FV3-LAM / SRW App.

(FV3-Limited Area Model / Short Range Weather Application)

(formerly known as '**FV3-SAR**')

Chan-Hoo Jeon, Ph.D.

Scientific Programmer/Analyst

Engineering and Implementation Branch (EIB)

NOAA/NWS/NCEP/EMC

August 5, 2020

Contents

.

Chapter 1

Introduction

1. **Basic level** : Runs a regional workflow.
2. **Intermediate level** : Creates a new regional case or modifies the regional workflow.
3. **Advanced level** : Modifies the source code of '*FV3*'.

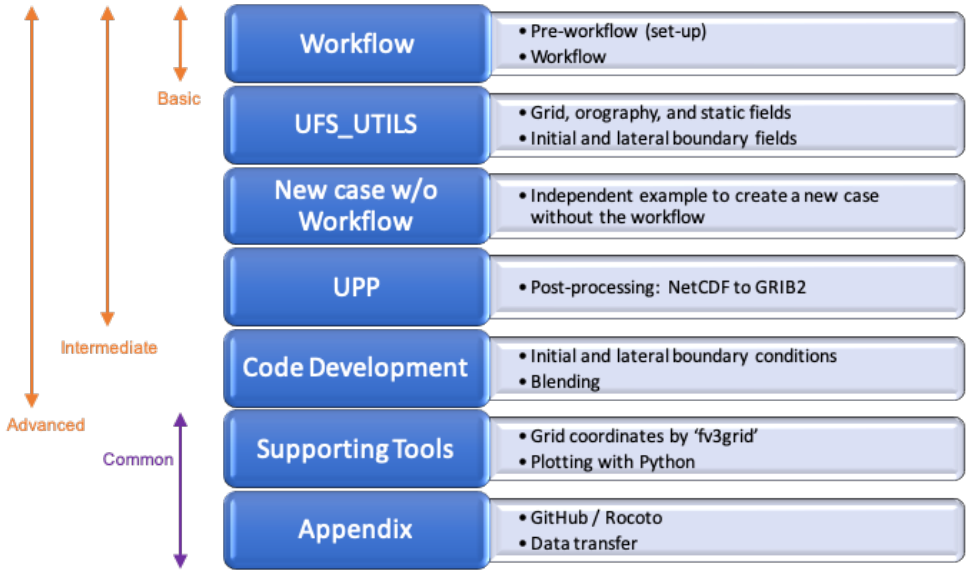

Figure 1.1: Structure of the document

Chapter 2

Pre-workflow with UFS Short Range Weather Application

2.1 Overall Procedure of SRW App

The overall procedure of the regional workflow in the UFS Short Range Weather Application (SRW App) is illustrated in Figure 2.1. Its steps are as follows:

1. Clone the SRW App from GitHub.
2. Check out the external components.
3. Build the executables.
4. Check if the pre-defined grid exists.
5. Modify the user-specific configuration file.
6. Generate a regional workflow experiment.
7. Load the appropriate *python* environment for the regional workflow
8. Run the regional workflow.

Each step will be described in detail in the following sections.

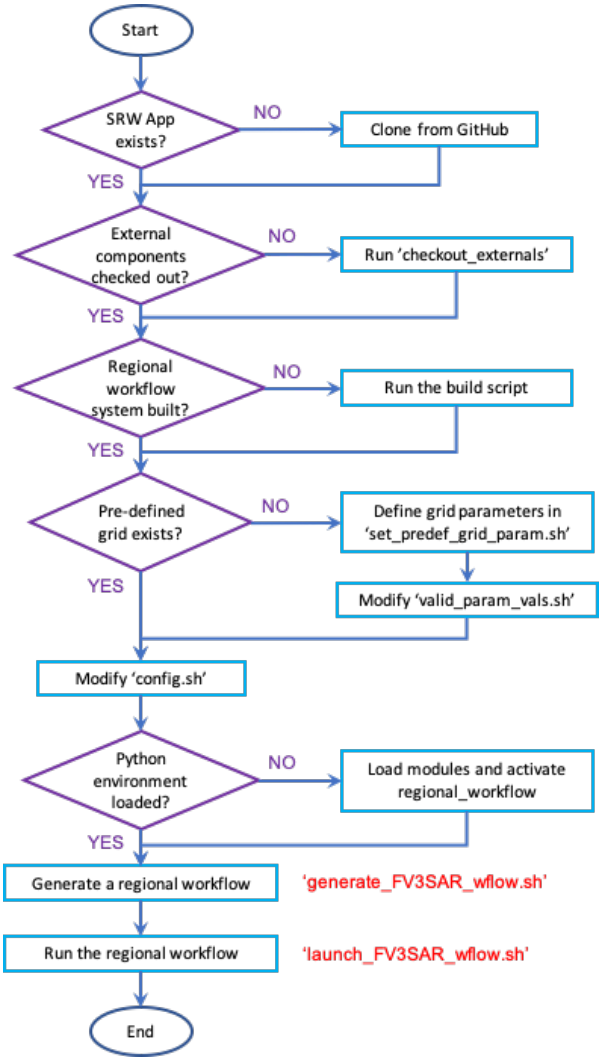

Figure 2.1: Overall procedure of Short Range Weather App

2.2 Download from GitHub

Retrieve the 'UFS Short Range Weather Application (SRWeather App)' repository from GitHub as follows:

```
git clone https://github.com/ufs-community/ufs-srweather-app.git
```

The cloned repository contains the following sub-repositories and files:

No.	Directory/file name	Description
1	docs	Release notes, documentation, user‘s guides
2	manage_externals	Method for checking out external components
3	src	Build scripts are initially located, and the external components will be cloned in this directory.
4	CMakeLists.txt	Main *cmake* file for UFS Short Range Weather Application
5	Externals.cfg	Hashes of the GitHub repositories/branch for the external components
6	LICENSE.md	(empty)
7	README.md	Quick user’s guide
8	ufs_srweather_app_meta.h.in	Meta information for UFS Short Range Weather Application which can be used by other packages
9	ufs_srweather_app.settings.in	UFS Short Range Weather App configuration summary
10	regional_workflow	Regional workflow (This directory will be created by running ‘checkout_externals’ in Section 2.3).
11	exec	Executables for workflow (This directory will be created by running ‘build_all.sh’ in Section 2.4.)
12	fix	Time-independent (fix) files in ‘fix_am’ will be copied here.

Table 2.1: Sub-directories of the short range weather application

2.3 External Components

Check out the external components including the regional workflow for the short range weather application:

```
cd {HOME}
./manage_externals/checkout_externals
```

where ‘{HOME}’ is the path to the ‘ufs-srweather-app’ repository. For example, it can be ‘scratch2/NCEPDEV/fv3-cam/{LOGNAME}/ufs-srweather-app’ on Hera.

Note) It should be run from the {HOME} directory where the ‘Externals.cfg’ file exists to avoid the root error saying ‘Model description file does not exist at path’.

This step will use the configuration file 'Externals.cfg' in the '{HOME}' directory to clone the develop branch of the regional workflow as well as the specific **hashes** (versions of codes) of the GitHub repositories/branches. As a result, the regional workflow are cloned in the {HOME} directory while the external components are cloned into the target sub-directories in the '{HOME}/src' directory as follows:

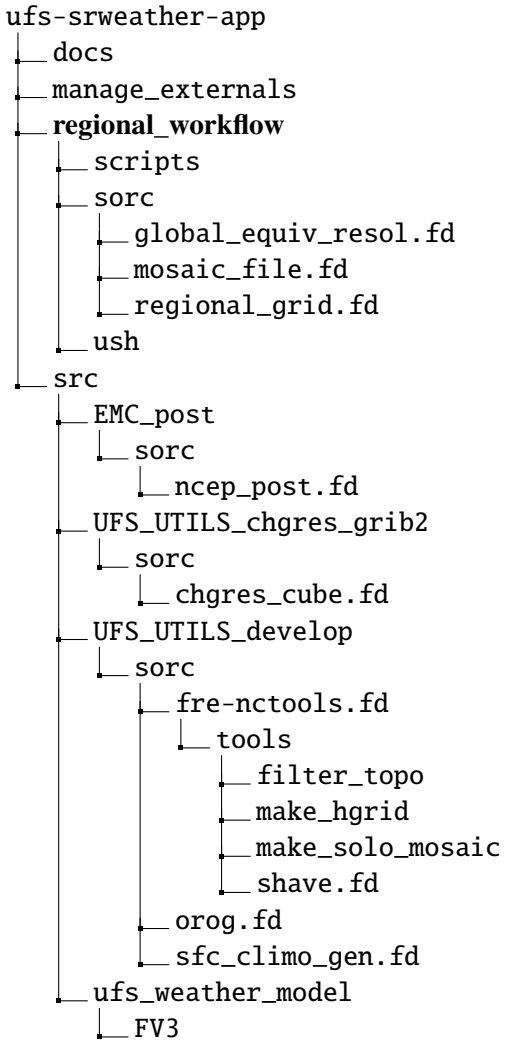

Figure 2.2: Structure of the Short Range Weather App

No.	Component	GitHub repository	Target sub-directory
1	regional_workflow	NOAA-EMC/regional_workflow	regional_workflow
2	ufs_utils	NCAR/UFS_UTILS	src/UFS_UTILS_develop
3	ufs_utils_chgres	NCAR/UFS_UTILS	src/UFS_UTILS_chgres_grib2
4	ufs_weather_model	NCAR/ufs-weather-model	src/ufs_weather_model
5	EMC_post	NOAA-EMC/EMC_post	src/EMC_post

Table 2.2: GitHub repository for the external components

Notes)

- The target sub-directories are not shown in the {HOME} or '{HOME}/src' directory until the 'checkout_externals' is executed.
- The codes of 'fre-nctools', 'orog', and 'sfc_climo_gen' are located in '{HOME}/src/UFS_ UTILS_develop/' ('ufs_utils' component).
- The 'chgres_cube' is located in '{HOME}/src/UFS_ UTILS_chgres_grib2/' ('ufs_utils_ chgres' component).

The regional workflow contains the sub-directories and files as follows:

```
cd {HOMErrfs}
```

where '{HOMErrfs}' is the path to the 'regional_workflow' repository. For example, it can be 'scratch2/NCEPDEV/fv3-cam/{LOGNAME}/ufs-srweather-app/regional_workflow' on Hera.

No.	Directory/file name	Description
1	docs	Release notes, documentation, user's guides
2	envs	Machine-specified environment variables
3	fix	fixed (time-independent) field data
4	jobs	J-job scripts which are launched by *Rocoto*
5	modulefiles	Module files required for running the model
6	scripts	Run scripts launched by the J-jobs
7	sorc	Build scripts, link scripts of fix files, source codes of some pre-processing tools

8	tests	Baseline experiment configurations
9	ush	Utility scripts called by various run scripts
10	environment.yml	
11	README.md	Quick user's guide to run a FV3-SAR workflow
12	update_fork.pl	

Table 2.3: Sub-directories of the regional workflow

Notes) The pre-processing tasks require the following three additional codes in the '{HOMErrfs}/sorc' directory:

1. 'regional_grid'
2. 'mosaic_file'
3. 'global_equiv_resol'

We have set up the 'Externals.cfg' file such that 'manage_externals' clones specific hashes from the various repositories/branches specified above. We choose the combination of hashes such that the resulting set of codes can successfully run the workflow end-to-end for a set of baseline experiment configurations. If you are curious about what these baselines are, their configuration files are located in '{HOMErrfs}/tests/baseline_configure'.

Workflow with Another Hash or HEAD

If you want to run the workflow with another hash or with the latest version (HEAD) of a given repository/branch (e.g. because there is a feature in that hash or in HEAD that you need), you will have to modify the 'Externals.cfg' file before issuing the above './checkout_externals' command.

1. Check the current hash or HEAD of the external component in 'Externals.cfg':

```
[ufs_utils]
protocol = git
repo_url = https://github.com/NCAR/UFS_UTILS
# Specify either a branch name or a hash but not both.
#branch = dtc/develop
hash = 5ddf829e
local_path = sorc/UFS_UTILS_develop
```

```
required = True
```

Note that the 'branch' property in the above block (for 'ufs_utils') is commented out (using the # character), and instead the 'hash' property is specified.

- In Git, you can obtain the branch that a given hash belongs to by using the 'git name-rev' command. To verify that the above hash (5ddf829e) is part of the history of the 'dtc/develop' branch, do the following:

  ```
  cd {HOMErrfs}/sorc/UFS_UTILS_develop/
  git name-rev 5ddf829e
  ```

  This will print the following to stdout:

  ```
  5ddf829e remotes/origin/dtc/develop~2
  ```

  You can see from the portion of this output that this hash is part of the 'dtc/develop' branch.

2. To have 'manage_externals' obtain a different hash, simply specify a different hash in the line 'hash=5ddf829e' in the block above.

3. To obtain the HEAD of the 'dtc/develop' branch, uncomment the line specifying the branch and comment out the line specifying the hash as follows:

```
branch = dtc/develop
#hash = 5ddf829e
```

2.4 Building a Regional Workflow System

Build a regional workflow system including the preprocessing utilities, forecast model, and post-processor:

```
cd {HOME}/src
module unload miniconda3
./build_all.sh >& out.build_all.log &
```

Note) On Hera and Jet, 'miniconda3' **CAN NOT** be loaded for this step, or the forecast model will fail to build. To be safe, you can use the 'unload' command before running the 'build_all.sh' script.

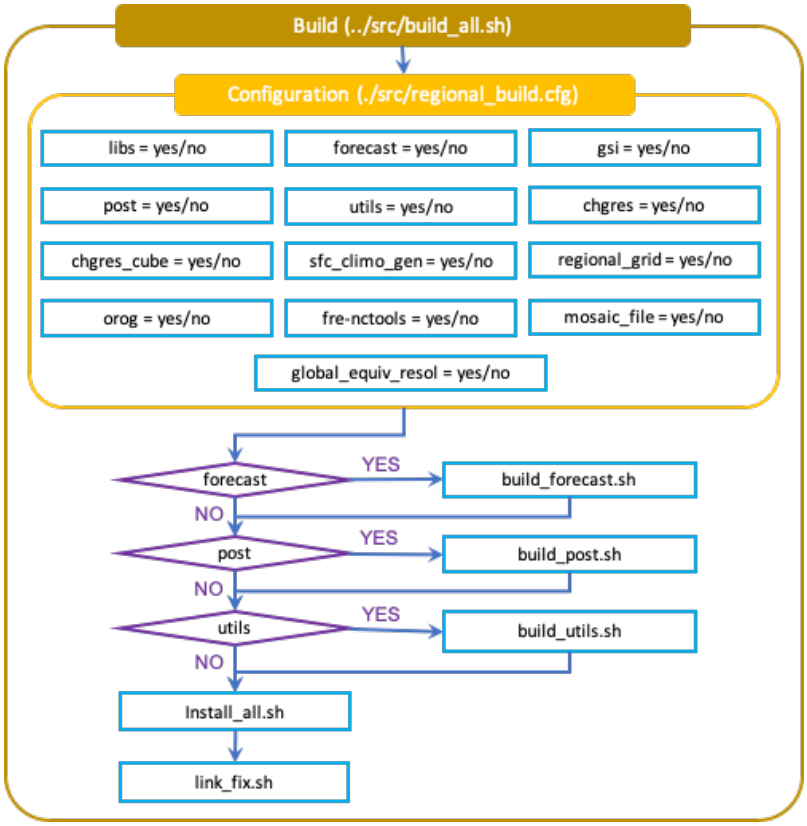

Figure 2.3: Structure of building a workflow system

During the build, log files will be written in the '{HOME}/src/logs' directory. You can follow the progress of the build using the 'tail' command:

```
tail -f $HOME/src/logs/build_forecast.log
tail -f $HOME/src/logs/build_post.log
```

1. Build forecast ('build_forecast.sh')
   - Directory: {HOMErrfs}/sorc/ufs_weather_model/tests/
   - Compile 'FV3'
     (a) Without CCPP: 'NCEP64LEV=Y HYDRO=N 32BIT=Y'
     (b) With CCPP ($CCPP='true'): 'CCPP=Y STATIC=N 32BIT=Y REPRO=Y'
     – 'CCPP=Y': has CCPP enabled.

  - 'STATIC=N': uses dynamic CCPP libraries. This is the default, but we include it for clarity. This allows flexibility (swapping of physics schemes, including whole physics suites, at runtime) at the expense of performance and memory footprint.
  - '32BIT=Y': Floating-point numbers in the dynamics portion of the source code are single-precision. Note that floating-point numbers in the physics parameterizations are always double-precision.
  - 'REPRO=Y': removes certain compiler optimization flags to obtain bit-for-bit identical results between CCPP-enabled and non-CCPP-enabled versions.
- Executable file: fv3.exe (in '{HOMEsar}/sorc/ufs_weather_model/tests/')

2. Build post ('build_post.sh')

   - Directory: {HOMErrfs}/sorc/EMC_post.fd/
   - Run 'build_ncep_post.sh'.

   Note) Make sure that the required modules in '{HOMErrfs}/sorc/EMC_post.fd/modulefiles/post/v8.0.0-[machine]' are available on HPC.

3. Build utils ('build_utils.sh')

   - Build 'chgres_cube' ('build_chgres_cube.sh').
   - Build 'orog' ('build_orog.sh').
   - Build 'fre-nctools' ('build_fre-nctools.sh').
   - Build 'sfc_climo_gen' ('build_sfc_climo_gen.sh').
   - Build 'regional_grid' ('build_regional_grid.sh').
   - Build 'global_equiv_resol' ('build_global_equiv_resol.sh').
   - Build mosaic file ('build_mosaic_file.sh').

4. Set up executables ('install_all.sh')

   After the build completes, you should see the fourteen pre- and post-processing executables in '{HOME}/exec/':

No.	Name	Description
1	chgres_cube.exe	Creates IC/LBC
2	filter_topo	Filtering
3	fregrid	Converts grid resolution
4	fregrid_parallel	Parallel version of 'fregrid'
5	global_equiv_resol	Calculates the regional grid's global uniform cubed-sphere grid equivalent resolution
6	make_hgrid	Creates GFDL grid files
7	make_hgrid_parallel	Creates GFDL grid files in parallel
8	make_solo_mosaic	Creates mosaic files with halos
9	mosaic_file	Creates the grid mosaic file that FV3 expects in the 'INPUT' sub-directory
10	ncep_post	Post-processing
11	orog.x	Creates orography files
12	regional_grid	Creates ESG (JP) grid files
13	sfc_climo_gen	Creates static field files
14	shave.x	Creates halo files

Table 2.4: List of executables.

