>	Hourly maximum 10-m wind
<code>gonzalo081.YYYYMMDDHH.wind10m.ascii</code>	10-m wind ascii
<code>gonzalo081.YYYYMMDDHH.wrfanl_d02</code>	Input analysis for d02
<code>gonzalo081.YYYYMMDDHH.wrfanl_d03</code>	Input analysis for d03
<code>gonzalo081.YYYYMMDDHH.wrfbdy_d01</code>	Boundary conditions for all forecast times
<code>gonzalo081.YYYYMMDDHH.wrfinput_d01</code>	Input analysis for d01
<code>gsi_status.GONZALO081.YYYYMMDDHH</code>	Defines whether to run gsi on d02/d03
<code>ocean_status.GONZALO081.YYYYMMDDHH</code>	Defines whether coupled or not
<code>storm1.conf</code>	Configuration settings for a run
<code>storm1.done</code>	Cycle completion info
<code>storm1.gsi_status</code>	Defines whether to run gsi on d02/d03
<code>storm1.holdvars.txt</code>	Environment variables
<code>storm1.ocean_status</code>	Defines whether coupled or not

## Products \_\_\_\_\_

Products is a module that calls an NHCProducts object, whose run module lives in `ush/h-wrf/nhc_products.py`, and is responsible for the following tasks:

1. Make namelist `products.nml`
2. Link the input files
3. Run `nhc_products.exe`
4. Deliver output

### Executables:

`nhc_products.exe` This executable performs several functions.

## 11. Forecast Products

### INPUT

1. wrfdiag\_d0\* files
2. Products namelist
3. Tcvitals
4. WRF hifreq output
5. ATCF file

### OUTPUT

1. Swath file
2. Modified track file for TPC
3. Ascii wind and rainfall
4. Modified hifreq output from d03
5. File for NHC showing track position, heading and storm speed

### Output files:

An example for Hurricane Gonzalo (2014):

```
gonzalo081.YYYYMMDDHH.wind10hrly.ascii  
gonzalo081.YYYYMMDDHH.rainfall.ascii  
gonzalo081.YYYYMMDDHH.wind10m.ascii  
gonzalo081.YYYYMMDDHH.afos  
GONZALO08L.YYYYMMDDHH.afos  
GONZALO08L.YYYYMMDDHH.stats.tpc  
GONZALO08L.YYYYMMDDHH.hwrf_d03.htcf
```

### Status Check:

The string "WARNING: No subtasks incomplete. I think I am done running. Will exit regribber now." will appear in each of the products standard out files.

## Gribber ---

The Gribber is a GRIBTask object whose run module resides in `ush/hwrf/gribtask.py`. Its primary function is to run `copygb.exe` to horizontally interpolate the native UPP output files to a variety of regular lat/lon grids.

### Output files:

The following sets of files get delivered to the `intercom/regribber`:

```
hwrftrk.YYYYMMDD.HH0000  
hwrftrk.YYYYMMDD.HH0000.grbindex  
hwrftrkd01.YYYYMMDD.HH0000  
hwrftrkd01.YYYYMMDD.HH0000.grbindex
```

## 11. Forecast Products

```
hwrfrkd02.YYYYMMDD.HH0000  
hwrfrkd02.YYYYMMDD.HH0000.grbindex  
parenthigh.YYYYMMDD.HH0000  
trkbasic.YYYYMMDD.HH0000
```

### Status Check:

The string "INFO: storm1: completed regridding job for" will appear in each of the POST standard output files, which will be set by the environment variable REGRIBBER\_LOGS in the products\_wrapper.

### Executables:

```
copygb.exe
```

This executable performs two functions. The functions are separated in the explanation by their respective numeric items.

#### FUNCTION:

1. Interpolates a GRIB file to a user-specified grid
2. Combines two GRIB files

#### INPUT:

1. `${hr_grid}` - User-specified grid  
One grib file, for example
2. `${hr_grid}` - User-specified grid  
Two GRIB files, for example

OUTPUT: GRIB file on grid `${hr_grid}`

#### USAGE:

1. `copygb.exe -xg "${hr_grid}" input_GRIB_file \ out_GRIB_file`
2. When an "-M" option is used and the argument following it is a GRIB file, the GRIB file will be interpreted as a merge file. This option can be used to combine two GRIB files.  
The following command will combine two GRIB files onto a third one named `out_GRIB_file`. All three files must use a grid specified by `${hr_grid}`.  
`copygb.exe -g "${hr_grid}" -xM input_GRIB_file_1 \ input_GRIB_file_2 out_GRIB_file`

## Tracker

---

The TrackerTask is responsible for running the GFDL Vortex Tracker. 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, as well as another track for the 12-hr forecast (stripped down version of `gonzalo081.YYYYMMDDHH.trak.hwrf.atcfunix` file), using a merged grid from all three domains with  $0.02^\circ$  resolution. The track for the 12-hr forecast is used in the vortex relocation procedure for the following cycle. The tracker module resides in `ush/hwrf/tracker.py` and performs the following actions:

1. Link the input GRIB files
2. Make the tracker namelist
3. Run `hwrf_gettrk.exe`
4. Deliver files

### Output files:

The following output files are an example for Hurricane Gonzalo (2014).

```
gonzalo081.YYYYMMDDHH.trak.hwrf.raw
gonzalo081.YYYYMMDDHH.trak.hwrf.atcfunix
gonzalo081.YYYYMMDDHH.trak.hwrf.3hourly
gonzalo081.YYYYMMDDHH.trak.hwrf.short6hr
```

Similar outputs are also generated from d01 and d02 domains with  $0.1^\circ$  and  $0.06^\circ$  resolution.

### Status Check:

The standard output file will contain the string "CRITICAL: Successful return status from `gettrk`."

### Executables:

```
hwrf_gettrk.exe
```

**FUNCTION:** Runs the GFDL Vortex Tracker

**INPUT:**

- `fort.11` - GRIB file containing the postprocessed HWRf forecast
- `fort.12` - TC Vitals file containing the first guess location of the forecast vortex
- `fort.14` - TC Vitals file used for tropical cyclogenesis 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
- `fort.31` - a GRIB index file generated by the program `grbindex`
- `input.namelist` - namelist

## 11. Forecast Products

OUTPUT: `fort.69` - Modified ATCF file  
`fort.64` - Modified ATCF file  
`fort.66` - Modified ATCF file produced only in "cyclgenesis mode"  
`fort.74` - Modified ATCF file produced when `IKEFLAY=Y`

USAGE: `hwrf_gettrk.exe <namelist`

Refer to Appendix C for a sample namelist and an explanation of contents of output files.

### 11.2.2 Additional Tracking Utilities

In addition to the actual tracking capability of the GFDL Vortex Tracker, the HWRF wrapper and Python scripts also automatically generate phase space diagnostics. Just for reference, the section below explains the steps to compute phase space diagnostics.

#### Phase-Space Diagnostics

---

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` to vertically interpolate the geopotential from 300 to 900 hPa at a 50-hPa 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` to vertically interpolate the temperature from 300 to 500 hPa at a 50-hPa interval. Then run `hwrf_tave.exe` to obtain the average temperature between 300 and 500 hPa. 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.

#### Executables:

`hwrf_vint.exe`

FUNCTION: Interpolates 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.

## 11. Forecast Products

**INPUT:** `fort.11` - GRIB file containing the postprocessed HWRP output with temperature at least at 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/" > namelist` 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:** `hwrp_vint.exe < namelist`

`hwrp_tave.exe`

**FUNCTION:** Vertically averages 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 `hwrp_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:** `hwrp_tave.exe < namelist`

## Obtaining Phase-Space Diagnostics when Running in Tracker Mode

---

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 `hwrp_vint.exe` to vertically interpolate the geopotential from 300 to 900 hPa at a 50-hPa interval. Then append these geopotential variables to the tracker's GRIB format input file.
3. If `phasescheme` is set to `vtt`, run `hwrp_vint.exe` to vertically interpolate the temperature from 300 to 500 hPa at a 50-hPa interval. Then run `hwrp_tave.exe` to

## 11. Forecast Products

obtain the average temperature between 300 and 500 hPa. 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 Appendix C).

### 11.3 How to Plot the Tracker Output Using ATCF\_PLOT

The GFDL Vortex Tracker comes with `atcf_plot`, a set of GrADS scripts that can be used to plot hurricane track files in ATCF format. These scripts can be found in the directory: `gfdl-vortextracker/trk_plot/plottrak`.

To use `atcf_plot` to plot the storm's track, perform the following steps:

- Enter the directory `gfdl-vortextracker/trk_plot`.
- Run `gribmap` on the GrADS `ctl` file `plottrak.ctl` by typing: **`gribmap -v -i plottrak.ctl`**. `gribmap` is a GrADS utility that maps the contents of the binary data using the `ctl` file. It creates a map (`plottrak.ix`) that points to the locations where the requested binary data start for the different variables and levels.