5. Link the fix files ('link_fix.sh')

   The sub-directory 'fix_am' (in '{HOMErrfs}/fix') for the static (time-independent) global input data is linked as follows:

   - Hera:

   ```
   ln -sfn /scratch2/NCEPDEV/global/glopara/fix/fix_am fix_am
   ```

   - Orion:

   ```
   ln -sfn
   ```

   - WCOSS:

   ```
   ln -sfn /gpfs/dell2/emc/modeling/noscrub/emc.campara/fix_fv3cam fix_am
   ```

2.5 Generating a Regional Workflow

2.5.1 Flowchart for Generating a Regional Workflow

Create a user-specified experiment configuration file ('{HOMErrfs}/ush/config.sh') and specify the values of your experiment parameters in it:

```
cd {HOMErrfs}/ush
cp config.[option].sh config.sh
# Check and modify 'config.sh' as needed.
# In NCO mode, link 'FIXsar' (Section 2.4.3)
```

where 'config.[option].sh' is described in Section 2.5.3. For users' convenience, we provide two example configuration files in '{HOMErrfs}/ush/' named config.community.sh and config.nco.sh. The first is a minimal example for creating and running an experiment in the 'community' mode (i.e. with the parameter 'RUN_ENVIR' set to "community"), while the second is an example of creating and running an experiment in the 'NCO' (operational) mode (i.e. with 'RUN_ENVIR' set to "nco").

Load the appropriate python environment for the workflow. The workflow requires *Python3* with the packages 'PyYAML', 'Jinja2', and 'f90nml' available.

- On Hera:

```
module use -a /contrib/miniconda3/modulefiles
module load miniconda3
conda activate regional_workflow
```

For 'nco' mode, link the 'fix_am' directory of '{HOME}/fix/' into '{HOMErrfs}/fix/':

```
cd {HOMErrfs}/fix/
ln -sfn {HOME}/fix/fix_am .
```

Generate the workflow:

```
cd {HOMErrfs}/ush
./generate_FV3SAR_wflow.sh
```

The flowchart for generating a new regional workflow is shown in Figure 2.4.

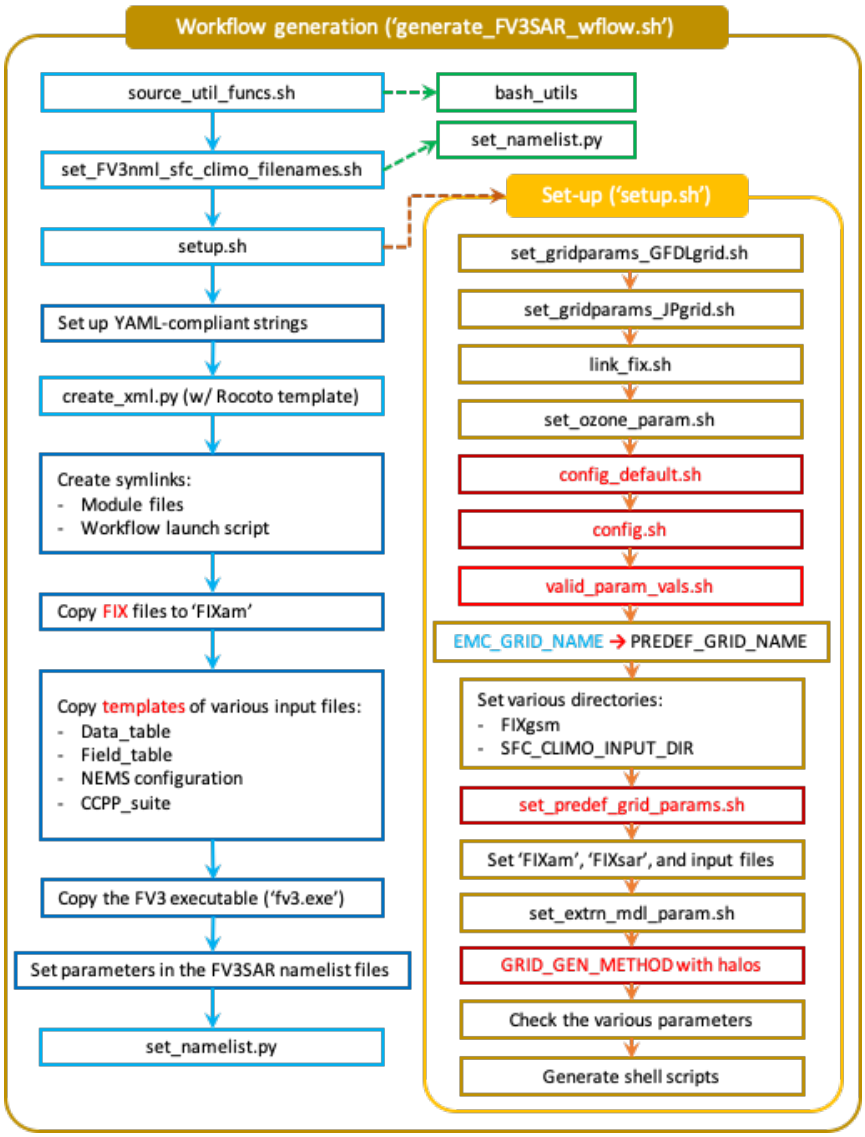

Figure 2.4: Structure of workflow generation

- The default configuration file ('{HOMErrfs}/ush/config_default.sh') assigns default values to the experiment parameters. Some of these default values are intentionally invalid in order to force the user to assign them valid values in the user-specified configuration file. There is usually no need for a user to modify the default configuration file. Note that the default configuration file also contains documentation describing the experiment parameters.

- Parameter settings in the user-specified configuration file (‘{HOMErrfs}/ush/config.sh’) will override settings in the default configuration file because the user-specified configuration file is sourced after the default one.
- As shown in the flowchart, the grid(domain)-specific parameters such as the GFDL grid refinement or ESG (JP) grid, write-component parameters when ‘QUILTING’=“TRUE”, ‘DT_ATMOS’, ‘LAYOUT_X(Y)’, and ‘BLOCKSIZE’, are separately set in ‘{HOMErrfs} /ush/set_predef_grid_params.sh’. If you want to use other regional domains that are not defined in this file, you should modify this file.
- The settings in the user-specified configuration file (‘config.sh’) depend on not only the user (e.g. the machine they are on, the account under which they are running experiments) but also on the experiment the user wants to run (e.g. grid on which to run the forecast, the length of the forecast, etc). For this reason, it is not included in the code base and thus must be created by the user.

2.5.2 Default Configuration: ‘config_default.sh’

This file sets the experiment’s configuration variables (global shell variables) to their default values. For many of these variables, the valid values are defined in ‘{HOMErrfs}/ush/valid_param_vals.sh’ (or Section 2.5.4). Since this file contains the default settings, you do **NOT** need to change it. The user-specified experimental parameters will be replaced by running the user-specified experimental configuration file ‘config.[option].sh’.

1. Experimental mode:

No.	Variable	Description
1	RUN_ENVIR	Set to “nco” if running in NCO’s production environment. Valid values are “nco” and “community”.

Table 2.5: Configuration - experimental mode

2. Machine and queue parameters:

No.	Variable	Description
1	MACHINE	HPC machine on which the workflow runs
2	ACCOUNT	Account name to submit jobs to the queue on HPC

3	SCHED	Job scheduler (e.g. 'slurm')
4	QUEUE_DEFAULT	Default queue to which workflow tasks are submitted
5	QUEUE_DEFAULT_TAG	Rocoto XML tag for the default queue. For most platforms this should be "queue".
6	QUEUE_HPSS	Queue for the external model files such as IC and LBC
7	QUEUE_HPSS_TAG	Rocoto XML tag for HPSS queues: "partition" for slurm-based platforms, or "queue" for others.
8	QUEUE_FCST	Queue for a forecast run
9	QUEUE_FCST_TAG	Rocoto XML tag for the 'fcst' queue: "queue" for most platform

Table 2.6: Configuration - machine and queue

3. Cron-associated parameters:

No.	Variable	Description
1	USE_CRON_TO_RELAUNCH	Flag to add a line to the user's cron table to call the experiment launch script every 'CRON_RELAUNCH_INTVL_MNTS' minutes
2	CRON_RELAUNCH_INTVL_MNTS	Interval (in minutes) between successive calls of the experiment launch script by a cron job

Table 2.7: Configuration - cron

4. Experiment directories:

No.	Variable	Description
1	EXPT_BASEDIR	Path and name of the base directory where the experiment is created
2	EXPT_SUBDIR	Name of the subdirectory in the base directory

Table 2.8: Configuration - Experiment directories

5. Variables only for the NCO mode (when RUN_ENVIR="nco"):

No.	Variable	Description
1	COMINgfs	Path to the parent directory of the GFS input files for IC/LBC

2	STMP	Path to the parent directory cycle-dependent model input files, symlinks to cycle-independent input files, and raw forecast output files
3	NET	Model name (first level of com directory structure)
4	envir	"test" for initial testing phase, "para" for run in parallel, "prod" in production
5	RUN	Name of model run (third level of com directory structure)
6	PTMP	Path to the parent directory of post-processed output files

Table 2.9: Configuration - NCO mode

6. Separator character:

No.	Variable	Description
1	DOT_OR_USCORE	Separator (character) used for the names of the gird, mosaic, and orography fixed files

Table 2.10: Configuration - Separator

7. File names:

No.	Variable	Description
1	EXPT_CONFIG_FN	Name of the user-specified configuration file for the forecast experiment ("**config.sh**")
2	RGNL_GRID_NML_FN	Name of the file containing the namelist settings for the ESG (JP) grid
3	DATA_TABLE_FN	Empty file
4	DIAG_TABLE_FN	Name of the file specifying the output fields of the forecast model
5	FIELD_TABLE_FN	Name of the file specifying the tracers of the forecast model
6	FV3_NML_BASE_FN	Name of the Fortran namelist file for *FV3*
7	FV3_NML_YAML_CONFIG_FN	Name of the YAML configuration file containing the forecast model's namelist settings for various physics suits
8	MODEL_CONFIG_FN	Name of the file containing settings and configurations for the NUOPC/ESMF main component
9	NEMS_CONFIG_FN	Name of the NEMS configuration file

10	FV3_EXEC_FN	Name of the *FV3* executable when it is copied from the directory where it is created in the build step to the executable directory ('EXECDIR')
11	WFLOW_XML_FN	Name of the rocoto workflow XML file
12	GLOBAL_VAR_DEFNS_FN	Name of the sell script containing the definitions of the primary experiment variables defined in this default configuration script and in the user-specified configuration as well as the secondary experiment variables generated by the experiment generation script
13	EXTRN_MDL_ICS_VAR_ DEFNS_FN	Name of the shell script containing the definitions of variables associated with the external model from which ICs are generated
14	EXTRN_MDL_LBCS_VAR_ DEFNS_FN	Name of the shell script containing the definitions of variables associated with the external model from which LBCs are generated
15	WFLOW_LAUNCH_ SCRIPT_FN	Name of the script to launch the experiment's rocoto workflow
16	WFLOW_LAUNCH_LOG_FN	Name of the log file that contains the output from successive calls to the workflow launch script

Table 2.11: Configuration - input files

8. Forecast parameters:

No.	Variable	Description
1	DATE_FIRST_CYCL	Starting date of the first forecast (format: YYYYMMDD)
2	DATE_LAST_CYCL	Starting date of the last forecast (format: YYYYMMDD)
3	CYCL_HRS	An array containing the hours of the day at which to launch forecasts. Each element of this array must be a two-digit string less than 24.
4	FCST_LEN_HRS	Length of each forecast (in integer hours)

Table 2.12: Configuration - forecast parameters

9. Initial and lateral boundary condition generation parameters:

No.	Variable	Description
1	EXTRN_MDL_NAME_ICS	Name of the external model that provides fields for initial conditions (“GSMGFS”/“FV3GFS”/“RAPX”/“HRRRX”)
2	EXTRN_MDL_NAME_LBCS	Name of the external model that provides fields for lateral boundary conditions
3	LBC_SPEC_INTVL_HRS	Interval of the LBC files (in integer hours)
4	FV3GFS_FILE_FMT_ICS	Format of the model files for IC (“nemsio” or “grib2”)
5	FV3GFS_FILE_FMT_LBCS	Format of the model files for LBC (“nemsio” or “grib2”)

Table 2.13: Configuration - IC and LBC

10. CCPP-associated parameters:

No.	Variable	Description
1	USE_CCPP	Flag for a CCPP-enabled version of the forecast model
2	CCPP_PHYS_SUITE	If ‘USE_CCPP=“TRUE”, this variable defines the physics suite for CCPP. The choice of physics suite determines the forecast model’s namelist file, diagnostics table file, field table file, and XML physics suite definition file. The valid options are described in ‘{HOMErrfs}/ush/valid)_param_vals.sh’.
3	OZONE_PARAM_NO_CCPP	Ozone parameterization in case of ‘USE_CCPP’=“FALSE”

Table 2.14: Configuration - CCPP parameters

11. Grid-generation method:

No.	Variable	Description
1	GRID_GEN_METHOD	Method of generating a regional grid (“GFDLgrid” or “JPgrid”)

Table 2.15: Configuration - grid generation method

12. Parameters for the GFDL grid refinement:

No.	Variable	Description

1	GFDLgrid_LON_T6_CTR	Longitude of the center of Tile 6 of the parent global domain (in degrees)
2	GFDLgrid_LAT_T6_CTR	Latitude of the center of Tile 6 of the parent global domain (in degrees)
3	GFDLgrid_RES	Total number of grid cells in either one of the longitudinal (x) and latitudinal (y) directions on the **parent** global cubed-sphere grid (Tile 6)
4	GFDLgrid_STRETCH_FAC	Schmidt stretching factor: shrink (≥ 1) or expand ($0 \leq [] \leq 1$)
5	GFDLgrid_REFINE_RATIO	Grid refinement ratio
6	GFDLgrid_ISTART_OF_ RGNL_DOM_ON_T6G	i-index on Tile 6 at which the regional grid (Tile 7) starts
7	GFDLgrid_IEND_OF_ RGNL_DOM_ON_T6G	i-index on Tile 6 at which the regional grid (Tile 7) ends
8	GFDLgrid_JSTART_OF_ RGNL_DOM_ON_T6G	j-index on Tile 6 at which the regional grid (Tile 7) starts
9	GFDLgrid_JEND_OF_ RGNL_DOM_ON_T6G	j-index on Tile 6 at which the regional grid (Tile 7) ends
10	GFDLgrid_USE_GFDLgrid_ RES_IN_FILENAMES	Flag of tile naming convention for grid, orography, and surface climatology files. These files usually start with the string 'C{RES}_'. • "TRUE": {RES} will be set to 'GFDLgrid_RES' just as in the global forecast model. • "FALSE": we calculate (in the grid generation task) an 'equivalent global uniform cubed-sphere resolution', called 'RES_EQUIV', and then set {RES} equal to it. 'RES_EQUIV' is the number of grid points on each tile that a global 'UNIFORM' (stretch factor of 1) cubed-sphere grid would have to have in order to have the same average grid size as the regional grid.

Table 2.16: Configuration - GFDL grid refinement

Notes)

- The regional grid is defined with respect to a 'parent' global cubed-sphere grid. Thus,

all the parameters for a global cubed-sphere grid must be specified in order to define this parent global grid even though the model equations are not integrated on (they are integrated only on the regional grid).

- The above parameters can be easily obtained from the supporting tool *fv3grid* described in Section 10.1.
- The above parameters for the pre-defined grids are specified in '{HOMErrfs}/ush/set_predef_grid_params.sh'.
- Setting 'GFDLgrid_USE_GFDLgrid_RES_IN_FILENAMES'="FALSE" is a more useful indicator of the grid size because it takes into account the effects of 'GFDLgrid_RES', 'GFDLgrid_STRETCH_FAC', and 'GFDLgrid_REFINE_RATIO' in determining the regional grid's typical grid size, whereas simply setting {RES} to 'GFDLgrid_RES' doesn't take into account the effects of 'GFDLgrid_STRETCH_FAC' and 'GFDLgrid_REFINE_RATIO' on the regional grid-resolution. Nevertheless, some users still prefer to use 'GFDLgrid_RES' in the file names, so we allow for that here by setting this flag to "TRUE".

13. Parameters for the ESG (JP) grid:

No.	Variable	Description
1	JPgrid_LON_CTR	Longitude of the center of the grid (in degrees)
2	JPgrid_LAT_CTR	Latitude of the center of the grid (in degrees)
3	JPgrid_DELX	Cell size in the zonal direction of the regional grid (in meters)
4	JPgrid_DELY	Cell size in the meridional direction of the regional grid (in meters)
5	JPgrid_NX	Total number of cells in the zonal direction on the regional grid
6	JPgrid_NY	Total number of cells in the meridional direction on the regional grid
7	JPgrid_WIDE_HALO_WIDTH	Width of the halo to add around the regional grid before shaving the halo down to the width(s) expeted by the forecast model (unit: number of grid cells)
8	JPgrid_ALPHA_PARAM	Alpha parameter in the ESG (JP) grid generation method
9	JPgrid_KAPPA_PARAM	Kappa parameter in the ESG (JP) grid generation method

Table 2.17: Configuration - ESG (JP) grid

Notes)

- In order to generate grid files containing halos that are 3- and 4-cell wide and orography files with halos that are 0- and 3-cell wide, the grid and orography tasks first create files with halos around the regional domain of width 'JPgrid_WIDE_HALO_WIDTH' cells. These are first stored in files. The files are then read in and shaved down to obtain grid files with 3- and 4-cell-wide halos and orography files with 0- and 3-cell-wide halos. For this reason, we refer to the original halo that then gets shaved down as the 'wide' halo, i.e. because it is wider than the 0-, 3-, and 4-cell-wide halos that we will eventually end up with. You do **NOT** need to make this a user-specified variable, but set it in the function of 'set_gridparams_JPgrid.sh'.
- The above parameters for the pre-defined grids are specified in '{HOMErrfs}/ush/set_predef_grid_params.sh'.

14. Parameters for configurations in 'input.nml' and 'model_configuration':

No.	Variable	Description
1	DT_ATMOS	Time integration step of the main forecast model
2	LAYOUT_X	The number of MPI tasks (processes) used in the zonal direction on the regional grid
3	LAYOUT_Y	The number of MPI tasks (processes) used in the meridional direction on the regional grid
4	BLOCKSIZE	Amount of data passed into the cache at a time. The number of vertical columns per MPI task needs to be divisible by 'BLOCKSIZE'. If the block size is non-uniform, the results may not be identical when compared to an all-uniform block-size configuration at full optimization (because of the peel loops). As long as the block sizes are uniform, the answers are the same, independent of what the block size value is. The GFDL dycore does not use this. the interface (i.e. copying the dycore data into the physics data) happens in 'atmos_model.F90)'. GFDL implemented for IPD and it was inherited for CCPP.
5	QUILTING	Flag for write component writing output files
6	PRINT_ESMF	Flag for extra (debugging) information from ESMF routines, Note that the write component uses ESMF library routines to interpolate from the native forecast model grid to the user-specified output grid.