You should see one line in the output that has "MATCH" in the string. Both the `plot-track.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. `screddir` - path to the working directory (for example, `/${SCRATCH}/sorc/gfdl-vortextracker/trk_plot/plottrak`)
  4. `plotdir` - path to the directory where the plot files will be created (for example, `/${SCRATCH}/sorc/gfdl-vortextracker/trk_plot/plottrak/tracks`)
- Edit `atcfplot.gs` to define the following paths:
  1. `rundir` - same as `screddir` in `atcfplot.sh` (note `rundir` must end with a `/`)
  2. `netdir` - same as `plotdir` in `atcfplot.sh` (note `netdir` must end with a `/`)
  3. Change all instances of `HTUT` to `HCOM`
- Edit `get_mods.sh` to define the following paths:
  1. `rundir` - same as `screddir` 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 `screddir` in `atcfplot.sh`
  2. `netdir` - same as `plotdir` in `atcfplot.sh`
  3. `ndate` - path to the script `ndate.ksh`

## 11. Forecast Products

### 4. nhour - path to the script nhour.ksh

- The users need to insert or append their vortex tracker output, `fort.64`, into the file `aBASIN|SID|YYYY.dat`. The following two commands are an example of how to do this for Hurricane Gonzalo a-deck files:

```
sed -i 's/HWRF/HCOM/g' $COMhwrp/YYYYMMDDHH/08L/aa1082014_HWRF_hPYHW_YYYYMMDDHH.dat
cat $COMhwrp/YYYYMMDDHH/08L/aa1082014_HWRF_hPYHW_YYYYMMDDHH.dat aa1082014.dat
```

- After setting up the paths to the correct locations in your system, run the script using the following 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`.

For example, the user can use the command `atcfplot.sh 2011 a1` to plot the track files `aal${SID}2011.dat` in `$rundir`.

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 `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`, `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 `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".



# 12

## **HWRF Idealized Tropical Cyclone Simulation**

### 12.1 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  $20 \text{ ms}^{-1}$  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 an f-plane at the latitude of  $12.5^\circ$ . 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 18, 6, and 2 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 10) can be used to postprocess the idealized HWRF simulation output. The HWRF Python scripts are not suitable to run

## 12. HWRF Idealized Tropical Cyclone Simulation

the idealized simulation. Therefore, the user should refer to the DTC UPP website at <http://www.dtcenter.org/upp/users> and to the UPP User's Guide at [http://www.dtcenter.org/upp/users/docs/user\\_guide/V3/upp\\_users\\_guide.pdf](http://www.dtcenter.org/upp/users/docs/user_guide/V3/upp_users_guide.pdf) for more information on how to run UPP for the idealized HWRF simulation.

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  $\beta$ -plane.

## 12.2 How to Use HWRF for Idealized Tropical Cyclone Simulations

### 12.2.1 Source Code

This section describes the process to implement HWRF v3.7a in the idealized setting. 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`.

### 12.2.2 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 `gfs.t12z.pgrb2.0p25.f000` and `gfs.t12z.pgrb2.0p25.f120`, are included in the tar file [http://www.dtcenter.org/HurrWRF/users/downloads/datasets/Idealized\\_2015/hwrfv3.7a\\_idealized.tar.gz](http://www.dtcenter.org/HurrWRF/users/downloads/datasets/Idealized_2015/hwrfv3.7a_idealized.tar.gz).

Next the user must ensure that all the input files below exist in `${SCRATCH}/hwrf/sorc/WRFV3/test/nmm_tropical_cyclone`.

## 12. HWRF Idealized Tropical Cyclone Simulation

<code>namelist.wps</code>	Namelist file for WPS; Note that <code>geog_data_path</code> should be modified to point to the actual path of the geog data files.
<code>namelist.input</code>	Namelist file for WRF
<code>input.d</code>	Vortex description file
<code>sound.d</code>	Sounding data; four sounding files ( <code>sound.d</code> , <code>sound_gfdl.d</code> , <code>sound_jordan.d</code> , and <code>sound_wet.d</code> ) are provided in <code>\${SCRATCH}/hwrfrun/sorc/WRFV3/test/\nmm_tropical_cyclone</code> , however, only the one named <code>sound.d</code> will be used. To use a different sounding, rename it to <code>sound.d</code> .
<code>storm.center</code>	Vortex center file
<code>sigma.d</code>	Sigma file

### 12.2.3 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 3.5.1.

The wrappers and Python scripts 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.

### 12.2.4 Running WPS to Create the ICs and LBCs

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:

```
mkdir $WORKDIR/wpsprd  
cd $WORKDIR/wpsprd
```

2. Run `geogrid`

- a) Copy the WPS namelist.

```
cp ${SCRATCH}/hwrfrun/sorc/WRFV3/test/\  
nmm_tropical_cyclone/namelist.wps .
```

- b) Edit `namelist.wps` to make sure `geog_data_path` points to the location of the WPS geographical data files, which are included in the fix tarball in the `fix/hwrf_wps_geo`.