7	WRTCMP_write_groups	The number of write groups (groups of MPI tasks) in the write component
8	WRTCMP_write_tasks_per_group	The number of MPI tasks allocated for each write group

Table 2.18: Configuration - configuration parameters

Note) The above variables and additional parameters related to the write component for the pre-defined grids are specified in '{HOMErrfs}/ush/set_predef_grid_params.sh'.

15. Other configurations:

No.	Variable	Description
1	PREDEF_GRID_NAME	Pre-defined regional grid, If it is set to a valid grid name, the grid parameters, time step (DT_ATMOS), computational parameters such as 'LAYOUT_X' and 'LAYOUT_Y', and write component parameters are overwritten by predefined values for the specific grid.
2	EMC_GRID_NAME	Parameter that allows EMC to use its original grid names
3	PREEXISTING_DIR_METHOD	Method for dealing with pre-existing directories: • "delete": The pre-existing directory is deleted and a new directory is created. • "rename": The pre-existing directory is renamed and a new directory is created ([original]_oldXXX). • "quit": The pre-existing directory is left unchanged, but the current running script is terminated.
4	VERBOSE	Flag for printing out more informational message of the experiment generation and workflow task scripts

Table 2.19: Configuration: Others

16. Grid, orography, and/or surface climatology file generation:

No.	Variable	Description
1	RUN_TASK_MAKE_GRID	Flag for the grid file generation task: • "TRUE": The grid generation task runs to create new grid files. • "FALSE": Pre-generated grid files in 'GRID_DIR' are used.

2	GRID_DIR	Path to the pre-generated grid files in case of 'RUN_TASK_MAKE_GRID'="FALSE"
3	RUN_TASK_MAKE_OROG	Flag for the orography generation task
4	OROG_DIR	Path to the pre-generated orography files in case of 'RUN_TASK_MAKE_OROG'="FALSE"
5	RUN_TASK_MAKE_SFC_CLIMO	Flag for the surface climatology generation task
6	SFC_CLIMO_DIR	Path to the pre-generated surface climatology files in case of 'RUN_TASK_MAKE_SFC_CLIMO'="FALSE"

Table 2.20: Configuration: grid, orography, and surface climatology

17. Fixed files:

No.	Variable	Description
1	SFC_CLIMO_FIELDS	Names of all the surface climatology fields
2	FNGLAC,$\cdots$,FNMSKH	Names of the global data files. These names also appear directly in the input namelist file ('input.nml') of the forecast model.
3	FIXgsm_FILES_TO_COPY_TO_FIXam	If not running in 'NCO' mode, this array contains the names of the files to copy from the FIXgsm system directory to the FIXam directory under the experiment directory. • The last element has a dummy value. It will get reset by the workflow generation scripts to the name of the ozone production/loss file to copy from FIXgsm. The name of this file depends on the ozone parameterization and then the CCPP physics suite specified for the experiment. These steps are carried out elsewhere (in one of the workflow generation scripts/functions)
4	FV3_NML_VARNAME_TO_FIXam_FILES_MAPPING	Some fixed files in the FIXam directory derived from the corresponding workflow variables containing file names
5	FV3_NML_VARNAME_TO_SFC_CLIMO_FIELD_MAPPING	Surface climatology files (on the native FV3SAR grid) in the FIXsar directory derived from the corresponding surface climatology fields

6	CYCLEDIR_LINKS_TO_FIXam_FILES_MAPPING	Mapping between the symlinks created in each cycle directory and their target files in FIXam

Table 2.21: Configuration: fixed files

18. Names of the various workflow tasks:

No.	Variable	Description
1	MAKE_GRID_TN	Task name for grid generation
2	MAKE_OROG_TN	Task name for orography generation
3	MAKE_SFC_CLIMO_TN	Task name for generating surface climatology
4	GET_EXTRN_ICS_TN	Task name for obtaining external data for IC
5	GET_EXTRN_LBCS_TN	Task name for obtaining external data for LBC
6	MAKE_ICS_TN	Task name for generating IC from the external data
7	MAKE_LBCS_TN	Task name for generating LBC from the external data
8	RUN_FCST_TN	Task name for running the forecast model
9	RUN_POST_TN	Task name for running the post-processing tool

Table 2.22: Configuration: workflow tasks

19. Number of nodes for each task:

No.	Variable	Description
1	NNODES_MAKE_GRID	Number of nodes for grid generation
2	NNODES_MAKE_OROG	Number of nodes for orography generation
3	NNODES_MAKE_SFC_CLIMO	Number of nodes for generating surface climatology
4	NNODES_GET_EXTRN_ICS	Number of nodes for obtaining external data for IC
5	NNODES_GET_EXTRN_LBCS	Number of nodes for obtaining external data for LBC
6	NNODES_MAKE_ICS	Number of nodes for generating IC from the external data
7	NNODES_MAKE_LBCS	Number of nodes for generating LBC from the external data
8	NNODES_RUN_FCST	Number of nodes for running the forecast model
9	NNODES_RUN_POST	Number of nodes for running the post-processing tool

Table 2.23: Configuration: number of nodes

20. Number of MPI processes per node for each task:

No.	Variable	Description
1	PPN_MAKE_GRID	Number of MPI processes for grid generation
2	PPN_MAKE_OROG	Number of MPI processes for orography generation
3	PPN_MAKE_SFC_CLIMO	Number of MPI processes for generating surface climatology
4	PPN_GET_EXTRN_ICS	Number of MPI processes for obtaining external data for IC
5	PPN_GET_EXTRN_LBCS	Number of MPI processes for obtaining external data for LBC
6	PPN_MAKE_ICS	Number of MPI processes for generating IC from the data
7	PPN_MAKE_LBCS	Number of MPI processes for generating LBC from the data
8	PPN_RUN_FCST	Number of MPI processes for running the forecast model
9	PPN_RUN_POST	Number of MPI processes for running the post-processing tool

Table 2.24: Configuration: MPI processes

21. Walltime for each task:

No.	Variable	Description
1	WTIME_MAKE_GRID	Walltime for grid generation
2	WTIME_MAKE_OROG	Walltime for orography generation
3	WTIME_MAKE_SFC_CLIMO	Walltime for generating surface climatology
4	WTIME_GET_EXTRN_ICS	Walltime for obtaining external data for IC
5	WTIME_GET_EXTRN_LBCS	Walltime for obtaining external data for LBC
6	WTIME_MAKE_ICS	Walltime for generating IC from the external data
7	WTIME_MAKE_LBCS	Walltime for generating LBC from the external data
8	WTIME_RUN_FCST	Walltime for running the forecast model
9	WTIME_RUN_POST	Walltime for running the post-processing tool

Table 2.25: Configuration: walltime

2.5.3 User-specified Configuration: 'config.[option].sh'

The variables in this file will replace the corresponding variables in the default settings (Section 2.5.2). You should NOT modify the 'config_default.sh' file but add new values of the variables

that you want to change in the 'config_default.sh' file to this 'config.[option].sh' file. All the variables for configuration are described in detail in Section 2.5.2. There are two sample files in '{HOMErrfs}/ush/': (1) 'config.community.sh' and (2) 'config.nco.sh'.

1. 'config.community.sh' for the 'community' mode:

No.	Variable	Default	New configuration
1	RUN_ENVIR	"nco"	"community"
2	MACHINE	"BIG_COMPUTER"	See Table 2.29
3	ACCOUNT	"project_name"	See Table 2.29
4	QUEUE_DEFAULT	"batch_queue"	See Table 2.29
5	QUEUE_HPSS	"hpss_queue"	See Table 2.29
6	QUEUE_FCST	"production_queue"	See Table 2.29
7	DATE_FIRST_CYCL	"YYYYMMDD"	"20190701"
8	DATE_LAST_CYCL	"YYYYMMDD"	"20190701"
9	CYCL_HRS	("HH1" "HH2")	"00"
10	FCST_LEN_HRS	"24"	"06"
11	USE_CCPP	"FALSE"	"TRUE"
12	CCPP_PHYS_SUITE	"FV3_GSD_V0"	"FV3_GFS_2017_gfdlmp"
13	PREDEF_GRID_NAME	" "	"GSD_HRRR25km"
14	PREEXISTING_DIR_METHOD	"delete"	"rename"

Table 2.26: New configuration for the community mode

Notes)

- There are additional variables, which are not shown in the above table, such as 'GRID_GEN_METHOD', 'VERBOSE', 'QUILTING', and 'EXTRN_MDL_NAME_ICS' in the 'config.community.sh' file. They have the same values as the default values in the 'config_default.sh' file. Any variables, even though they have the same value as the default values, can be added to the 'config.[option].sh' file to make sure that the variables have what we intend in the configuration.
- Path to the data set for IC/LBC ('EXTRN_MDL_FILES_SYSBASEDIR_ICS/LBCS') are set in '{HOMErrfs}/ush/set_extrn_mdl_params.sh'. For example, the 'FV3GFS' data can be found as follows:

No.	Machine	Path to the GFS data for IC/LBC
1	Hera	/scratch1/NCEPDEV/rstprod/com/gfs/prod/
2	Orion	
3	WCOSS	/gpfs/dell1/nco/ops/com/gfs/prod/

Table 2.27: Path to the GFS data

- The available 'GRID_GEN_METHOD' for 'PREDEF_GRID_NAME' can be found in Section 2.5.5.

2. 'config.nco.sh' for the 'nco' mode:

No.	Variable	Default	New configuration
1	MACHINE	"BIG_COMPUTER"	See Table 2.29
2	ACCOUNT	"project_name"	See Table 2.29
3	QUEUE_DEFAULT	"batch_queue"	See Table 2.29
4	QUEUE_HPSS	"hpss_queue"	See Table 2.29
5	QUEUE_FCST	"production_queue"	See Table 2.29
6	COMINgfs	"/base/path/$\cdots$"	"/scratch1/NCEPDEV/$\cdots$"
7	STMP	"/base/path/$\cdots$"	"/scratch2/NCEPDEV/$\cdots$"
8	RUN	"experiment_name"	"an_experiment"
9	PTMP	"/base/path/$\cdots$"	"/scratch2/NCEPDEV/$\cdots$"
10	DATE_FIRST_CYCL	"YYYYMMDD"	"20190901"
11	DATE_LAST_CYCL	"YYYYMMDD"	"20190901"
12	CYCL_HRS	("HH1" "HH2")	"18"
13	FCST_LEN_HRS	"24"	"06"
14	USE_CCPP	"FALSE"	"TRUE"
15	CCPP_PHYS_SUITE	"FV3_GSD_V0"	"FV3_GFS_2017_gfdlmp"
16	GRID_GEN_METHOD	"JPgrid"	"GFDLgrid"
13	EMC_GRID_NAME	" "	"conus_c96"
14	PREEXISTING_DIR_METHOD	"delete"	"rename"

Table 2.28: New configuration for the NCO mode

Notes)

- If running in NCO mode, a valid EMC grid must be specified. Make sure that 'EMC_GRID_NAME' is set to a valid value. Its valid values are shown in Table 2.30.
- In NCO mode, 'RUN_TASK_MAKE_GRID', 'RUN_TASK_MAKE_OROG', and 'RUN_TASK_MAKE_SFC_CLIMO' do not need to be explicitly set to "FALSE" in this configuration file because the experiment generation script will do this (along with printing out an informational message).
- In NCO mode, the user must manually (e.g. after doing the build step) create the symlink '${FIXrrfs}/fix_sar' that points to EMC's FIXsar directory on the machine. For example, on Hera, the symlink's target needs to be '/scratch2/NCEPDEV/fv3-cam/emc.campara/fix_fv3cam/fix_sar'. The experiment generation script will then set FIXsar to 'FIXsar="${FIXrrfs}/fix_sar/${EMC_GRID_NAME}" where 'EMC_GRID_NAME' has the value set above.

Copy '{FIXrrfs}/FIXsar' to '{HOMErrfs}/fix/':

```
cd {HOMErrfs}/fix/
cp -r {FIXrrfs}/fix_sar .
```

where {FIXrrfs} is

- On Hera:

```
{FIXrrfs} = /scratch2/NCEPDEV/fv3-cam/emc.campara/fix_fv3cam
```

Mosaic halo files ('mosaic.halo[3,4].nc'):

If the mosaic halo files named 'mosaic.halo[3,4].nc' do not exist in the 'fix_sar' directory, you need to create them as shown in Section 3.3.2.

Link the 'halo4' files to the designated file names

```
cd {HOMErrfs}/fix/fix_sar/
ln -sfn C{res}_[fix field name].tile7.halo4.nc C{res}_[fix field name].
    tile7.nc
```

```
ln -sfn C{res}_[fix field name].tile1.halo4.nc C{res}_[fix field name].
    tile1.nc
```

3. Machine and Queue parameters:

No.	Parameter	Machine		
		Hera	Orion	WCOSS
1	MACHINE	“hera”		“wcoss_cray” / “wcoss_dell_p3”
2	ACCOUNT	“fv3-cam”		
2	QUEUE_DEFAULT	“batch”		“dev”
3	QUEUE_HPSS	“service”		“dev_transfer”
4	QUEUE_FCST	“batch”		“dev”

Table 2.29: Machine-specific queue parameters

2.5.4 Valid Values for Configuration Parameters: ‘valid_param_vals.sh’

The valid values for the experiment’s configuration variables:

No.	Variable (parameter)	Valid values
1	RUN_ENVIR	“nco”, “community”
2	VERBOSE	“TRUE”, “YES”, “FALSE”, “NO”
3	MACHINE	“WCOSS_CRAY”, “WCOSS_DELL_P3”, “THEIA”, “HERA”, “JET”, “ODIN”, “CHEYENNE”
4	SCHED	“slurm”, “pbspro”, “lsf”, “lsfcray”, “none”
5	PREDEF_GRID_NAME	“EMC_CONUS_3km”, “EMC_CONUS_coarse”, “EMC_AK”, “EMC_HI”, “EMC_PR”, “EMC_GU”, “GSD_HAFSV0.A3km”, “GSD_HAFSV0.A13km”, “GSD_HAFSV0.A25km”, “GSD_RRFSAK_3km”, “GSD_HRRR3km”, “GSD_HRRR13km”, “GSD_HRRR25km”, “GSD_RAP13km”
6	EMC_GRID_NAME	“ak”, “conus”, “conus_c96”, “conus_orig”, “guam”, “hi”, “pr”
7	USE_CCPP	“TRUE”, “YES”, “FALSE”, “NO”

8	CCPP_PHYS_SUITE	“FV3_GFS_2017_gfdlmp”, “FV3_GFS_2017_gfdlmp_regional”, “FV3_GSD_v0”, “FV3_GSD_SAR”, “FV3_CPT_v0”, “FV3_GFS_v15p2”, “FV3_GFS_v16beta”, “FV3_GSD_SAR_v1”
9	OZONE_PARAM_NO_CCPP	“ozphys_2015” “ozphys”
10	GFDLgrid_RES	“48”, “96”, “192”, “384”, “768”, “1152”, “3072”
11	EXTRN_MDL_NAME_ICS	“GSMGFS”, “FV3GFS”, “RAPX”, “HRRRX”
12	EXTRN_MDL_NAME_LBCS	“GSMGFS”, “FV3GFS”, “RAPX”, “HRRRX”
13	FV3GFS_FILE_FMT_ICS	“nemsio”, “grib2”
14	FV3GFS_FILE_FMT_LBCS	“nemsio”, “grib2”
15	GIRD_GEN_METHOD	“GFDLgrid”, “JPgrid”
16	PREEXISTING_DIR_METHOD	“delete”, “rename”, “quit”
17	GTYPE	“regional”
18	WRTCMP_output_grid	“rotated_latlon”, “lambert_conformal”, “regional_latlon”
19	RUN_TASK_MAKE_GRID	“TRUE”, “YES”, “FALSE”, “NO”
20	RUN_TASK_MAKE_OROG	“TRUE”, “YES”, “FALSE”, “NO”
21	RUN_TASK_MAKE_SFC_CLIMO	“TRUE”, “YES”, “FALSE”, “NO”
22	QUILTING	“TRUE”, “YES”, “FALSE”, “NO”
23	PRINT_ESMF	“TRUE”, “YES”, “FALSE”, “NO”
24	USE_CRON_TO_RELAUNCH	“TRUE”, “YES”, “FALSE”, “NO”
25	DOT_OR_USCORE	“.”, “_”

Table 2.30: Valid values for the configuration parameters

2.5.5 Pre-defined Grid Parameters : ‘set_predef_grid_params.sh’

This file defines grid parameters for the specific pre-defined grids. The list of the pre-defined grids can be found in Table 2.30 (‘PREDEF_GRID_NAME’). The grid parameters, defined in this file, are as follows:

1. Pre-defined grids:

No.	Grid name	Description	Method(s)
1	GSD_HAFSV0.A3km	HAFS v0.A grid at 3km	JPgrid

2	GSD_HAFSV0.A13km	HAFS v0.A grid at 13*km*	JPgrid
3	GSD_HAFSV0.A25km	HAFS v0.A grid at 25*km*	JPgrid
4	GSD_RAP13km	GSD's RAP grid	GFDLgrid / JPgrid
5	GSD_HRRR25km	GSD's 25*km* CONUS domain	JPgrid
6	GSD_HRRR13km	GSD's 13*km* CONUS domain	JPgrid
7	GSD_HRRR3km	GSD's 3*km* CONUS domain	GFDLgrid / JPgrid
8	EMC_CONUS_3km	EMC's 3*km* CONUS grid	GFDLgrid
9	EMC_CONUS_coarse	EMC's 25*km* CONUS grid	GFDLgrid
10	EMC_AK	EMC's 3*km* Alaska grid	GFDLgrid / JPgrid
11	GSD_RRFSAK_3km	GSD's 3*km* HRRR Alaska grid	GFDLgrid / JPgrid
12	GSD_HRRR_AK_50km	GSD's 50*km* HRRR Alaska grid	JPgrid
13	EMC_HI	EMC's 3*km* Hawaii grid	JPgrid
14	EMC_PR	EMC's 3*km* Puerto Rico grid	JPgrid
15	EMC_GU	EMC's 3*km* Guam grid	JPgrid

Table 2.31: Pre-defined grids

Note) In NCO mode, 'PREDEF_GRID_NAME' is set by 'EMC_GRID_NAME' in 'setup.sh' as follows:

No.	EMC_GRID_NAME	PREDEF_GRID_NAME
1	ak	EMC_AK
2	conus	EMC_CONUS_3km
3	conus_c96	EMC_CONUS_coarse
4	conus_orig	Not set up yet
5	guam	Not set up yet
6	hi	Not set up yet
7	pr	Not set up yet

Table 2.32: Pre-defined grid name based on 'EMC_GRID_NAME'

2. Parameters for the GFDL grid refinement (Table 2.16) or ESG (JP) grid (Table 2.17).

3. Parameters for configurations in 'input.nml' and 'model_configuration' (Table 2.18).

4. Parameters for the write components in case of 'QUILTING'="TRUE":

   (a) 'regional_latlon' or 'rotated_latlon':

No.	Parameter	Description
1	WRTCMP_cen_lon	Central longitude of the output domain (in degrees)
2	WRTCMP_cen_lat	Central latitude of the output domain (in degrees)
3	WRTCMP_lon_lwr_left	Longitudinal distance from the center to the bottom-left corner (in degrees)
4	WRTCMP_lat_lwr_left	Latitudinal distance from the center to the bottom-left corner (in degrees)
5	WRTCMP_lon_upr_rght	Longitudinal distance from the center to the top-right corner (in degrees)
6	WRTCMP_lat_upr_rght	Latitudinal distance from the center to the top-right corner (in degrees)
7	WRTCMP_dlon	Longitudinal distance between two adjacent output-grid points (in degrees)
8	WRTCMP_dlat	Latitudinal distance between two adjacent output-grid points (in degrees)

Table 2.33: Parameters for 'regional_latlon' or 'rotated_latlon'

   (b) 'lambert_conformal':

No.	Parameter	Description
1	WRTCMP_cen_lon	Central longitude of the output domain (in degrees)
2	WRTCMP_cen_lat	Central latitude of the output domain (in degrees)
3	WRTCMP_stdlat1	Latitude of the first standard parallel (in degrees)
4	WRTCMP_stdlat2	Latitude of the second standard parallel (in degrees)
5	WRTCMP_nx	Total number of points along x-axis in Lambert conformal (x, y) plane
6	WRTCMP_ny	Total number of points along y-axis in Lambert conformal (x, y) plane
7	WRTCMP_lon_lwr_left	Longitude of grid point at the bottom-left corner of the output domain (in degrees)

8	WRTCMP_lat_lwr_left	Latitude of grid point at the bottom-left corner of the output domain (in degrees)
9	WRTCMP_dx	Grid cell size in *x* direction (in meters)
10	WRTCMP_dy	Grid cell size in *y* direction (in meters)

Table 2.34: Parameters for 'lambert_conformal'

Notes)

- As shown in Figure 2.4, The script 'set_predef_grid_params.sh' is issued later than 'config.sh'. Therefore, if any parameters are set in both scripts, the parameters in 'set_predef_grid_params.sh' will be finally used in the model.
- The templates of the write components for 'regional_latlon', 'regional_latlon', and 'lambert_conformal' can be found in '{HOMErrfs}/ush/templates/'.