- c) Link the `geogrid` table.

```
ln -fs ${SCRATCH}/hwrfrun/sorc/WPSV3/geogrid/\  
GEOGRID.TBL.NMM ./GEOGRID.TBL
```

## 12. HWRF Idealized Tropical Cyclone Simulation

- d) Run executable `geogrid.exe` on the command line or submit it to a compute node or batch system.  
`${SCRATCH}/hwrfrun/sorc/WPSV3/geogrid.exe`
- e) Verify that the output files were created.  
`ls -l geo_nmm_nest.10[12].nc geo_nmm.d01.nc`
3. Run `ungrib`
  - a) Link the `ungrib` table.  
`ln -fs ${SCRATCH}/hwrfrun/sorc/WPSV3/ungrib/\`  
`Variable_Tables/Vtable.GFS ./Vtable`
  - b) Extract the two input GFS files.  
Download tarfile with GFS input data [http://www.dtcenter.org/HurrWRF/users/downloads/datasets/Idealized\\_2015/hwrfv3.7a\\_idealized.tar.gz](http://www.dtcenter.org/HurrWRF/users/downloads/datasets/Idealized_2015/hwrfv3.7a_idealized.tar.gz).  
`tar -xzf hwrfv3.7a_idealized.tar.gz`  
`gunzip gfs.t12z.pgrb2.0p25.f000.gz`  
`gunzip gfs.t12z.pgrb2.0p25.f120.gz`  
`ls -l gfs.t12z.pgrb2.0p25.f000 gfs.t12z.pgrb2.0p25.f120`
  - c) Link the GFS files to the names expected by `ungrib`.  
`${SCRATCH}/hwrfrun/sorc/WPSV3/link_grib.csh \`  
`gfs.t12z.pgrb2.0p25.f000 gfs.t12z.pgrb2.0p25.f120`
  - d) Run executable `ungrib.exe` on the command line or submit it to a compute node or batch system.  
`${SCRATCH}/hwrfrun/sorc/WPSV3/ungrib.exe`
  - e) Verify that the output files were created.  
`ls -l GFS:2012-10-26_12 GFS:2012-10-31_12`
4. Run `metgrid`
  - a) Link the `metgrid` table.  
`ln -fs ${SCRATCH}/hwrfrun/sorc/WPSV3/metgrid/\`  
`METGRID.TBL.NMM ./METGRID.TBL`
  - b) Run executable `mod_levels.exe` twice on the command line or submit it to a compute node or batch system. This program is used to reduce the number of vertical levels in the GFS file. Only the levels listed in variable `press_pa` in `namelist.wps` will be retained.  
`${SCRATCH}/hwrfrun/sorc/WPSV3/util/mod_levs.exe \`  
`GFS:2012-10-26_12 new_GFS:2012-10-26_12`  
`${SCRATCH}/hwrfrun/sorc/WPSV3/util/mod_levs.exe \`  
`GFS:2012-10-31_12 new_GFS:2012-10-31_12`
  - c) Verify that the output files were created.  
`ls -l new_GFS:2012-10-26_12 new_GFS:2012-10-31_12`
  - d) Run executable `metgrid.exe` on the command line or submit it to a compute node or batch system.  
`${SCRATCH}/hwrfrun/sorc/WPSV3/metgrid.exe`
  - e) Verify that the output files were created.  
`ls -l met_nmm.d01.2012-10-26_12_00_00.nc \`  
`met_nmm.d01.2012-10-31_12_00_00.nc`

## 12.2.5 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`.

```
mkdir $WORKDIR/wrfprd
cd $WORKDIR/wrfprd
```

2. Run `ideal`

- a) Link WRF input files.

```
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/ETAMPNEW_DATA .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/ETAMPNEW_DATA.expanded_rain
.
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/GENPARM.TBL .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/LANDUSE.TBL .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/SOILPARM.TBL .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/VEGPARM.TBL .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/tr49t67 .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/tr49t85 .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/tr67t85 .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/ozone.formatted .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/ozone_lat.formatted .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/ozone_plev.formatted
.
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/RRTM_DATA .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/RRTMG_LW_DATA .
ln -fs ${SCRATCH}/hwrfrun/sorc/WRFV3/run/RRTMG_SW_DATA .
```

- 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}/hwrfrun/sorc/WRFV3/test/\
nmm_tropical_cyclone/input.d .
cp ${SCRATCH}/hwrfrun/sorc/WRFV3/test/\
nmm_tropical_cyclone/sigma.d .
cp ${SCRATCH}/hwrfrun/sorc/WRFV3/test/\
nmm_tropical_cyclone/sound.d .
cp ${SCRATCH}/hwrfrun/sorc/WRFV3/test/\
nmm_tropical_cyclone/storm.center .
```

- d) Copy namelist input.

```
cp ${SCRATCH}/hwrfrun/sorc/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. To use a sounding with different number of levels, it is necessary to modify the source code in `${SCRATCH}/hwrfrun/sorc/WRFV3/dyn_nmm/`

## 12. HWRF Idealized Tropical Cyclone Simulation

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 submit it to a compute node or batch system.

```
`${SCRATCH}`/hwrfrun/sorc/WRFV3/main/ideal.exe
```

g) Verify that the output files were created.

```
ls -l wrfinput_d01 wrfdbdy_d01 fort.65
```

### 3. Run WRF

a) Run executable wrf.exe on the command line or submit it to a compute node or batch system.

```
`${SCRATCH}`/hwrfrun/sorc/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*
```



# Example of Computational Resources

Table A.1 gives an example of the resources required to run HWRF compiled with Intel in operations on the NOAA Research Supercomputer Jet and should be used as a guideline for scaling to the resources at the individual user's disposal. In the example below, the available resources include access to 5440 cores with 16 2.6 GHz cores per node. Each node has 32 GB (2 GB per core). With peak performance capable of 113.2 TF, and end-to-end run of HWRF would take about four hours.

A. Example of Computational Resources

	<b>Wall clock time</b>	<b>Total Cores</b>	<b>Core Layout</b>	<b>Virtual Memory</b>
launcher_wrapper	15:00	1	n/a	n/a
init_gdas_wrapper	1:39:00	48	n/a	n/a
init_gfs_wrapper	1:39:00	48	n/a	30 GB
init_ocean_wrapper	0:59:00	9	n/a	30 GB
bufwrwrapper	0:19:00	1	n/a	30 GB
relocate_wrapper	2:19:00	n/a	nodes=1	20 GB
gsi_d02_wrapper	1:49:00	60	nodes=10:ppn=6	25 GB
gsi_d03_wrapper	1:49:00	80	nodes=10:ppn=8	25 GB
merge_wrapper	1:39:00	1	n/a	n/a
unpost_wrapper	0:05:00	1	n/a	n/a
forecast_wrapper	4:59:00	477	see Table A.2	n/a
post_wrapper	5:59:00	24	n/a	25 GB
products_wrapper	5:59:00	1	n/a	30 GB

Table A.1: Example of resources required to run a 5-day HWRF forecast at near operational capability.

<b>Configuration</b>	<b>Total Cores</b>	<b>Core Layout</b>	<b>nproc_x</b>	<b>nproc_y</b>
2km Coupled (default)	477	32:ppn=12+7:ppn=11+2:ppn=8	16	28
2km Uncoupled	464	30:ppn=12+8:ppn=11+2:ppn=8	16	28
3km Coupled	210	1:ppn=10+8:ppn=11+8:ppn=12+2:ppn=8	8	23
3km Uncoupled	200	8:ppn=11+8:ppn=12+2:ppn=8	8	23

Table A.2: Example of forecast\_wrapper required resources.



## Example HWRF Namelist

The HWRF namelist used for the release case, Hurricane Gonzalo (2014), is listed below. Note: the namelist is generated by the HWRF scripts based on the settings in the configuration files of the `parm` directory.

```
&time_control
start_year           = 2014, 2014, 2014,
start_month          = 10, 10, 10,
start_day            = 14, 14, 14,
start_hour           = 12, 12, 12,
start_minute         = 0, 0, 0,
start_second         = 0, 0, 0,
end_year             = 2014, 2014, 2014,
end_month            = 10, 10, 10,
end_day              = 19, 19, 19,
end_hour             = 18, 18, 18,
end_minute           = 0, 0, 0,
end_second           = 0, 0, 0,
interval_seconds     = 21600,
history_interval     = 180, 180, 180,
auxhist1_interval    = 60, 60, 60,
auxhist2_interval    = 60, 60, 60,
auxhist3_interval    = 180, 180, 180,
history_end          = 540, 540, 540,
auxhist2_end         = 540, 540, 540,
auxhist1_outname     = "wrfdiag_d<domain>",
auxhist2_outname     = "wrfout_d<domain>_<date>",
auxhist3_outname     = "wrfout_d<domain>_<date>",
frames_per_outfile   = 1, 1, 1,
frames_per_auxhist1  = 999, 999, 999,
frames_per_auxhist2  = 1, 1, 1,
frames_per_auxhist3  = 1, 1, 1,
analysis             = F, T, T,
```