2.6 Templates of Input Files

The template files for a regional workflow are located in '{HOMErrfs}/ush/template/':

No.	File name	Description
1	data_table	Cycle-independent file that the forecast model reads in at the start of each forecast. It is an **empty** file. **No** need to change.
2	diag_table_[CCPP]	File specifying the output fields of the forecast model
3	field_table_[CCPP]	Cycle-independent file that reads in at the start of each forecast. It specifies the scalars that the forecast model will advect.
4	input.nml.FV3	Namelist file of *FV3*.
5	model_configure_[CCPP]	Settings and configurations for the NUOPC/ESMF main component
6	nems.configure	NEMS (NOAA Environmental Modeling System) configuration file, **no** need to change because *FV3* runs stand-alone.
7	FV3.input.yml	YAML configuration file containing the forecast model's namelist settings for various physics suites
8	FV3SAR_wflow.xml	Rocoto XML file to run the workflow
9	regional_grid.nml	Namelist settings for the code that generates an EGS (JP) grid.
10	wrtcmp_lambert_conformal	Write component for 'lambert_conformal' in 'model_configure'

11	wrtcmp_regional_latlon	Write component for 'regional_latlon' in 'model_configure'
12	wrtcmp_rotated_latlon	Write component for 'rotated_latlon' in 'model_configure'

Table 2.35: Template files for a regional workflow

2.6.1 Migratory Route of the Input Files in Workflow

Figure 2.5 shows how the case-specific input files in the templates directory are migrated to the experiment directory. The value of 'CCPP_PHYS_SUITE' is specified in the configuration file 'config.sh'. The template input files corresponding to 'CCPP_PHYS_SUITE' will be copied to the experiment directory '{EXPTDIR} by running the script 'generate_FV3SAR_wflow.sh'. While running the task 'RUN_FCST' in the regional workflow as shown in Figure 3.1, the 'field_table', 'nems.configure', and 'input.nml' files, located in {EXPTDIR} are linked to the cycle directory {CYCLE_DIR} and 'diag_table' and 'model_configure' are copied from the templates directory. Finally, these files are updated with the variables specified in 'var_defn.sh'.

2.6.2 data_table

The 'data_table' is empty.

2.6.3 diag_table.[CCPP]

This file specifies the output fields of the forecast model.

1. Header section:
    - The 'Header' section must reside in the first two lines of the file and contain the title and date of the experiment.
    - The title must be a Fortran character string.
    - The base is the reference time used for the time units, and must be greater than or equal to the model start time.
    - The base date consists of six space-separated integers in the format of 'year month day hour minute second'.
2. Output file entries:

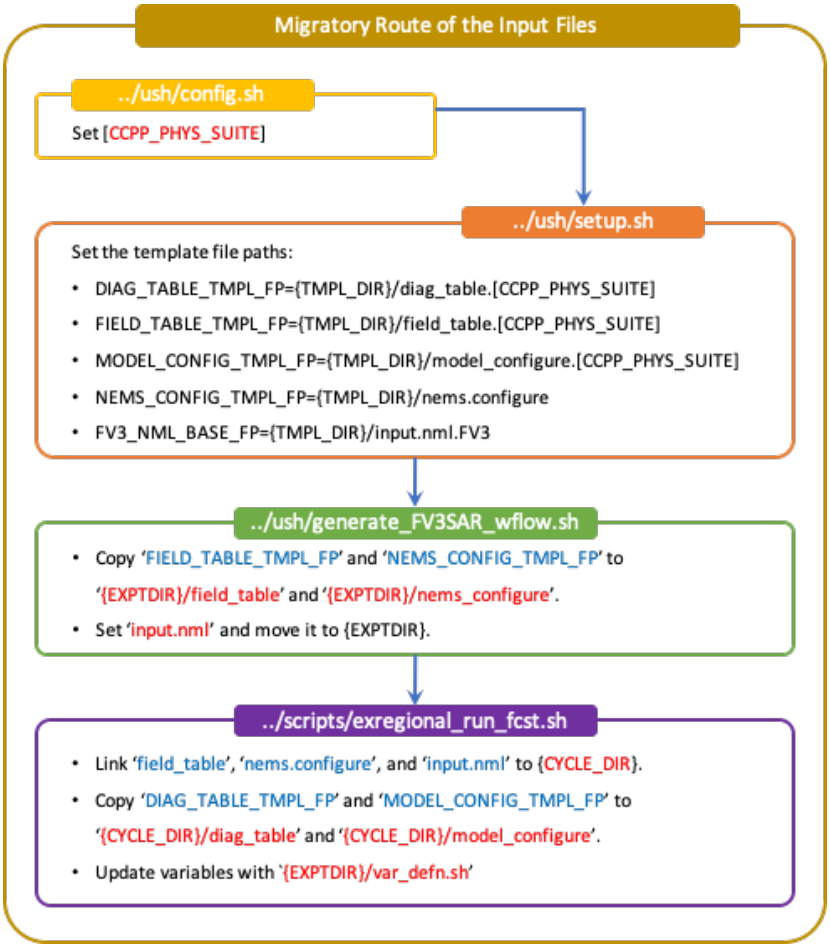

Figure 2.5: Migratory route of the input files

Column	Description	Options
1	File name	"grid_spec", "atmos_static", "fv3_history", "fv3_history2d"
2	Output frequency	• **-1** : only at the end • **0** : every time step • **>0** : designated in column 3
3	Units used for output frequency	"days", "months", "days", "hours", "minutes", "seconds"

4	Format	1 (NetCDF)
5	Units used to label the time axis	“days”, “hours”, “minutes”, “seconds”
6	Long name	“time”

Table 2.36: Output file entries for diagnotics

3. Output format entries:

Column	Description	Options
1	Module name	“dynamics”, “gfs_dyn”, “gfs_phys”, “gfs_sfc”
2	Field name	Described in Table 2.38
3	Output name	Described in Table 2.38
4	File name	File name in Table 2.36 (column 1)
5	Sampling time	“all” (not used)
6	Data reduction method	.false. (not supported)
7	Bounds of the regional section to capture	“none”
8	Packing	Fortran number ‘KIND’ of the data written: • **1** : Double precision • **2** : Float • **4** : Packed 16-bit integer • **8** : Packed 1-bit integer

Table 2.37: Output file entries for diagnotics

- Field and output names in the output files:

No.	Field name	Output name	Description	Output file
1	grid_lon	grid_lon	Longitude of the bottom-left corner (index; top-right on the map) of each grid cell	grid_spec
2	grid_lat	grid_lat	Latitude of the bottom-left corner (index; top-right on the map) of each grid cell	grid_spec
3	grid_lont	grid_lont	Longitude of the center of each grid cell	grid_spec
4	grid_latt	grid_latt	Latitude of the center of each grid cell	grid_spec

5	area	area	Cell area	grid_spec
6	pk	pk	Pressure at the interface 'k' (hybrid coordinate)	atmos_static
7	bk	bk	Vertical coordinate sigma value	atmos_static
8	hyam	hyam	Hybrid coefficient A at the vertical levels	atmos_static
9	hybm	hybm	Hybrid coefficient B at the vertical levels	atmos_static
10	zsurf	zsurf	Surface height (orography)	atmos_static
11	ucomp	ugrd	*u*-comp. of (zonal/longitudinal) wind	fv3_history
12	vcomp	vgrd	*v*-comp. of (meridional/lat.) wind	fv3_history
13	sphum	spfh	Specific humidity	fv3_history
14	temp	tmp	Temperature	fv3_history
15	liq_wat	clwmr	Cloud-water mixing ratio	fv3_history
16	o3mr	o3mr	Ozone mixing ratio	fv3_history
17	delp	dpres		fv3_history
18	delz	delz		fv3_history
19	w	dzdt	Vertical velocity	fv3_history
20	ice_wat	icmr	Cloud-ice mixing ratio	fv3_history
21	rainwat	rwmr	Rain-water mixing ratio	fv3_history
22	snowwat	snmr	Snow-water mixing ratio	fv3_history
23	graupel	grle	Graupel mixing ratio	fv3_history
24	ps	pressfc	Surface air pressure	fv3_history
25	hs	hgtsfc		fv3_history
26	refl_10cm	refl_10cm	Reflectivity from GFDL microphysics	fv3_history
27	wmaxup	upvvelmax		fv3_history
28	wmaxdn	dnvvelmax		fv3_history
29	uhmax03	uhmax03		fv3_history
30	uhmax25	uhmax25		fv3_history
31	uhmin03	uhmin03		fv3_history
32	uhmin25	uhmin25		fv3_history
33	maxvort01	maxvort01		fv3_history
34	maxvort02	maxvort02		fv3_history
35	maxvorthy1	maxvorthy1		fv3_history

36	nwfa	nwfa		fv3_history
37	nifa	nifa		fv3_history
38	cleffr	cleffr		fv3_history
39	cieffr	cieffr		fv3_history
40	cseffr	cseffr		fv3_history
41	QC_BL	qc_bl		fv3_history
42	CLDFRA_BL	cldfra_bl		fv3_history
43	EL_PBL	el_pbl		fv3_history
44	QKE	qke		fv3_history
45	sppt_wts	sppt_wts		fv3_history
46	skebu_wts	skebu_wts		fv3_history
47	diss_est	diss_est		fv3_history
48	shum_wts	shum_wts		fv3_history
49	ALBDO_ave	albdo_ave	Albedo: average	fv3_history2d
50	cnvprcp_ave	cprat_ave	Convective precipitation	fv3_history2d
51	cnvprcpb_ave	cpratb_ave		fv3_history2d
52	totprcp_ave	prate_ave	Total precipitation	fv3_history2d
53	totprcpb_ave	prateb_ave		fv3_history2d
54	DLWRF	dlwrf_ave	Downward long wave rad. flux: avg.	fv3_history2d
55	DLWRFI	dlwrf	Downward long wave rad. flux	fv3_history2d
56	ULWRF	ulwrf_ave	Upward long wave rad. flux: average	fv3_history2d
57	ULWRFI	ulwrf	Upward long wave rad. flux	fv3_history2d
58	DSWRF	dswrf_ave	Downward short wave rad. flux: avg.	fv3_history2d
59	DSWRFI	dswrf	Downward short wave rad. flux	fv3_history2d
60	USWRF	uswrf_ave	Upward short wave rad. flux: average	fv3_history2d
61	USWRFI	uswrf	Upward short wave rad. flux	fv3_history2d
62	DSWRFtoa	dswrf_avetoa	Downward short wave rad. flux: top of atmosphere	fv3_history2d
63	USWRFtoa	uswrf_avetoa	Upward short wave rad. flux: top of atmosphere	fv3_history2d
64	gflux_ave	gflux_ave	Ground heat flux: average	fv3_history2d
65	hpbl	hpbl	Planetary boundary layer height	fv3_history2d

66	lhtfl_ave	lhtfl_ave	Latent heat net flux: average	fv3_history2d
67	shtfl_ave	shtfl_ave	Sensible heat net flux: average	fv3_history2d
68	pwat	pwatclm	Precipitable water	fv3_history2d
69	soilm	soilm	Soil moisture content	fv3_history2d
70	TCDC_aveclm	tcdc_aveclm	Total cloud cover:	fv3_history2d
71	TCDC_avebndcl	tcdc_avebndcl	Total cloud cover:	fv3_history2d
72	TCDC_avehcl	tcdc_avehcl	Total cloud cover: high cloud layer	fv3_history2d
73	TCDC_avelcl	tcdc_avelcl	Total cloud cover: low cloud layer	fv3_history2d
74	TCDC_avemcl	tcdc_avemcl	Total cloud cover: mid. cloud layer	fv3_history2d
75	TCDCcnvcl	tcdccnvcl	Total cloud cover:	fv3_history2d
76	PREScnvclt	prescnvclt		fv3_history2d
77	PREScnvclb	prescnvclb		fv3_history2d
78	PRES_avehct	pres_avehct	Pressure: high cloud top level	fv3_history2d
79	PRES_avehcb	pres_avehcb	Pressure: high cloud bottom level	fv3_history2d
80	TEMP_avehct	tmp_avehct	Temperature: high cloud top level	fv3_history2d
81	PRES_avemct	pres_avemct	Pressure: middle cloud top level	fv3_history2d
82	PRES_avemcb	pres_avemcb	Pressure: middle cloud bottom level	fv3_history2d
83	TEMP_avemct	tmp_avemct	Temperature: middle cloud top level	fv3_history2d
84	PRES_avelct	pres_avelct	Pressure: low cloud top level	fv3_history2d
85	PRES_avelcb	pres_avelcb	Pressure: low cloud bottom level	fv3_history2d
86	TEMP_avelct	tmp_avelct	Temperature: low cloud top level	fv3_history2d
87	u-gwd_ave	u-gwd_ave	Zonal flux of gravity wave stress	fv3_history2d
88	v-gwd_ave	v-gwd_ave	Meridional flux of gravity wave stress	fv3_history2d
89	dusfc	uflx_ave	u-component of momentum flux	fv3_history2d
90	dvsfc	vflx_ave	v-component of momentum flux	fv3_history2d
91	psurf	pressfc	Surface pressure	fv3_history2d
92	u10m	ugrd10m	u-component of $10m$ wind	fv3_history2d
93	v10m	vgrd10m	v-component of $10m$ wind	fv3_history2d
94	crain	crain	Categorical rain (yes=1, no=0)	fv3_history2d
95	tprcp	tprcp	Precipitation rate	fv3_history2d
96	hgtsfc	orog	Surface altitude	fv3_history2d
97	weasd	weasd	Snow-liquid equivalent	fv3_history2d

98	f10m	f10m	10m wind speed over lowest value	fv3_history2d
99	q2m	spfh2m	2 meter specific humidity	fv3_history2d
100	t2m	tmp2m	2 meter temperature	fv3_history2d
101	tsfc	tmpsfc	Sea surface temperature	fv3_history2d
102	vtype	vtype	Vegetation type	fv3_history2d
103	stype	sotyp	Soil type	fv3_history2d
104	slmsksfc	land	Land cover (land=1, sea=0)	fv3_history2d
105	vgracsfc	veg	Vegetation (%)	fv3_history2d
106	zorlsfc	sfcr	Roughness length	fv3_history2d
107	uustar	fricv	Friction velocity	fv3_history2d
108	soilt1	soilt1	Soil column temperature 1	fv3_history2d
109	soilt2	soilt2	Soil column temperature 2	fv3_history2d
110	soilt3	soilt3	Soil column temperature 3	fv3_history2d
111	soilt4	soilt4	Soil column temperature 4	fv3_history2d
112	soilw1	soilw1	Total volumetric soil moisture 1	fv3_history2d
113	soilw2	soilw2	Total volumetric soil moisture 2	fv3_history2d
114	soilw3	soilw3	Total volumetric soil moisture 3	fv3_history2d
115	soilw4	soilw4	Total volumetric soil moisture 4	fv3_history2d
116	slc_1	soill1	Vol. frac. of unfrozen soil moisture 1	fv3_history2d
117	slc_2	soill2	Vol. frac. of unfrozen soil moisture 2	fv3_history2d
118	slc_3	soill3	Vol. frac. of unfrozen soil moisture 3	fv3_history2d
119	slc_4	soill4	Vol. frac. of unfrozen soil moisture 4	fv3_history2d
120	slope	sltyp	Surface slope type	fv3_history2d
121	alnsf	alnsf	Near-IR black sky albedo at zenith 60 degree	fv3_history2d
122	alnwf	alnwf	Near-IR white sky albedo	fv3_history2d
123	alvsf	alvsf	Visible black sky albedo at zenith 60 degree	fv3_history2d
124	alvwf	alvwf	Visible white sky albedo	fv3_history2d
125	canopy	cnwat	Canopy moisture content	fv3_history2d
126	facsf	facsf	Fractional coverage with strong cosz dependency	fv3_history2d

127	facwf	facwf	Fractional coverage with weak cosz dependency	fv3_history2d
128	ffhh	ffhh	Surface exchange coefficient for heat	fv3_history2d
129	ffmm	ffmm	Surface exchange coeff. for momentum	fv3_history2d
130	fice	icec	Sea-ice fraction	fv3_history2d
131	hice	icetk	Sea-ice depth	fv3_history2d
132	snoalb	snoalb	Max. snow albedo over land in fraction	fv3_history2d
133	shdmax	shdmax	Maximum areal fractional coverage of annual green vegetation	fv3_history2d
134	shdmin	shdmin	Minimum areal fractional coverage of annual green vegetation	fv3_history2d
135	snowd	snod	Physical snow depth	fv3_history2d
136	tg3	tg3	Deep soil temperature	fv3_history2d
137	tisfc	tisfc	Sea-ice surface temperature	fv3_history2d
138	tref	tref	Reference temperature	fv3_history2d
139	z_c	zc	Sub-layer cooling thickness	fv3_history2d
140	c_0	c0	Coefficient 1 for d(tz)/d(ts)	fv3_history2d
141	c_d	cd	Coefficient 2 for d(tz)/d(ts)	fv3_history2d
142	w_0	w0	Coefficient 3 for d(tz)/d(ts)	fv3_history2d
143	w_d	wd	Coefficient 4 for d(tz)/d(ts)	fv3_history2d
144	xt	xt	Heat content in DTL	fv3_history2d
145	xz	xz	DTL thickness	fv3_history2d
146	dt_cool	dtcool	Sub-layer cooling thickness	fv3_history2d
147	xs	xs	Salinity content in DTL	fv3_history2d
148	xu	xu	*u*-current content in DTL	fv3_history2d
149	xv	xv	*v*-current content in DTL	fv3_history2d
150	xtts	xtts	d(xt)/d(ts)	fv3_history2d
151	xzts	xzts	d(xz)/d(ts)	fv3_history2d
152	d_conv	dconv	Thickness of free convective layer	fv3_history2d
153	qrain	qrain	Sensible heat flux due to rainfall	fv3_history2d
154	acond	acond	Aerodynamic conductance	fv3_history2d
155	cduvb_ave	cduvb_ave	Clear sky UV-B downward solar flux	fv3_history2d

156	cpofp	cpofp	Percent of frozen precipitation	fv3_history2d
157	duvb_ave	duvb_ave	UV-B downward solar flux	fv3_history2d
158	csdlf_ave	csdlf	Clear sky downward long wave flux	fv3_history2d
159	csusf_ave	csusf	Clear sky upward solar flux	fv3_history2d
160	csusf_avetoa	csusftoa	Clear sky upward solar flux: top of atmosphere	fv3_history2d
161	csdsf_ave	csdsf	Clear sky downward solar flux	fv3_history2d
162	csulf_ave	csulf	Clear sky upward long wave flux	fv3_history2d
163	csulf_avetoa	csulftoa	Clear sky upward long wave flux: top of atmosphere	fv3_history2d
164	cwork_ave	cwork_aveclm	Cloud work function	fv3_history2d
165	evbs_ave	evbs_ave	Direct evaporation from bare soil	fv3_history2d
166	evcw_ave	evcw_ave	Canopy water evaporation	fv3_history2d
167	fldcp	fldcp	Field capacity fraction	fv3_history2d
168	hgt_hyblev1	hgt_hyblev1	Height of hybrid level 1	fv3_history2d
169	spfh_hyblev1	spfh_hyblev1	Specific humidity of hybrid level 1	fv3_history2d
170	ugrd_hyblev1	ugrd_hyblev1	*u*-comp. of current of hybrid level 1	fv3_history2d
171	vgrd_hyblev1	vgrd_hyblev1	*v*-comp. of current of hybrid level 1	fv3_history2d
172	tmp_hyblev1	tmp_hyblev1	Temperature of hybrid level 1	fv3_history2d
173	gfluxi	gflux	Ground heat flux	fv3_history2d
174	lhtfl	lhtfl	Latent heat net flux	fv3_history2d
175	shtfl	shtfl	Sensible heat net flux	fv3_history2d
176	pevpr	pevpr	Potential evaporation rate	fv3_history2d
177	pevpr_ave	pevpr_ave	Potential evaporation rate: average	fv3_history2d
178	sbsno_ave	sbsno_ave	Sublimation (evaporation from snow)	fv3_history2d
179	sfexc	sfexc	Exchange coefficient	fv3_history2d
180	snohf	snohf	Snow phase-change heat flux	fv3_history2d
181	snowc_ave	snowc_ave	Snow cover: average	fv3_history2d
182	spfhmax2m	spfhmax_max2m		fv3_history2d
183	spfhmin2m	spfhmin_min2m		fv3_history2d
184	tmpmax2m	tmax_max2m		fv3_history2d
185	tmpmin2m	tmin_min2m		fv3_history2d

186	ssrun_acc	ssrun_acc	Storm surface runoff:	fv3_history2d
187	sunsd_acc	sunsd_acc	Sunshine duration:	fv3_history2d
188	watr_acc	watr_acc	Water runoff:	fv3_history2d
189	wilt	wilt	Wilting point fraction	fv3_history2d
190	vbdsf_ave	vbdsf_ave	Visible beam downward solar flux	fv3_history2d
191	vddsf_ave	vddsf_ave	Visible diffuse downward solar flux	fv3_history2d
192	nbdsf_ave	nbdsf_ave	Near IR beam downward solar flux	fv3_history2d
193	nddsf_ave	nddsf_ave	Near IR diffuse downward solar flux	fv3_history2d
194	trans_ave	trans_ave	Transpiration	fv3_history2d
195	nwfa2d	nwfa2d		fv3_history2d
196	nifa2d	nifa2d		fv3_history2d
197	maxmf	maxmf		fv3_history2d
198	zol	zol		fv3_history2d
199	flhc	flhc		fv3_history2d
200	flqc	flqc		fv3_history2d

Table 2.38: Field and output name in the output format entries

2.6.4 field_table.[CCPP]

The FMS field and tracer managers manage the tracers in 'field_table'.

No.	Type	Model	Variable	Long name	Unit	Profile
1	TRACER	atmos_mod	sphum	Specific humidity	kg/kg	fixed
2	TRACER	atmos_mod	liq_wat	Cloud-water mixing ratio	kg/kg	fixed
3	TRACER	atmos_mod	ice_wat	Cloud-ice mixing ratio	kg/kg	fixed
4	TRACER	atmos_mod	rainwat	Rain-water mixing ratio	kg/kg	fixed
5	TRACER	atmos_mod	snowwat	Snow-water mixing ratio	kg/kg	fixed
6	TRACER	atmos_mod	graupel	Graupel mixing ratio	kg/kg	fixed
7	TRACER	atmos_mod	water_nc	Cloud liquid water number concentration	$/kg$	fixed
8	TRACER	atmos_mod	ice_nc	Cloud ice water number concentration	$/kg$	fixed
9	TRACER	atmos_mod	rain_nc	Rain number concentration	$/kg$	fixed
10	TRACER	atmos_mod	o3mr	Ozone mixing ratio	kg/kg	fixed

11	TRACER	atmos_mod	liq_aero	Water-friendly aerosol number concentration	/kg	fixed
12	TRACER	atmos_mod	ice_aero	Ice-friendly aerosol number concentration	/kg	fixed
13	TRACER	atmos_mod	sgs_tke	Subgrid scale turbulent kinetic energy	m2/s2	fixed

Table 2.39: Tracers

where the options for 'Profile' are listed as follows:

No.	Method	Method control
1	fixed	surface_value=A
2	profile	surface_value=A, top_value=B (exponential profile)

Table 2.40: Options for 'profile_type'

2.6.5 input.nml.FV3

1. Namelist of *FV3* Dycore ('&fv_core_nml'):

   Reference: UFS FV3DYCORE documentation ('noaa-emc.github.io/FV3_Dycore_ufs-v1.00/html/index.html')

   (a) Layout

No.	Variable	Description	Data type
1	layout	Processor layout on each tile ($N_{\text{PE}} = N_1 \times N_2 \times$ ntiles)	Integers (N_1, N_2)
2	npx	Number of grid **corners** in the x-direction on one time of the domain ($= N_{\text{grid cells}} + 1$)	Integer
3	npy	Number of grid **corners** in the y-direction on one time of the domain	Integer
4	npz	Number of vertical levels	Integer
5	ntimes	Number of tiles on the domain (global=6, regional=1)	Integer
6	blocksize	Number of columns in each 'block' sent to the physics. OpenMP threading is done over the number of blocks. For best performance this number should divide the number of grid cells per processor (npx-1)(npy-1)/($N_1 \times N_2$).	Integer

Table 2.41: Input fields: *FV3* dycore

Note) 'npx' and 'npy' can be obtained by running the python script 'plot_fv3sar_grid_oro.py' in Section 10.2.

(b) Initialization

No.	Variable	Description	Data type
1	external_ic	Initialize the model using the data in the externally specified file in 'res_latlon_dynamics'	Logical
2	ncep_ic	Whether or not the file in 'res_latlon_lynamics' is an NCEP analysis or reanalysis file	Logical
3	nggps_ic	Read initial conditions from horizontally interpolated output from *chgres*	Logical
4	res_latlon_dynamics	File name of the input IC file	Character
5	warm_start	Whether to start from restart files instead of cold starting	Logical

Table 2.42: Input fields: initialization

(c) I/O and diagnostics

No.	Variable	Description	Data type
1	agrid_vel_rst	Whether to write the unstaggered latitude-longitude winds to the restart files	Logical
2	io_layout	Layout of output files on each tile. • 1,1 : Combines all restart and history files on a tile into one file • 0,0 : Every process writes out its own restart and history files	Integers (N_1, N_2)
3	print_freq	Number of hours between print-out of max/min and air/-tracer mass diagnostics to standard output (0=no output, -1=every 'dt_atmos')	Integer

Table 2.43: Input fields: I/O

(d) Time steps

No.	Variable	Description	Data type
1	k_split	Number of vertical remappings per 'dt_atmos' (physical time step)	Integer
2	n_split	Number of small dynamics (acoustic) time steps between vertical remapping	Integer
3	q_split	Number of time steps for sub-cycled tracer advection	Integer

Table 2.44: Input fields: time steps

(e) Grid options

No.	Variable	Description	Data type
1	do_schmidt	Enable grid stretching and rotation using 'stretch_fac', 'target_lat', and 'target_lon'	Logical
2	stretch_fac	Stretching factor for the Schmidt transformation	Real
3	target_lat	Latitude to which the center of tile 6 will be rotated	In degrees
4	target_lon	Longitude to which the center of tile 6 will be rotated	In degrees
5	regional	Flag for a regional domain	Logical
6	refinement	Refinement ratio of the nested grid (default=3)	Integer
7	bc_update_interval	Default setting for interval (hours) between external regional BC data files	Integer
8	regional_bcs_from_gsi	Flag for blending boundary rows in the DA restart files	Logical
9	write_restart_with_bcs	Flag for remapping in the vertical and rotating wind to the grid orientation in DA	Logical
10	nrows_blend	Number of rows for blending	Integer

Table 2.45: Input fields: grid options

2. Namelist of Physics-related options in CCPP ('&gfs_physics_nml'):

   Reference) CCPP scientific documentation ('https://dtcenter.org/GMTB/v4.0/sci_doc/ CCPP-suite_nml_desp.html')

No.	Variable	Description	Default
1	fhcyc	Frequency for surface data cycling in hours (should be **0.0**)	0.0

Table 2.46: Namelist of CCPP

Note) The value of 'fhcyc' should be set to zero, otherwise some of the surface fields will be replaced with climatology data at the frequency.

3. Namelist of FIX files ('&namsfc'):

   Check that the file names of static fields match the files in the 'FIX' directory.

No.	Variable	File name	Directory
1	FNALBC2	C{res}.facsf.tileX.nc	fix_sar
2	FNALBC	C{res}.snowfree_albedo.tileX.nc	fix_sar
3	FNTG3C	C{res}.substrate_temperature.tileX.nc	fix_sar
4	FNVEGC	C{res}.vegetation_greenness.tileX.nc	fix_sar
5	FNSOTC	C{res}.soil_type.tileX.nc	fix_sar
6	FNVMNC	C{res}.vegetation_greenness.tileX.nc	fix_sar
7	FNVMXC	C{res}.vegetation_greenness.tileX.nc	fix_sar
8	FNSLPC	C{res}.slope_type.tileX.nc	fix_sar
9	FNABSC	C{res}.maximum_snow_albedo.tileX.nc	fix_sar
10	FNGLAC	global_glacier.2x2.grb	fix_am
11	FNMXIC	global_maxice.2x2.grb	fix_am
12	FNTSFC	RTGSST.1982.2012.monthly.clim.grb	fix_am
13	FNSNOC	global_snoclim.1.875.grb	fix_am
14	FNZORC	igbp	fix_am
15	FNAISC	CFSR.SEAICE.1982.2012.monthly.clim.grb	fix_am
16	FNSMCC	global_soilmgldas.t1534.3072.1536.grb	fix_am
17	FNMSKH	seaice_newland.grb	fix_am

Table 2.47: Input fields: FIX

Note) These files can be plotted by the post-processing tools shown in Section 10.2.

2.6.6 model_configure.[CCPP]

This file contains settings and configurations for the NUOPC/ESMF main component, including the simulation start time, the processor layout/configuration, and the I/O selections.

No.	Variable name	Meaning	Type	Default
1	print_esmf	Flag for ESMF PET files	Logical	.true.
2	total_number	Total number of ensemble member	Integer	1
3	PE_MEMBER01	Total number of tasks for ensemble number 1	Integer	150
4	start_year	Start year of simulation	Integer	2019
5	start_month	Start month of simulation	Integer	09
6	start_day	Start day of simulation	Integer	12
7	start_hour	Start hour of simulation	Integer	00
8	start_minute	Start minute of simulation	Integer	00
9	start_second	Start second of simulation	Integer	00
10	nhours_fcst	Total forecast length	Integer	48
11	RUN_CONTINUE	Flag for more than one NEMS run	Logical	.false.
12	ENS_SPS	Flag for the ensemble stochastic coupling	Logical	.false.
13	dt_atmos	Atmosphere time step in second	Integer	1800
14	cpl	Flag for coupling with MOM/CICE	Logical	.false.
15	calendar	Type of calendar year	character	‘gregorian’
16	memuse_verbose	Flag for printing out memory usage	Logical	.false.
17	atmos_nthreads	Number of threads for atmosphere	Integer	4
18	use_hyper_thread			
19	ncores_per_node	Number of cores per node	Integer	
20	debug_affinity			
21	restart_interval	Frequency to output restart file	Integer	0
22	quilting	Flag to use the writing component for output (in ‘filename_base’)	Logical	.true.
23	write_groups	Total number of writing groups (groups of MPI tasks) used in the writing component	Integer	2

24	write_tasks_per_group	Total number of tasks in each writing group	Integer	6
25	num_files	Total number of history files	Integer	2
26	filename_base	Name base for history files	Character	‘atm’ ‘sfc’
27	output_file	Format of history files (‘netcdf’)	Character	‘nemsio’
28	write_nemsioflip	Flipping vertical levels to nemsio	Logical	.true.
29	write_fsyncflag	Checking if a file is synced to disk	Logical	.true.
30	output_grid	Grid type of output files	Character	‘gaussian_grid’
31	cen_lon	Central longitude of the output domain	In degrees	
32	cen_lat	Central latitude of the output domain	In degrees	
33	lon1	Longitudinal distance from the center to the bottom-left corner in the rotated coordinate system (negative)	In degrees	
34	lat1	Latitudinal distance from the center to the bottom-left corner in the rotated coordinate system (negative)	In degrees	
35	lon2	Longitudinal distance from the center to the top-right corner (same as ‘lon1’ but positive)	In degrees	
36	lat2	Latitudinal distance form the center to the top-right corner (same as ‘lat1’ but positive)	In degrees	
37	dlon	Longitudinal distance between two adjacent output-grid points	In degrees	
38	dlat	Latitudinal distance between two adjacent output-grid points	In degrees	
39	nfhout	Output frequency of history files	Integer	3
40	nfhmax_hf	Forecast length of high-frequency history files	Integer	0 (no output)
41	nfhout_hf	Output frequency of high-frequency history files	Integer	1
42	nsout	Output frequency of number of time step	Integer	-1 (turn off)

Table 2.48: Variables for *FV3* configuration

Note) The parameters of 'cen_lon', 'cen_lat', 'lon1', 'lat1', 'lon2', and 'lat2' can be found by 'fv3grid' as shown in Section 10.1.2.
Note) If 'quilting=.true.', the output history files of 'fv3_history' and 'fv3_history2d' are split into new output files based on the 'filename_base' such as 'atmfXXX.nc' and 'sfcfXXX.nc'.

2.6.7 nems.configure

Various NEMS (NOAA Environmental Modeling System) components and their run-sequence are specified in this file. Since *FV3* runs stand-alone for the regional workflow, this file is shown as follows:

```
EARTH_component_list: ATM
ATM_model:                fv3
runSeq::
        ATM
::
```

2.7 HPC Environment Configuration

No.	Variable name in configuration	Variable name in scripts	Description
1	layout_1	LAYOUT_X	The number of tasks along the longitudinal direction (x)
2	layout_2	LAYOUT_Y	The number of tasks along the latitudinal direction (y)
3	blocksize	BLOCKSIZE	
4	-	nx_per_task	= NX/LAYOUT_X
5	-	ny_per_task	= NY/LAYOUT_Y
6	PE_MEMBER01	PE_MEMBER01	Total number of MPI tasks for the forecast
7	atmos_nthreads	-	The number of processes per node for atmosphere
8	ncores_per_node	NCORES_PER_NODE	Total number of processes per node in use

9	write_groups	WRTCMP_write_groups	Total number of writing groups
10	write_tasks_per_group	WRTCMP_write_tasks_per_group	The number of tasks in each writing group
11	ppn	PPN_[task]	The number of processes per node
12	OMP_NUM_THREADS	-	The number of OpenMP threads (='--cpus-per-task')
			• $(1)\times(2)+(9)\times(10)=(6)$ • $(4)\times(5)=N\times(3)$ • $(11)\times(12)=(8)$ • $(7)=(12)$

Table 2.49: HPC configurations for *FV3*

Chapter 3

Workflow: Forecast

3.1 Structure of the Regional Workflow

The user-defined experimental directory '{EXPTDIR}' is created by running 'generate_FV3SAR_wflow.sh' as follows:

```
cd {HOME}/../expt_dirs/{EXPT_SUBDIR}   (={EXPTDIR})
```

where '{EXPT_SUBDIR}' is specified in the 'config.[option].sh' file, and the '{EXPTDIR}' directory contains:

No.	File name	Description
1	config.sh	User-specified configuration file (Table 2.28)
2	data_table	Cycle-independent input file (empty)
3	field_table	Scalar fields in the forecast model (Section 2.6.4)
4	FV3SAR_wflow.xml	Rocoto XML file to run the workflow
5	input.nml	Namelist file of *FV3* (Section 2.6.5)
6	launch_FV3SAR_wflow.sh	Symlink to the shell script of '{HOMErrfs}/ush/launch_FV3SAR_wflow.sh' that can be used to (re)launch the Rocoto workflow. Each time this script is called, it appends to a log file named 'log.launch_FV3SAR_wflow'.
7	log.generate_FV3SAR_wflow	Log of the output from the experiment generation script ('generate_FV3SAR_wflow.sh')
8	nems.configure	NEMS configuration file (Section 2.6.7)

9	suite_{CCPP}.xml	CCPP suit definition file that the forecast model reads in if using 'CCPP_PHYS'="TRUE"
10	var_defns.sh	Shell script defining the experiment parameters. It contains all of the primary parameters specified in the default and user-specified configuration files plus many secondary parameters that are derived from the primary ones by the experiment generation script. This file is ourced by various other scripts in order to make all the experiment variables available to these scripts.