## B. Example HWRF Namelist

```
restart = F,
restart_interval = 36000,
reset_simulation_start = F,
io_form_input = 11,
io_form_history = 11,
io_form_restart = 11,
io_form_boundary = 11,
io_form_auxinput1 = 2,
io_form_auxhist1 = 202,
io_form_auxhist2 = 11,
io_form_auxhist3 = 11,
auxinput1_inname = "met_nmm.d<domain>.<date>",
debug_level = 1,
tg_reset_stream = 1,
override_restart_timers = T,
io_form_auxhist4 = 11,
io_form_auxhist5 = 11,
io_form_auxhist6 = 11,
io_form_auxinput2 = 2,
nocolons = T,
/
&fdda
/
&domains
time_step = 38,
time_step_fract_num = 4,
time_step_fract_den = 7,
max_dom = 3,
s_we = 1, 1, 1,
e_we = 288, 142, 265,
s_sn = 1, 1, 1,
e_sn = 576, 274, 472,
s_vert = 1, 1, 1,
e_vert = 61, 61, 61,
dx = 0.135, 0.045, 0.015,
dy = 0.135, 0.045, 0.015,
grid_id = 1, 2, 3,
tile_sz_x = 0,
tile_sz_y = 0,
numtiles = 1,
nproc_x = 16,
nproc_y = 28,
parent_id = 0, 1, 2,
parent_grid_ratio = 1, 3, 3,
parent_time_step_ratio = 1, 3, 3,
i_parent_start = 0, 125, 27,
j_parent_start = 0, 243, 57,
feedback = 1,
num_moves = -99,
num_metgrid_levels = 4,
p_top_requested = 200.0,
ptsgm = 15000.0,
```

## B. Example HWRF Namelist

```
eta_levels = 1.0, 0.995253, 0.990479, 0.985679, 0.980781,  
0.975782, 0.970684, 0.965486, 0.960187,  
0.954689, 0.948991, 0.943093, 0.936895,  
0.930397, 0.923599, 0.916402, 0.908404,  
0.899507, 0.888811, 0.876814, 0.862914,  
0.847114, 0.829314, 0.809114, 0.786714,  
0.762114, 0.735314, 0.706714, 0.676614,  
0.645814, 0.614214, 0.582114, 0.549714,  
0.517114, 0.484394, 0.451894, 0.419694,  
0.388094, 0.356994, 0.326694, 0.297694,  
0.270694, 0.245894, 0.223694, 0.203594,  
0.185494, 0.169294, 0.154394, 0.140494,  
0.127094, 0.114294, 0.101894, 0.089794,  
0.078094, 0.066594, 0.055294, 0.044144,  
0.033054, 0.022004, 0.010994, 0.0,  
use_prep_hybrid = T,  
num_metgrid_soil_levels = 4,  
coral_x = 6, 9, 8,  
coral_y = 6, 18, 16,  
smooth_option = 0,  
/  
&physics  
num_soil_layers = 4,  
mp_physics = 5, 5, 5,  
ra_lw_physics = 4, 4, 4,  
ra_sw_physics = 4, 4, 4,  
sf_sfclay_physics = 88, 88, 88,  
sf_surface_physics = 2, 2, 2,  
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 = 28, 84, 252,  
nradl = 28, 84, 252,  
nphs = 2, 6, 6,  
ncnvc = 2, 6, 6,  
movemin = 7, 7, 14,  
gfs_alpha = -1.0, -1.0, -1.0,  
sas_pgcon = 0.55, 0.2, 0.2,  
sas_mass_flux = 0.5, 0.5, 0.5,  
co2tf = 1,  
vortex_tracker = 2, 2, 7,  
nomove_freq = 0, 6, 6,  
tg_option = 1,  
ntornado = 2, 5, 15,  
ens_pblamp = 0.2,  
ens_random_seed = 99,  
ens_sasamp = 50.0,  
icloud = 3,  
icoef_sf = 2, 2, 2,  
lcurr_sf = F, F, F,  
pert_pbl = F,  
pert_sas = F,
```

## *B. Example HWRF Namelist*

```
/
&dynamics
non_hydrostatic      = T, T, T,
euler_adv            = F,
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             = T,
/
&namelist_quilt
poll_servers          = T,
nio_tasks_per_group  = 4,
nio_groups            = 4,
/
&logging
compute_slaves_silent = T,
io_servers_silent     = T,
stderr_logging        = 0,
/
```