Table 3.1: User-defined experimental directory

The flowchart of the regional workflow described in 'FV3SAR_wflow.xml' is shown in Figure 3.1. Even though each task calls its own J-job script located in '{HOMErrfs}/jobs/', the main scripts of the J-jobs can be found in '{HOMErrfs}/scripts/'. Table 3.2 describes the tasks in the regional workflow and the sections for the detail.

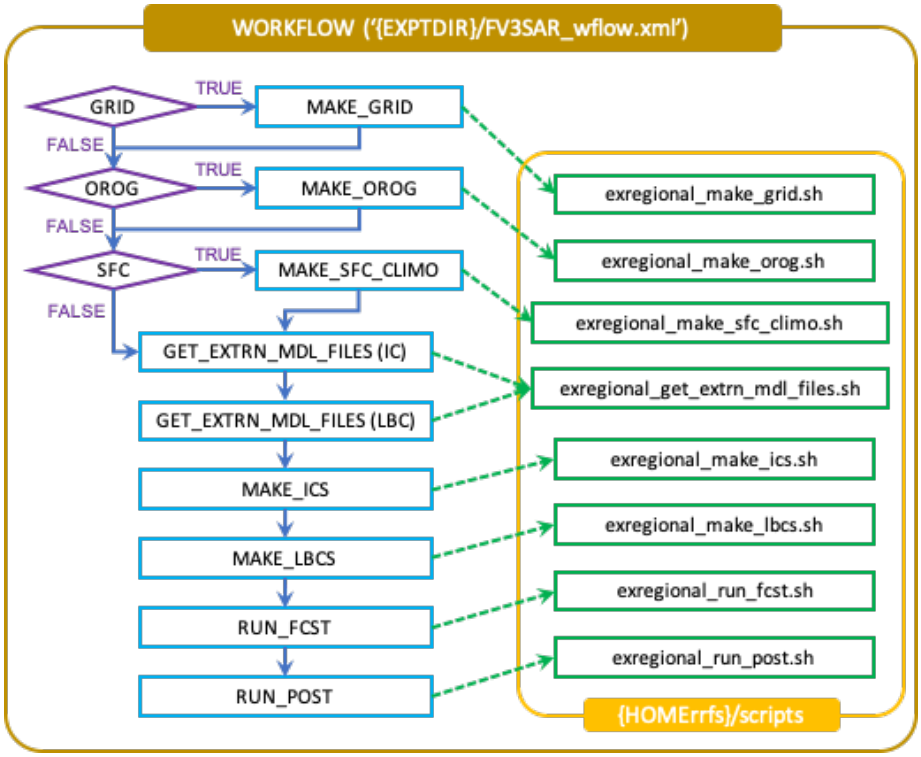

Figure 3.1: Structure of the regional workflow

No.	Workflow task	Description	Detail
1	make_grid	Pre-processing task to generate regional grid files	Section 4.1
2	make_orog	Pre-processing task to generate regional orography files	Section 4.2
3	make_sfc_climo	Pre-processing task to generate surface climatology files	Section 4.3
4	get_extrn_ics	Cycle-specific task to obtain external data for the initial conditions	Section 6.1
5	get_extrn_lbcs	Cycle-specific task to obtain external data for the lateral boundary conditions	Section 6.1
6	make_ics	Generate initial conditions from the external data	Section 6.4
7	make_lbcs	Generate lateral boundary conditions from the external data	Section 6.5
8	run_fcst	Run the forecast model *FV3*	-
9	run_post	Run the post-processing tool *UPP*	Section 8.1

Table 3.2: Workflow tasks

In addition, the 'community' mode creates the 'fix_am' and 'fix_sar' directories in '{EXPTDIR}'. The 'fix_sar' directory is initially empty but will contain some fixed (time-independent) files after the grid, orography, and/or surface climatology generation tasks are run. In the 'nco' mode, make sure that the 'fix_am' and 'fix_sar' directories are located in '{HOMErrfs}/fix/'.

No.	Directory name	Description
1	fix_am	Directory containing the global fixed (time-independent) data files. The experiment generation script copies these files from a machine-dependent system directory.
2	fix_sar	Directory containing the regional fixed (time-independent) data files that describe the regional grid, orography, and various surface climatology fields as well as symlinks to pre-generated files.

Table 3.3: Fix directories

3.2 Launch of Workflow: 'community' Mode

There are two ways to launch the workflow: (1) using the 'launch_FV3SAR_wflow.sh' script, and (2) manually calling the *rocotorun* command as follows:

3.2.1 Launch with the 'launch_FV3SAR_wflow.sh' Script

To launch the 'launch_FV3SAR_wflow.sh' script, simply call it without any arguments.

```
cd {EXPTDIR}
./launch_FV3SAR_wflow.sh
```

This script will create (or append to if it already exists) a log file named 'log.launch_FV3SAR_wflow' in {EXPTDIR}. You can check the contents towards the end of this log file (e.g. the last 30 lines) using the command:

```
tail -n 30 log.launch_FV3SAR_wflow
```

After only one call to 'launch_FV3SAR_wflow.sh', the *tail* command will output something like the following:

```
      CYCLE              TASK                   JOBID          STATE       EXIT STATUS    TRIES    DURATION
==========================================================================================================
202006170000         make_grid    druby://hfe01:33728    SUBMITTING                 -        0         0.0
202006170000         make_orog                      -             -                 -        -           -
202006170000     make_sfc_climo                     -             -                 -        -           -
202006170000      get_extrn_ics    druby://hfe01:33728    SUBMITTING                 -        0         0.0
202006170000     get_extrn_lbcs    druby://hfe01:33728    SUBMITTING                 -        0         0.0
202006170000          make_ics                      -             -                 -        -           -
202006170000         make_lbcs                      -             -                 -        -           -
202006170000          run_fcst                      -             -                 -        -           -
202006170000       run_post_00                      -             -                 -        -           -
202006170000       run_post_01                      -             -                 -        -           -
202006170000       run_post_02                      -             -                 -        -           -
202006170000       run_post_03                      -             -                 -        -           -
202006170000       run_post_04                      -             -                 -        -           -
202006170000       run_post_05                      -             -                 -        -           -
202006170000       run_post_06                      -             -                 -        -           -

Summary of workflow status:
~~~~~~~~~~~~~~~~~~~~~~~~~~

 0 out of 1 cycles completed.
 Workflow status:  IN PROGRESS
```

This shows that the 'make_grid' task (which is cycle-independent and thus run at most once per experiment regardless of the number of cycles specified in 'config.sh') as well as the 'get_extrn_ics' and 'get_extrn_lbcs' tasks (which are cycle-dependent and thus must be run for each cycle; here, they are being run for the first cycle starting on 202006170000) are being submitted to the batch system. The overall status of the workflow is "IN PROGRESS".

Check if any tasks have error messages:

```
cd {EXPTDIR}/log/
```

```
grep -n -i error *.log
```

In order to launch more tasks in the workflow, we must now call the launch script again. We can combine the call to this script with the *tail* command to see output from its log file into one line as follows:

```
cd {EXPTDIR}
./launch_FV3SAR_w ; tail -n 30 log.launch_FV3SAR_wflow
```

The output from this is something like:

```
     CYCLE              TASK                 JOBID          STATE       EXIT STATUS  TRIES  DURATION
==========================================================================================================
202006170000         make_grid               8794245      SUCCEEDED                 -      1       4.0
202006170000         make_orog  druby://hfe01:33728     SUBMITTING                 -      0       0.0
202006170000    make_sfc_climo                     -              -                 -      -         -
202006170000     get_extrn_ics               8794246      SUCCEEDED                 -      1      47.0
202006170000    get_extrn_lbcs               8794246      SUCCEEDED                 -      1      61.0
202006170000          make_ics                     -              -                 -      -         -
202006170000         make_lbcs                     -              -                 -      -         -
202006170000          run_fcst                     -              -                 -      -         -
202006170000       run_post_00                     -              -                 -      -         -
202006170000       run_post_01                     -              -                 -      -         -
202006170000       run_post_02                     -              -                 -      -         -
202006170000       run_post_03                     -              -                 -      -         -
202006170000       run_post_04                     -              -                 -      -         -
202006170000       run_post_05                     -              -                 -      -         -
202006170000       run_post_06                     -              -                 -      -         -

Summary of workflow status:
~~~~~~~~~~~~~~~~~~~~~~~~~~

 0 out of 1 cycles completed.
 Workflow status:  IN PROGRESS
```

This shows that the ‘make_grid’ task as well as ‘get_extrn_ics’ and ‘get_extrn_lbcs’ tasks for the 202006170000 cycle have completed successfully and that the ‘make_orog’ task (which is cycle-independent and thus run at most once per experiment regardless of the number of cycles specified in ‘config.sh’) is being submitted to the batch system.

Once the pre-processing tasks complete without any errors, you should find the files representing completion of the tasks in ‘{EXPTDIR}/log/’ as follows:

No.	Task	File of completion
1	make_grid	make_grid_task_complete.txt
2	make_orog	make_orog_task_complete.txt
3	make_sfc_climo	make_sfc_climo_task_complete.txt

Table 3.4: File names representing completion of the pre-processing tasks

Note) You can call the launch script as often as you like, i.e. you do not have to wait until the currently running tasks are finished. If they are still running, the script will simply append to the log file a new table (which is generated using *Rocoto*'s *rocotostat* command) listing the state of any running tasks as "RUNNING".

Once the 'make_grid', 'make_orog', and 'make_sfc_climo' tasks and the 'get_extrn_ics' and 'get_extrn_lbcs' tasks for the 202006170000 cycle have completed successfully, the new files and sub-directories are as follows:

No.	Directory/file name	Description
1	2020061700	This is the directory created when the first cycle-specific workflow tasks are run, which are 'get_extrn_ics' and 'get_extrn_lbcs' (they are launched simultaneously for each cycle in the experiment). We refer to this as a "cycle directory". Cycle directories are created to contain cycle-specific files for each cycle that the experiment runs. If 'DATE_FIRST_CYCL' and 'DATE_LAST_CYCL' were different, and/or 'CYCL_HRS' contained more than one element in the 'config.sh' file, then more than one cycle directory would be created under the experiment directory.
2	grid	This is a directory generated by the 'make_grid' task. It contains the grid files for the experiment.
3	log	This is a directory in which the log files generated by the overall workflow as well as its various tasks are stored. Look in these files to trace why a task may have failed.
4	orog	This is a directory generated by the 'make_orog' task. It contains the orography files for the experiment.
5	sfc_climo	This is a directory generated by the 'make_sfc_climo' task. It contains the surface climatology files for the experiment.
6	FV3SAR_wflow.db FV3SAR_wlfow_lock.db	These are database files that are generated with *rocoto* is called (by the launch script) to launch the workflow. There is usually no need for the user to modify these files. If you relaunch the workflow from scratch, **delete** these files and then call the launch script (multiple times, as usual).

7	log.launch_FV3SAR_wflow	This is the log file to which the launch script ('launch_FV3SAR_wflow.sh') appends its output each time it is called. Take a look at the last 30–50 lines of this file to check the status of the workflow.

Table 3.5: New directories and files created by *Rocoto*

Notes) To avoid having to manually call the launch script, the experiment generation script allows the user to automatically place the call to this script in the user's crontab and have *cron* call it with a specified frequency.

If everything goes smoothly, you will eventually get the following workflow status table as follows:

CYCLE	TASK	JOBID	STATE	EXIT STATUS	TRIES	DURATION
202006170000	make_grid	8854765	SUCCEEDED	0	1	6.0
202006170000	make_orog	8854809	SUCCEEDED	0	1	27.0
202006170000	make_sfc_climo	8854849	SUCCEEDED	0	1	36.0
202006170000	get_extrn_ics	8854763	SUCCEEDED	0	1	54.0
202006170000	get_extrn_lbcs	8854764	SUCCEEDED	0	1	61.0
202006170000	make_ics	8854914	SUCCEEDED	0	1	119.0
202006170000	make_lbcs	8854913	SUCCEEDED	0	1	98.0
202006170000	run_fcst	8854992	SUCCEEDED	0	1	655.0
202006170000	run_post_00	8855459	SUCCEEDED	0	1	6.0
202006170000	run_post_01	8855460	SUCCEEDED	0	1	6.0
202006170000	run_post_02	8855461	SUCCEEDED	0	1	6.0
202006170000	run_post_03	8855462	SUCCEEDED	0	1	6.0
202006170000	run_post_04	8855463	SUCCEEDED	0	1	6.0
202006170000	run_post_05	8855464	SUCCEEDED	0	1	6.0
202006170000	run_post_06	8855465	SUCCEEDED	0	1	6.0

If all the tasks completed successfully, the workflow status in this log file will be set to "SUCCESS". If for some reason one or more tasks did not complete successfully, the workflow status will be set to "FAILURE".

```
The end-to-end run of the workflow for the forecast experiment specified
by expt_name has completed with the following workflow status (wflow_-
status):
  expt_name = "test_community_hrrr25"
  wflow_status = "SUCCESS"
```

3.2.2 Launch Manually by Calling the *rocotorun* Command

Load the *Rocoto* module to facilitate execution of the workflow:

```
module load rocoto
```

Launch the workflow as follows:

```
cd {EXPTDIR}
rocotorun -w FV3SAR_wflow.xml -d FV3SAR_wflow.db -v 10
```

where '{EXPTDIR}' is the user-defined experimental directory.

To see the status of the workflow, issue a *rocotostat* command as follows:

```
cd {EXPTDIR}
rocotostat -w FV3SAR_wflow.xml -d FV3SAR_wflow.db -v 10
```

Wait a few seconds and issue a second set of *rocotorun* and *rocotostat* commands as follows:

```
rocotorun -w FV3SAR_wflow.xml -d FV3SAR_wflow.db -v 10
rocotostat -w FV3SAR_wflow.xml -d FV3SAR_wflow.db -v 10
```

Note) Issuing a 2nd (3rd, etc) *rocotostat* command without an intervening *rocotorun* command will not result in an updated workflow status table. It is the *rocotorun* command that updates the workflow database file (in this case 'FV3SAR_wflow.db', located in {EXPTDIR}). The *rocotostat* command just reads the database file and prints the table to the screen. Thus, to see an updated table, you must always first issue a *rocotorun* command and then follow it up with a *rocotostat* command.

3.2.3 Turning On/off the Cycle-independent Workflow Tasks

The first three tasks of the workflow ('make_grid', 'make_orog', and 'make_sfc_climo') generate fixed files that are not cycle dependent. Thus, they do not need to be (and are not) run for each cycle. The following three experiment parameters determine whether or not each of these tasks is run at all in the workflow:

1. RUN_TASK_MAKE_GRID
2. RUN_TASK_MAKE_OROG
3. RUN_TASK_MAKE_SFC_CLIMO

By default, these are set to "TRUE" in '{HOMErrfs}/ush/config_defaults.sh' as follows:

```
RUN_TASK_MAKE_GRID=''TRUE''
RUN_TASK_MAKE_OROG=''TRUE''
RUN_TASK_MAKE_SFC_CLIMO=''TRUE''
```

In this case, these tasks will run **only once** at the beginning of the workflow before any cycle-dependent tasks are launched. They will generate grid, filtered orography, and surface climatology files that are needed by the subsequent cycle-dependent tasks.

If you're running a set of cycles for which you've already generated these files (e.g. from a previous end-to-end run of the workflow), then you can skip these three tasks by setting the above flags to "FALSE" in your 'config.sh' file, i.e.

```
RUN_TASK_MAKE_GRID=''FALSE''
RUN_TASK_MAKE_OROG=''FALSE''
RUN_TASK_MAKE_SFC_CLIMO=''FALSE''
```

In this case, you also need to specify the directories in which the workflow should look for your pre-generated grid, orography, and surface climatology files. These directories are specified in the following three experiment parameters:

1. GRID_DIR
2. OROG_DIR
3. SFC_CLIMO_DIR

Thus, to skip the 'make_grid', 'make_orog', and 'make_sfc_climo' tasks, you would set the above three 'RUN_TASK_MAKE_[task]' flags to "FALSE" and specify the paths to your pre-generated files in your '{HOMErrfs}/ush/config.sh' file as follows:

```
RUN_TASK_MAKE_GRID=''FALSE''
GIRD_DIR=''/path/to/pre-generated/grid/files''
RUN_TASK_MAKE_OROG=''FALSE''
OROG_DIR=''/path/to/pre-generated/orog/files''
RUN_TASK_MAKE_SFC_CLIMO=''FALSE''
SFC_CLIMO_DIR=''/path/to/pre-generated/surface/climo/files''
```

Note) If 'RUN_TASK_MAKE_GRID' is set to "TRUE", the value that 'GRID_DIR' is set to in either 'config_defaults.sh' and/or in 'config.sh' is ignored (and, in fact, 'GRID_DIR' gets reset to the directory in which the newly generated grid files that your workflow will create are placed). The same is true for 'RUN_TASK_MAKE_OROG' and 'OROG_DIR' and for 'RUN_TASK_MAKE_SFC_CLIMO' and 'SFC_CLIMO_DIR'.

You can choose to skip any one or any two of these three tasks instead of all three. For example, you can skip the 'make_grid' and 'make_orog' tasks but run the 'make_sfc_climo' task by:

1. Setting 'RUN_TASK_MAKE_GRID'="FALSE" and 'RUN_TASK_MAKE_OROG'="FALSE" in your 'config.sh'.
2. Setting 'GRID_DIR' and 'OROG_DIR' in your 'config.sh' to the locations in which grid and orography files for the grid you want to run on can be found.
3. Setting 'RUN_TASK_MAKE_SFC_CLIMO'="TRUE" in your 'config.sh' (and not setting 'SFC_CLIMO_DIR' in 'config.sh' since its value will not be used).

3.3 Launch of Workflow: 'nco' Mode

3.3.1 Mosaic Halo Files

Different from the previous EMC's regional workflow, the unified regional workflow in the short range weather application requires 'mosaic_halo3.nc' and 'mosaic_halo4.nc' files as well as 'mosaic_halo0.nc'. Previously, there was just one mosaic file 'C{res}_mosaic.nc' that points to the pre-shave **halo6** file. A mosaic file is used by both the preprocessing codes and by *FV3*. Since halo6 has more layers than halo3, things will not go out of bounds but it might cause some potential issues. Therefore, it is important for each grid file to have its own mosaic file corresponding to the number of halos.

```
# Load the same modules used for 'make_solo_mosaic'.
cd {HOMErrfs}/fix/fix_sar/{domain}/
{EXEC}/make_solo_mosaic --num_tile 1 --dir ''{HOME}/fix/fix_sar'' --tile_file ''C{
    res}_grid.tile7.halo3.nc'' --mosaic ''C{res}_mosaic.halo3''
{EXEC}/make_solo_mosaic --num_tile 1 --dir ''{HOME}/fix/fix_sar'' --tile_file ''C{
    res}_grid.tile7.halo4.nc'' --mosaic ''C{res}_mosaic.halo4''
```

where '{EXEC}' is the directory where the executable 'make_solo_mosaic' is located (typically ='{HOME}/exec/').