# Additional GFDL Tracker Information

## Sample namelist

<code>inp%bcc</code>	First two digits of the year for the initial time of the forecast (e.g., the "20" in "2012").
<code>inp%byy</code>	Last two digits of the year for the initial time of the forecast (e.g., the "12" in "2012").
<code>inp%bmm</code>	Two-digit month (01, 02, etc) for the initial time of the forecast.
<code>inp%bdd</code>	Two-digit day for the initial time of the forecast.
<code>inp%bhh</code>	Two-digit hour for the initial time of the forecast.
<code>inp%model</code>	Model ID number as defined by the user in the script. This is used in subroutine <code>getdata</code> 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 <code>getdata</code> . 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) HDAS,
- (20) Ensemble RELOCATION (21) UKMET hi-res (NHC)

### C. Additional GFDL Tracker Information

inp%modtyp	Type of the model. Either "global" or "regional". For HWRF, choose "regional".
inp%lt_units	"hours" or "minutes", this defines the lead time unit 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%nesttyp	Type of the nest grid. Either "moveable" or "fixed". For HWRF, choose "moveable".
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%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 TC Vitals. "tcgen" and "midlat" run in genesis mode and will look for new storms in addition to tracking from TC Vitals. "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 hPa/km) that must be met to continue tracking.
trkrinfo%v850thresh	Threshold for the minimum azimuthally-average 850 hPa 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 "TC Vitals" record for any storms that it finds at the model initialization time. For HWRF, choose "n".
trkrinfo%g2_jpdtn	This is set to zero if trying to read deterministic model data, and 1 if trying to read ensemble model data.
trkrinfo%gribver	This is set to 1 if reading GRIB1 files. This is the only version supported with HWRF v3.7a.
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.

### C. Additional GFDL Tracker Information

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.
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-hPa 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-hPa 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).
gmodname	Defines the model name in the input files, e.g., "hwrp". Only when inp%file_seq="multi".
rundescri	Describes the model runs in the input files, e.g., "combined". Only when inp%file_seq="multi".
atcfdescri	Describe the storm information in the input files, e.g., "irene091". Only when inp%file_seq="multi".
verb	Level of detail printed to terminal. Choose from 0 (no output), 1 (error messages only), 2 (more messages), 3 (all messages).
use_waitfor	"y" or "n", for waiting for input files. Use "n" unless for real-time operational runs.

### Contents of the output files

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

```
AL, 08, 2014101412, 03, HWRF, 00000, 197N, 649W, 97, 973, XX, 34, NEQ, 0120, 0086, 0036, 0117, 0, 0, 25
AL, 08, 2014101412, 03, HWRF, 00000, 197N, 649W, 97, 973, XX, 50, NEQ, 0062, 0034, 0014, 0059, 0, 0, 25
AL, 08, 2014101412, 03, HWRF, 00000, 197N, 649W, 97, 973, XX, 64, NEQ, 0043, 0021, 0002, 0040, 0, 0, 25
AL, 08, 2014101412, 03, HWRF, 00100, 198N, 650W, 84, 962, XX, 34, NEQ, 0097, 0066, 0034, 0089, 0, 0, 25
AL, 08, 2014101412, 03, HWRF, 00100, 198N, 650W, 84, 962, XX, 50, NEQ, 0047, 0033, 0014, 0042, 0, 0, 25
AL, 08, 2014101412, 03, HWRF, 00100, 198N, 650W, 84, 962, XX, 64, NEQ, 0035, 0017, 0000, 0031, 0, 0, 25
AL, 08, 2014101412, 03, HWRF, 00200, 200N, 652W, 80, 967, XX, 34, NEQ, 0104, 0073, 0051, 0092, 0, 0, 26
AL, 08, 2014101412, 03, HWRF, 00200, 200N, 652W, 80, 967, XX, 50, NEQ, 0051, 0032, 0015, 0046, 0, 0, 26
AL, 08, 2014101412, 03, HWRF, 00200, 200N, 652W, 80, 967, XX, 64, NEQ, 0037, 0000, 0007, 0034, 0, 0, 26
AL, 08, 2014101412, 03, HWRF, 00300, 203N, 653W, 77, 970, XX, 34, NEQ, 0097, 0080, 0062, 0085, 0, 0, 16
AL, 08, 2014101412, 03, HWRF, 00300, 203N, 653W, 77, 970, XX, 50, NEQ, 0044, 0031, 0024, 0036, 0, 0, 16
AL, 08, 2014101412, 03, HWRF, 00300, 203N, 653W, 77, 970, XX, 64, NEQ, 0028, 0020, 0017, 0024, 0, 0, 16
AL, 08, 2014101412, 03, HWRF, 00400, 205N, 654W, 75, 971, XX, 34, NEQ, 0093, 0080, 0065, 0087, 0, 0, 15
AL, 08, 2014101412, 03, HWRF, 00400, 205N, 654W, 75, 971, XX, 50, NEQ, 0039, 0032, 0035, 0031, 0, 0, 15
AL, 08, 2014101412, 03, HWRF, 00400, 205N, 654W, 75, 971, XX, 64, NEQ, 0024, 0022, 0020, 0022, 0, 0, 15
AL, 08, 2014101412, 03, HWRF, 00500, 206N, 655W, 78, 970, XX, 34, NEQ, 0097, 0081, 0060, 0084, 0, 0, 14
AL, 08, 2014101412, 03, HWRF, 00500, 206N, 655W, 78, 970, XX, 50, NEQ, 0039, 0035, 0041, 0034, 0, 0, 14
AL, 08, 2014101412, 03, HWRF, 00500, 206N, 655W, 78, 970, XX, 64, NEQ, 0026, 0024, 0021, 0023, 0, 0, 14
AL, 08, 2014101412, 03, HWRF, 00600, 208N, 657W, 82, 970, XX, 34, NEQ, 0110, 0083, 0061, 0088, 0, 0, 15
```