Note) You can check which modules were loaded in installing 'make_solo_mosaic' as follows:

```
cd {HOME}/src/logs
vim build_fre-nctools.log
# Check ''Currently Loaded Modules:''
```

- On Hera:

```
module load intel/18.0.5.274
module load impi/2018.0.4
module load netcdf/4.7.0
module load hdf5/1.10.5
```

3.3.2 Launch with the 'launch_FV3SAR_wflow.sh' Script

Launch the 'launch_FV3SAR_wflow.sh' script:

```
cd {EXPTDIR}
./launch_FV3SAR_wflow.sh
```

This script will create (or append to if it already exists) a log file named 'log.launch_FV3SAR_wflow' in {EXPTDIR}. You can check the contents towards the end of this log file (e.g. the last 30 lines) using the command:

```
tail -n 30 log.launch_FV3SAR_wflow
```

After only one call to 'launch_FV3SAR_wflow.sh', the *tail* command will output something like the following:

```
    CYCLE              TASK                   JOBID          STATE       EXIT STATUS  TRIES  DURATION
==========================================================================================================
202006180000     get_extrn_ics  druby://hfe01:33424     SUBMITTING                 -      0       0.0
202006180000     get_extrn_lbcs  druby://hfe01:33424     SUBMITTING                 -      0       0.0
202006180000        make_ics                     -              -                 -      -         -
202006180000        make_lbcs                    -              -                 -      -         -
202006180000          run_fcst                   -              -                 -      -         -
202006180000      run_post_00                    -              -                 -      -         -
202006180000      run_post_01                    -              -                 -      -         -
202006180000      run_post_02                    -              -                 -      -         -
202006180000      run_post_03                    -              -                 -      -         -
202006180000      run_post_04                    -              -                 -      -         -
202006180000      run_post_05                    -              -                 -      -         -
202006180000      run_post_06                    -              -                 -      -         -

Summary of workflow status:
~~~~~~~~~~~~~~~~~~~~~~~~~~

 0 out of 1 cycles completed.
 Workflow status:  IN PROGRESS
```

In order to launch more tasks in the workflow, we must now call the launch script again. We can combine the call to this script with the *tail* command to see output from its log file into one line as follows:

```
cd {EXPTDIR}
./launch_FV3SAR_w ; tail -n 30 log.launch_FV3SAR_wflow
```

The output from this is something like:

CYCLE	TASK	JOBID	STATE	EXIT STATUS	TRIES	DURATION
201909011800	get_extrn_ics	8887541	SUCCEEDED	0	1	4.0
201909011800	get_extrn_lbcs	8887542	SUCCEEDED	0	1	4.0
201909011800	make_ics	druby://hfe11:33475	SUBMITTING	–	0	0.0
201909011800	make_lbcs	druby://hfe11:33475	SUBMITTING	–	0	0.0
201909011800	run_fcst	–	–	–	–	–
201909011800	run_post_00	–	–	–	–	–
201909011800	run_post_01	–	–	–	–	–
201909011800	run_post_02	–	–	–	–	–
201909011800	run_post_03	–	–	–	–	–
201909011800	run_post_04	–	–	–	–	–
201909011800	run_post_05	–	–	–	–	–
201909011800	run_post_06	–	–	–	–	–

If everything goes smoothly, you will eventually get the following workflow status table as follows:

CYCLE	TASK	JOBID	STATE	EXIT STATUS	TRIES	DURATION
201909011800	get_extrn_ics	8887541	SUCCEEDED	0	1	2.0
201909011800	get_extrn_lbcs	8887542	SUCCEEDED	0	1	2.0
201909011800	make_ics	8887551	SUCCEEDED	0	1	160.0
201909011800	make_lbcs	8887552	SUCCEEDED	0	1	148.0
201909011800	run_fcst	8887753	SUCCEEDED	0	1	365.0
201909011800	run_post_00	8888085	SUCCEEDED	0	1	7.0
201909011800	run_post_01	8888086	SUCCEEDED	0	1	7.0
201909011800	run_post_02	8888087	SUCCEEDED	0	1	8.0
201909011800	run_post_03	8888106	SUCCEEDED	0	1	6.0
201909011800	run_post_04	8888107	SUCCEEDED	0	1	7.0
201909011800	run_post_05	8888108	SUCCEEDED	0	1	8.0
201909011800	run_post_06	8888109	SUCCEEDED	0	1	6.0

If all the tasks completed successfully, the workflow status in this log file will be set to “SUCCESS”. If for some reason one or more tasks did not complete successfully, the workflow status will be set to “FAILURE”.

```
The end-to-end run of the workflow for the forecast experiment specified
by expt_name has completed with the following workflow status (wflow_-
status):
  expt_name = "test_nco_conus96"
  wflow_status = "SUCCESS"
```

3.4 Output

3.4.1 Type of Output Files

The output files that contain the physical variables are converted into other formats such as NetCDF and GRIB2 in the regional workflow as shown in Figure 3.2. The file names for the physical variables, 'fv3_history' and 'fv3_history2d', are specified in the input file 'diag_table'. When 'quilting' is set to '.true.' for the write components in the input file 'model_configure', the two files of 'fv3_history' and 'fv3_history2d' are converted into the two NetCDF files named 'dynfXXX.nc' and 'phyfXXX.nc'. The bases of the file names are specified in the input file 'model_configure'. These NetCDF files are converted into the two GRIB2 files named 'BGDAWP' and 'BGRD3D'. Finally, the 'BGDAWP' and 'BGRD3D' files are linked to '[domain].tXXz.bgdawpXX.tmXX' and '[domain].bgrd3dXX.tmXX', respectively.

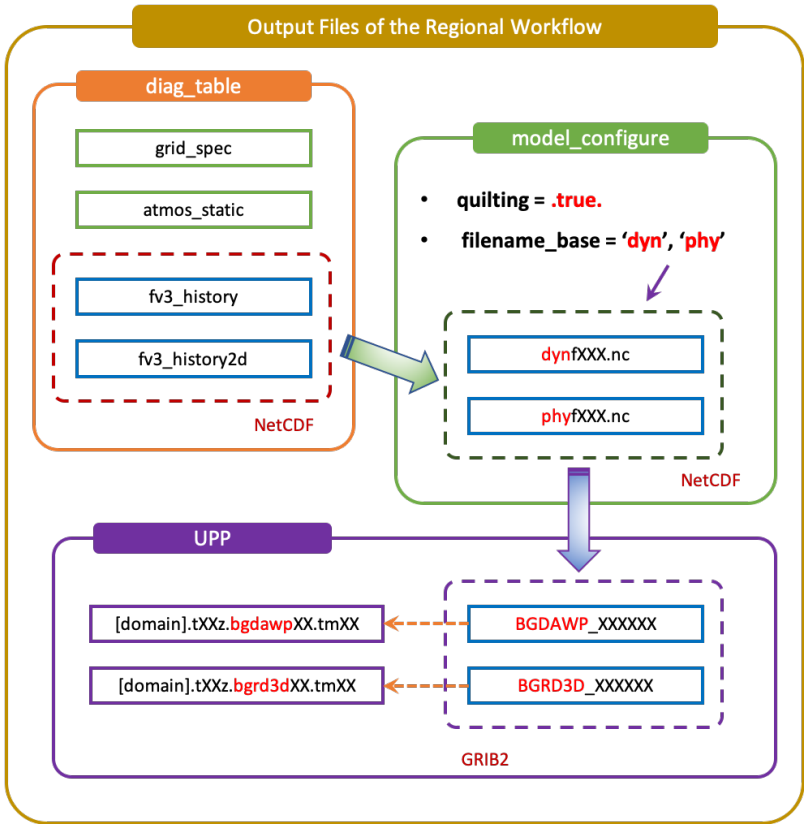

Figure 3.2: Output files of the regional workflow

3.4.2 Output Files: 'community' Mode

1. NetCDF files:

```
cd {EXPTDIR}/YYYYMMDDHH
```

   - dynfXXX.nc (See Section 10.2.14 for plotting)
   - phyfXXX.nc (See Section 10.2.15 for plotting)

2. GRIB2 files:

```
cd {EXPTDIR}/YYYYMMDDHH/postprd
```

   - {domain}.t{cyc}z.bgdawp{fhour}.tmXX (or) BGDAWP_20170000XXX00 (See Section 10.2.17 or Section 3.4.4 for plotting)
   - {domain}.t{cyc}z.bgrd3d{fhour}.tmXX (or) BGRD3D_20170000XXX00 (See Section 10.2.16 for plotting)

3.4.3 Output Files: 'nco' Mode

1. NetCDF files:

```
cd {STMP}/stmp/tmpnwprd/{RUN}/YYYYMMDDHH
```

   - dynfXXX.nc (See Section 10.2.14 for plotting)
   - phyfXXX.nc (See Section 10.2.15 for plotting)

2. GRIB2 files:

```
cd {PTMP}/com/rrfs/para/{RUN}.YYYYMMDD/HH
```

   - {domain}.t{cyc}z.bgdawp{fhour}.tmXX (or) BGDAWP_20170000XXX00 (See Section 10.2.17 or Section 3.4.4 for plotting)
   - {domain}.t{cyc}z.bgrd3d{fhour}.tmXX (or) BGRD3D_20170000XXX00 (See Section 10.2.16 for plotting)

3.4.4 Python Script Provided with Workflow

A python script that can plot the result GRIB2 file, {domain}.t{cyc}z.bgdawp{fhour}.tmXX, is provided with the regional workflow as follows:

```
cd {HOMErrfs}/ush/Python/
```

You can find two files in the above directory as:

1. env.yaml: This is a *yaml* file to define the *Python* virtual environment.
    - 'EXPT_BASEDIR': {HOME}/../expt_dirs
    - 'EXPT_SUBDIR': Name of the user-defined experimental directory
    - 'CARTOPY_DIR': Path to the directory where the Natural Earth shape files are located
2. plot_allvars.py: Python script

To run the python script:

```
cd {HOMErrfs}/ush/Python/
vim env.yaml
# Edit the paths and name
# Load necessary python modules on HPC
python3 plot_allvars.py {YYYYMMDDHH}  {CYC}
```

The python script produces several output files as:

1. 10m wind speed: compare10mwind_{domain}_fXX.png
2. 250mb wind speed: compare250wind_{domain}_fXX.png
3. 2m dew point temperature: compare2mdew_{domain}_fXX.png
4. 2m temperature: compare2mt_{domain}_fXX.png
5. 500mb wind speed: compare500_{domain}_fXX.png
6. Composite reflectivity: comparerefc_{domain}_fXX.png
7. Surface-based CAPE: comparefcape_{domain}_fXX.png
8. Sea Level Pressure: compareslp_{domain}_fXX.png

Chapter 4

UFS_UTILS: Grid and Static Fields with Workflow

4.1 ESG (JP) Grid and GFDL Grid Refinement

4.1.1 Adding a New Pre-defined Grid to Workflow

1. Add the name of a new grid to the list defined by 'valid_vals_PREDEF_GRID_NAME':

```
cd {HOMErrfs}/ush/
vim valid_param_vals.sh
```

2. Add the new grid to the list of grids in the case statement of '{PREDEF_GRID_NAME}':

```
cd {HOMErrfs}/ush/
vim set_predef_grid_params.sh
```

Note) There is an if-statement that checks whether it is a 'GFDLgrid' type or 'JPgrid' (ESG grid) type of grid. If you provide the parameters only for the 'JPgrid' case, you should put in a 'print_err_msg_exit' call for the 'GFDLgrid'.

4.1.2 Flowchart of the Grid-generation Job in Workflow

The flowchart of the grid-generation job in the regional workflow is illustrated in Figure 4.1. As a result, the grid and mosaic files with three different halos of 3, 4, and 6 are generated in '{EXPTDIR}/grid/'. The list of executables used for the grid-generation job is as follows:

1. ‘make_hgrid’
2. ‘regional.grid’
3. ‘global_equiv_resol’
4. ‘shave.x’

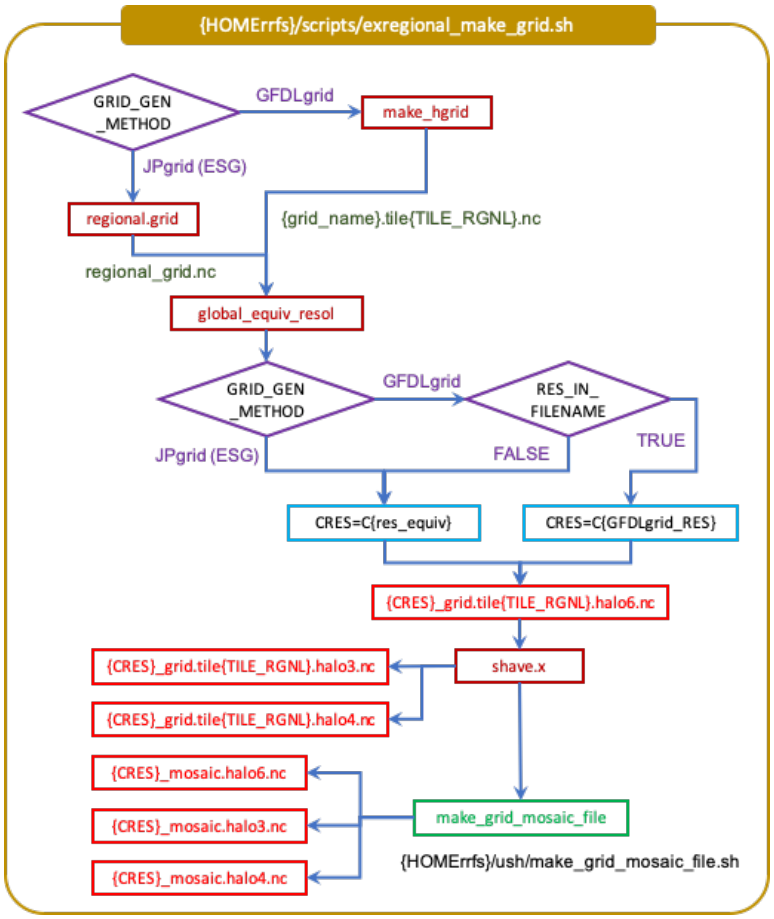

Figure 4.1: Flowchart of ‘exregional_make_grid.sh’

4.1.3 Global Equivalent Resolution

In the above flowchart, the executable ‘global_equiv_resol’ calculates an equivalent global uniform cubed-sphere resolution (unit: number of cells) for the pre-defined regional grid. This is the ‘C{res}’

that a global uniform (i.e. stretch factor of 1) cubed-sphere grid would need to have in order to have the same nominal cell size as the average cell size of the regional grid. This value is required for the topography filtering to work correctly.

$$\text{RES\_equiv} = \text{int}\left(\frac{2.0\times\pi\times R_E}{4.0\times\text{avg\_cell\_size}}\right) \tag{4.1}$$

where R_E is the mean radius of the earth ($\approx 6371km$). Therefore, the 'RES_equiv' physically means the number of the average cells between the equator and the pole along the earth's surface. For example, the global equivalent resolution of a regional grid whose average-cell-size is $2.99km$ is C3343.

4.1.4 Parameters for an ESG (JP) grid

The default values of the parameters related to an ESG (JP) grid are set in the 'config_default.sh' file as shown in Table 2.17. In addition, the grid-specific parameters such as those in Table 2.18 and Table 2.43, and 'write-component' parameters for a new regional grid can be specified in the 'set_predef_grid_params.sh'.

1. Namelist of the ESG (JP) grid generator 'regional_grid' ('regional_grid.nml' in '{HOMErrfs}/scripts/exregional _make_grid.sh'):

No.	Variable	Description
1	plon	Longitude of the center of the regional domain (in degrees)
2	plat	Latitude of the center of the regional domain (in degrees)
3	delx	Grid spacing in the longitudinal direction on the *FV3* super-grid (**in degrees**) $\rightarrow$ delx = $\Delta_x/(2\times6371)\times(180/\pi)$ where (Δ_x : km)
4	dely	Grid spacing in the latitudinal direction on the *FV3* super-grid (**in degrees**) $\rightarrow$ dely = $\Delta_y/(2\times6371)\times(180/\pi)$ where (Δ_y : km)
5	lx	Negative of the number of cells in the longitudinal (x) direction on the super-grid from the lower-left corner to the grid center $\rightarrow$ lx= $-(\text{nx}+2\times N_{\text{halo}})$
6	ly	Negative of the number of cells in the longitudinal (y) direction on the super-grid from the lower-left corner to the grid center $\rightarrow$ ly= $-(\text{ny}+2\times N_{\text{halo}})$

7	a	Parameter α of the generalized gnomonic mapping centered at (plon,plat)
8	k	Parameter κ of the generalized gnomonic mapping centered at (plon,plat)

Table 4.1: Namelist of the the ESG (JP) grid generator.

Notes)

- Since the super-grid has twice the resolution of the actual grid, the grid spacing in the x direction on the actual grid will be twice this resolution.
- The physical grid spacing Δ_x on the actual grid is related to 'delx' by $\Delta_x = 2 \times \text{delx} \times (P_E/360)$ where P_E is the Earth's circumference, and the factor 2 appears due to 'delx' being the grid angle in the x direction of the super-grid. Therefore, delx = $\Delta_x/(2 \times R_E) \times (360/2\pi)$ where R_E is the radius of the Earth ($\approx 6371km$). For example, 'delx=0.0135' for the $3km$ actual grid spacing.
- These parameters can be found by *fv3grid* as shown in Section 10.1.3.

2. Variables in 'set_predef_grid_params.sh' and 'set_gridparams_JPgrid.sh':

No.	Variable in 'set_predef_ grid_params.sh'	Variable in 'set_ gridparams_JPgrid.sh'	Variable in Table 4.1
1	JPgrid_LON_CTR	lon_ctr	plon
2	JPgrid_LAT_CTR	lat_ctr	plat
3	JPgrid_NX	nx	nx
4	JPgrid_NY	ny	ny
5	JPgrid_DELX (in **meters**)	delx (in **meters**)	Δ_x
6	JPgrid_DELY (in **meters**)	dely (in **meters**)	Δ_y
7	JPgrid_WIDE_HALO_WIDTH	halo_width	N_{halo}
8	JPgrid_ALPHA_PARAM	-	a
9	JPgrid_KAPPA_PARAM	-	k
10	-	stretch_factor	-
11	-	del_angle_x_sg	delx
12	-	del_angle_y_sg	dely
13	-	neg_nx_of_dom_with_ wide_halo	lx

14	-	neg_ny_of_dom_with_ wide_halo	ly

Table 4.2: Grid parameters for an ESG (JP) grid.

Notes)

- Make sure that 'JPgrid_DELX' and 'JPgrid_DELY' are in meters because the 'radius_Earth' (R_E in Table 4.1) is set in meters in 'constants.sh'.
- The variable names of 'delx' and 'dely' means totally different parameters between 'set_ gridparams_JPgrid.sh' and the namelist of 'regional_grid'. The grid spacing parameters 'delx' and 'dely' in degrees are calculated in 'set_gridparams_JPgrid.sh' based on the input parameters 'delx' and 'dely' in meters of 'set_predef_grid_params.sh'.
- If any 'JPgrid_XXX' variables are not specified in 'set_predef_grid_params.sh', the default values in 'config_defaults.sh' or the pre-defined values in 'config.sh' will be used.

4.1.5 Parameters for a GFDL grid

The default values of the parameters related to a GFDL grid are set in the 'config_default.sh' file as shown in Table 2.16. In addition, the grid-specific parameters such as those in Table 2.18 and Table 2.43, and 'write-component' parameters can be specified in the 'set_predef_grid_params.sh'.

1. Namelist of the GFDL grid generator 'make_hgrid' (shown in '{HOMErrfs}/scripts/exregional _make_grid.sh'):

No.	Variable name	Description
1	grid_type	Type of topography (='gnomonic_ed': equal distance gnomonic cubic grid)
2	nlon	Number of model grid points (super-grid) for each zonal regions of varying resolution (2×{res})
3	grid_name	Name of output grid (=C{res}_grid)
4	do_schmidt	Set to do Schmidt transformation to create stretched grid
5	stretch_factor	Stretching factor for the grid
6	target_lon	Center longitude of the regional domain (tile 7)

7	target_lat	Center latitude of the regional domain (tile 7)
8	nest_grid	Option for a regional grid
9	parent_tile	Parent tile number of a regional grid (=6)
10	refine_ratio	Grid refinement ratio for a regional grid
11	istart_nest	Starting *i*-directional index of a regional grid on the parent tile super-grid (Fortran index)
12	jstart_nest	Ending *i*-directional index of a regional grid on the parent tile super-grid (Fortran index)
13	iend_nest	Starting *j*-directional index of a regional grid on the parent tile super-grid (Fortran index)
14	jend_nest	Ending *j*-directional index of a regional grid on the parent tile super-grid (Fortran index)
15	halo	Lateral boundary halo size (only for a regional grid)
16	great_circle_ algorithm	When specified, the 'greate_circle_algorithm' will be used to compute grid cell areas

Table 4.3: Namelist of the the GFDL grid generator.

2. Variables in 'set_predef_grid_params.sh' and 'set_gridparams_GFDLgrid.sh':

No.	Variable in 'set_predef_ grid_params.sh'	Variable in 'set_gridparams_ GFDLgrid.sh'	Variable in Table 4.3
1	GFDLgrid_LON_T6_CTR	lon_of_t7_ctr	target_lon
2	GFDLgrid_LAT_T6_CTR	lat_of_t7_ctr	target_lat
3	GFDLgrid_STRETCH_FAC	stretch_factor	stretch_factor
4	GFDLgrid_RES	res_of_t6g	-
5	GFDLgrid_REFINE_RATIO	refine_ratio_t6g_to_t7g	refine_ratio
6	GFDLgrid_ISTART_OF_ RGNL_DOM_ON_T6G	istart_of_t7_on_t6g	-
7	GFDLgrid_IEND_OF_RGNL_ DOM_ON_T6G	iend_of_t7_on_t6g	-
8	GFDLgrid_JSTART_OF_ RGNL_DOM_ON_T6G	jstart_of_t7_on_t6g	-

9	GFDLgrid_JEND_OF_RGNL_DOM_ON_T6G	jend_of_t7_on_t6g	-
10	GFDLgrid_USE_GFDLgrid_RES_IN_FILENAMES	jend_of_t7_on_t6g	-
11	-	nx_of_t7_on_t7g	-
12	-	ny_of_t7_on_t7g	-
13	-	halo_width_on_t7g	-
14	-	istart_of_t7_with_halo_on_t6sg	istart_nest
15	-	iend_of_t7_with_halo_on_t6sg	iend_nest
16	-	jstart_of_t7_with_halo_on_t6sg	jstart_nest
17	-	jend_of_t7_with_halo_on_t6sg	jend_nest

Table 4.4: Grid parameters for a GFDL grid.

Notes)

- The index numbers of 5–9 in Table 4.4 are **NOT** those on the super-grid but they are the numbers on a general grid such as orography. Therefore, if you use 'fv3grid' to set up the size of a new regional domain, you should divide the numbers on 'fv3grid' by 2.
- In the scripts for the GFDL grid-generation, 'GFDLgrid_RES' is used to set the number of cells along the longitudinal/latitudinal direction on a general grid (not super-grid). For example, 'GFDLgrid_RES' = 768 in the 'C768' grid system. The number of cells along the longitudinal (latitudianl) direction on the super-grid is 2×768.

The GFDL gird generation executable 'make_hgrid' requires the index limits such as starting and ending indices of the regional grid on the super-grid of its **parent** tile (Tile 6). The index limits on the super-grid of the parent tile are represented as