### C. Additional GFDL Tracker Information

Column 1:	basin name. "AL" represents Atlantic and "EP" northeast Pacific.
Column 2:	ATCF storm ID number. Gonzalo was the 8th storm in the Atlantic Basin in 2014.
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 h).
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, 08, 2014101412, 03, HWRF, 000, 197N, 649W, 97, 973, XX, 34, NEQ, 0120, 0086, 0036, 0117, -99, -99, 25, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 000, 197N, 649W, 97, 973, XX, 50, NEQ, 0062, 0034, 0014, 0059, -99, -99, 25, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 000, 197N, 649W, 97, 973, XX, 64, NEQ, 0043, 0021, 0002, 0040, -99, -99, 25, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -9999, -9999, -9999, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 001, 198N, 650W, 84, 962, XX, 34, NEQ, 0097, 0066, 0034, 0089, -99, -99, 25, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -57, 1476, 2582, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 001, 198N, 650W, 84, 962, XX, 50, NEQ, 0047, 0033, 0014, 0042, -99, -99, 25, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -57, 1476, 2582, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 001, 198N, 650W, 84, 962, XX, 64, NEQ, 0035, 0017, 0000, 0031, -99, -99, 25, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -57, 1476, 2582, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 002, 200N, 652W, 80, 967, XX, 34, NEQ, 0104, 0073, 0051, 0092, -99, -99, 26, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -60, 1437, 2067, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 002, 200N, 652W, 80, 967, XX, 50, NEQ, 0051, 0032, 0015, 0046, -99, -99, 26, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -60, 1437, 2067, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 002, 200N, 652W, 80, 967, XX, 64, NEQ, 0037, 0000, 0007, 0034, -99, -99, 26, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -60, 1437, 2067, Y, 10, DT, -999
AL, 08, 2014101412, 03, HWRF, 003, 203N, 653W, 77, 970, XX, 34, NEQ, 0097, 0080, 0062, 0085, -99, -99, 16, 0, 0, , 0, , 0, 0, , , , 0, 0, 0, 0, THERMO PARAMS, -55, 1688, 1824, Y, 10, DT, -999
```

Column 1-20:	same as <code>fort.69</code> except that column 6, the forecast lead time, instead of being a 5-digit integer as in <code>fort.69</code> , is a 3-digit integer.
Column 21-35:	space fillers.
Column 36:	"THERMO PARAMS" indicating that thermodynamics parameters will follow.

### C. Additional GFDL Tracker Information

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

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 hPa 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 hPa).

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

Column 21: radius of maximum wind (nm).

### C. Additional GFDL Tracker Information

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

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 tenths of  $\text{ms}^{-1}$ ).

Column 28: mean 850-hPa vorticity ( $\text{s}^{-1}\times 10^5$ ).

Column 29: max (gridpoint) 850-hPa vorticity ( $\text{s}^{-1}\times 10^5$ ).

Column 30: mean 700 hPa vorticity ( $\text{s}^{-1}\times 10^5$ ).

Column 31: max (gridpoint) 700 hPa vorticity ( $\text{s}^{-1}\times 10^5$ ).

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  $10 \text{ ms}^{-1}$ ,  $18 \text{ ms}^{-1}$  and  $33 \text{ ms}^{-1}$  winds, respectively.

Column 19-21: IKE for  $25\text{-}40 \text{ ms}^{-1}$ ,  $41\text{-}54 \text{ ms}^{-1}$  and  $55 \text{ ms}^{-1}$  winds, currently not computed, therefore are always zero.

Column 22-23: vortex center position (latitude and longitude multiplied by 100).