```
istart_of_t7_on_t6sg = 2 × istart_of_t7_on_t6g -1
iend_of_t7_on_t6sg = 2 × iend_of_t7_on_t6g
jstart_of_t7_on_t6sg = 2 × jstart_of_t7_on_t6g -1
jend_of_t7_on_t6sg = 2 × jend_of_t7_on_t6g
```

where 't6sg' denotes the super-grid of Tile 6, and 't6g' means the normal grid of Tile 6.

To obtain a regional grid with a halo, the index limits must include the halo number. The starting indices on the super-grid of Tile 6 with wide halo must be odd while the ending indices must be even. We subtract 1 from the starting indices if they are even (which ensures that there will be at least 'halo_width_on_t7g' halo cells along the left and bottom boundaries), and we add 1 to the ending indices if they are odd for the right and top boundaries:

istart_of_t7_with_halo_on_t6sg = istart_of_t7_on_t6g − halo_width_on_t6sg (−1, **if** even)
iend_of_t7_with_halo_on_t6sg = iend_of_t7_on_t6g + halo_width_on_t6sg (+1, **if** odd)
jstart_of_t7_with_halo_on_t6sg = jstart_of_t7_on_t6g − halo_width_on_t6sg (−1, **if** even)
jend_of_t7_with_halo_on_t6sg = jend_of_t7_on_t6g + halo_width_on_t6sg (+1, **if** odd)

where

halo_width_on_t6sg={2×({$NH4 = 4$} + 1)+ refine_ratio_t6g_to_t7g −1}/refine_ratio_t6g_to_t7g

To calculate the number of cells along the longitudinal (x) and latitudinal (y) directions in the regional domain:

nx_of_t7_on_t7g={(iend_of_t7_on_t6sg−istart_of_t7_on_t6sg+1)/2}×refine_ratio_t6g_to_t7g
ny_of_t7_on_t7g={(jend_of_t7_on_t6sg−jstart_of_t7_on_t6sg+1)/2}×refine_ratio_t6g_to_t7g

Note) There are two different ways to create a regional grid as shown in Table 4.5. One is to set the indices of 11 to 14 in Table 4.3 to be the same as the exact regional domain and set 'halo' to 3 as shown in Sections 5.3 and 7.2 ('without workflow'). The other is to include the adjusted halo number ('halo_width_on_t6sg') in the indices of 11 to 14 and set 'halo' to 1 as described in this section ('with workflow'). However, in either case, the three different domains with halo0, halo3, and halo4 will be created by the executable 'shave.x' later.

No.	Namelist Variable	Case 1 (w/o workflow)	Case 2 (w/ workflow)
1	istart_nest	istart_of_t7_on_t6g	istart_of_t7_with_halo_on_t6sg
2	iend_nest	iend_of_t7_on_t6g	iend_of_t7_with_halo_on_t6sg
3	jstart_nest	jstart_of_t7_on_t6g	jstart_of_t7_with_halo_on_t6sg
4	jend_nest	jend_of_t7_on_t6g	jend_of_t7_with_halo_on_t6sg
5	halo	3	1

Table 4.5: Two different approaches to create a GFDL grid.

4.1.6 Running a Workflow for a New Grid with an Example

1. Get the parameters for a new grid from *fv3grid* as describe in Section 10.1:

   - ESG (JP) grid parameters:

     Note) To match the parameters of *fv3grid* with those of the script, make sure that lx × −2 = nx and ly × −2 = ny.

No.	Parameter	Value	Variable in the script
1	plon	-75.0	JPgrid_LON_CTR
2	plat	28.5	JPgrid_LAT_CTR
3	delx	0.0135	(=3000 for 'JPgrid_DELX')
4	dely	0.0135	(=3000 for 'JPgrid_DELX')
5	lx	-1500	-(JPgrid_NX)
6	ly	-1200	-(JPgrid_NY)
7	nx	3000	2×JPgrid_NX
8	ny	2400	2×JPgrid_NY

Table 4.6: Parameters for ESG grid

   - GFDL grid parameters:

     Note)

     – Since we supposed that the center of the regional domain is located at the center of the parent tile (Tile 6), we recommend to make the domain symmetry by setting 'I end'=2×'resolution C'-('I start'-1) and 'J end'=2×'resolution C'-('J start'-1).

     – You can use 'num_margin_cells_T6_[left/right/bottom/top]' to make a symmetry domain along the centerline of the parent tile 6.

     – The starting and ending indices (7–10 in Table 4.7) should be adjusted in 'set_ gridparams_GFDLgrid.sh' to satisfy the HPC environment configuration shown in Section 2.7.

No.	Parameter	Value	Variable in the script
1	resolution C	768	GFDLgrid_RES
2	stretch factor	1.50	GFDLgrid_STRETCH_FAC

3	target lon	-75.0	GFDLgrid_LON_T6_CTR
4	target lat	28.5	GFDLgrid_LAT_T6_CTR
5	parent	6	-
6	refine ratio	3	GFDLgrid_REFINE_RATIO
7	I start	261	≈ 2×GFDLgrid_ISTART_OF_RGNL_DOM_ON_T6G
8	J start	361	≈ 2×GFDLgrid_JSTART_OF_RGNL_DOM_ON_T6G
9	I end	1280	≈ 2×GFDLgrid_IEND_OF_RGNL_DOM_ON_T6G
10	J end	1160	≈ 2×GFDLgrid_JEND_OF_RGNL_DOM_ON_T6G

Table 4.7: Parameters for GFDL grid

- Rotated Lat-Lon (RLL) parameters (for 'QUILTING'='TRUE'):

No.	Parameter	Value	Variable in the script
1	tlm0d	-75.0	WRTCMP_cen_lon
2	tph0d	28.5	WRTCMP_cen_lat
3	wbd	-20.0	WRTCMP_lon_lwr_left
4	sbd	-15.5	WRTCMP_lat_lwr_left

Table 4.8: Parameters for the write component of the GFDL grid

2. Add the name of the new grid to the list defined by 'valid_vals_PREDEF_GRID_NAME':

```
cd {HOMErrfs}/ush/
vim valid_param_vals.sh
(add) ''JCH_ECOAST_3km'' \
```

3. Add the new grid to the list of grids in the case statement of '{PREDEF_GRID_NAME}':

```
cd {HOMErrfs}/ush/
vim set_predef_grid_params.sh
```

Add below:

```
#----------------------------------
# Test grid for East Coast by Chan-Hoo Jeon
#----------------------------------
#
"JCH_ECOAST_3km")
```

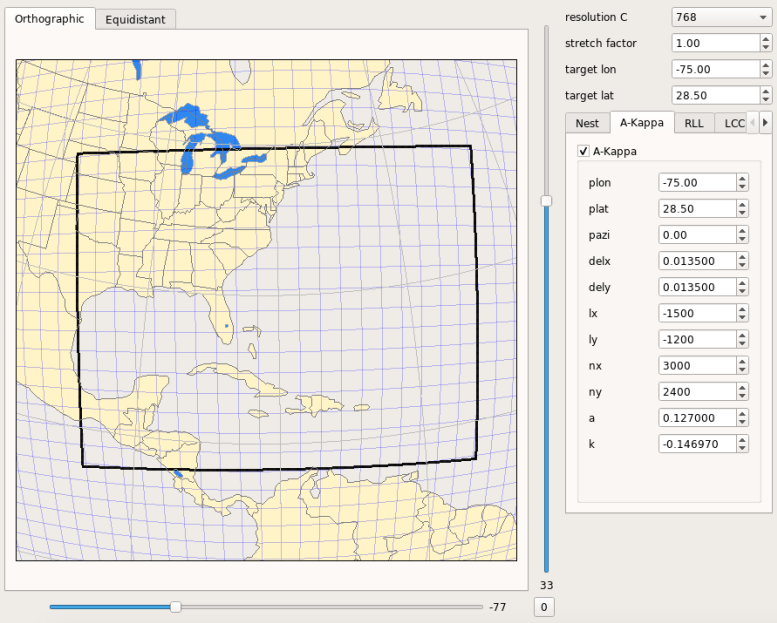

Figure 4.2: ESG grid parameters for the East Coast domain by *fv3grid*

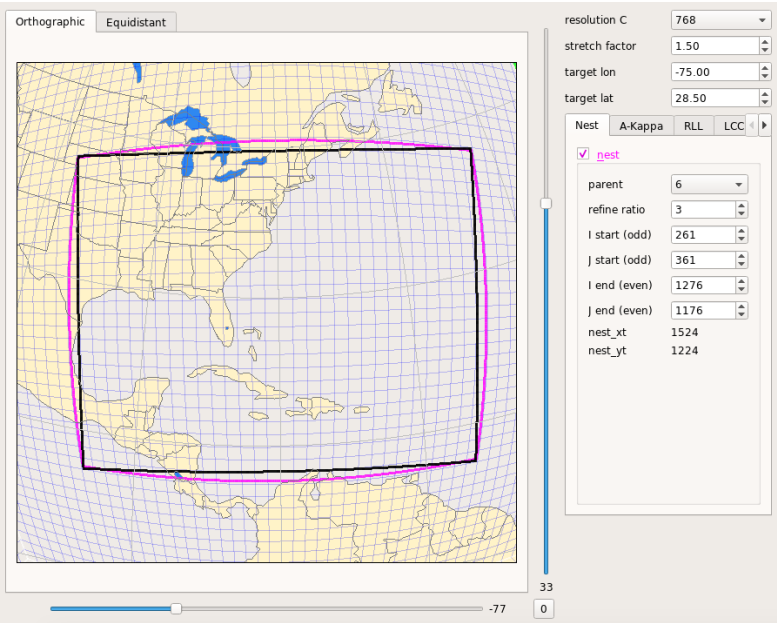

Figure 4.3: GFDL grid parameters for the East Coast domain by *fv3grid*

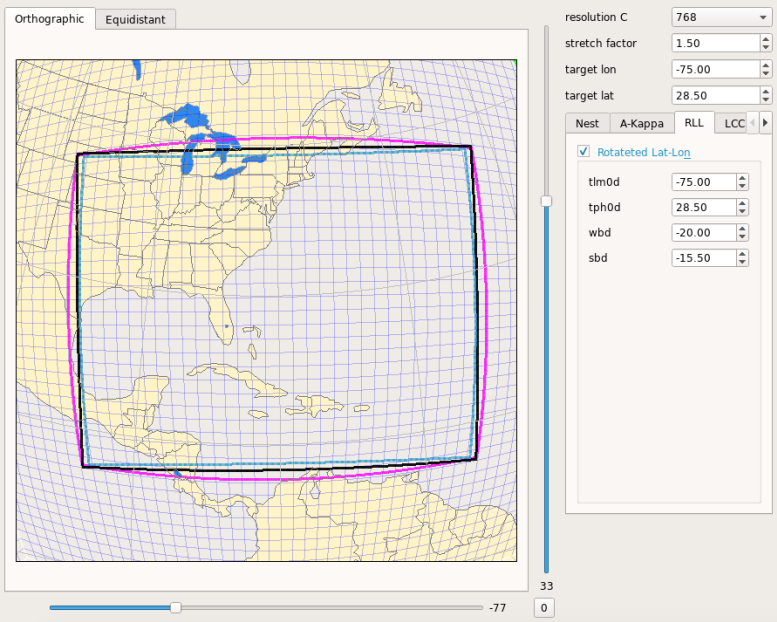

Figure 4.4: Write-component parameters for the East Coast domain by *fv3grid*

```
if [ "${GRID_GEN_METHOD}" = "GFDLgrid" ]; then
  GFDLgrid_LON_T6_CTR=-75.0
  GFDLgrid_LAT_T6_CTR=28.5
  GFDLgrid_STRETCH_FAC=1.5
  GFDLgrid_RES="768"
  GFDLgrid_REFINE_RATIO=3

  num_margin_cells_T6_left=120
  GFDLgrid_ISTART_OF_RGNL_DOM_ON_T6G=$(( num_margin_cells_T6_left + 1 ))
  num_margin_cells_T6_right=120
  GFDLgrid_IEND_OF_RGNL_DOM_ON_T6G=$(( GFDLgrid_RES - num_margin_cells_T6_right ))
  num_margin_cells_T6_bottom=180
  GFDLgrid_JSTART_OF_RGNL_DOM_ON_T6G=$(( num_margin_cells_T6_bottom + 1 ))
  num_margin_cells_T6_top=180
  GFDLgrid_JEND_OF_RGNL_DOM_ON_T6G=$(( GFDLgrid_RES - num_margin_cells_T6_top ))

  GFDLgrid_USE_GFDLgrid_RES_IN_FILENAMES="TRUE"

  DT_ATMOS="18"
  LAYOUT_X="24"
  LAYOUT_Y="24"
  BLOCKSIZE=18

  if [ "$QUILTING" = "TRUE" ]; then
    WRTCMP_write_groups="1"
    WRTCMP_write_tasks_per_group=$(( 1*LAYOUT_Y ))
    WRTCMP_output_grid="rotated_latlon"
    WRTCMP_cen_lon="-75.0"
    WRTCMP_cen_lat="28.5"
    WRTCMP_lon_lwr_left="-20.0"
    WRTCMP_lat_lwr_left="-15.5"
```

```
    WRTCMP_lon_upr_rght="20.0"
    WRTCMP_lat_upr_rght="15.5"
    WRTCMP_dlon="0.02"
    WRTCMP_dlat="0.02"
  fi
elif [ "${GRID_GEN_METHOD}" = "JPgrid" ]; then
  JPgrid_LON_CTR=-75.0
  JPgrid_LAT_CTR=28.5
  JPgrid_DELX="3000.0"
  JPgrid_DELY="3000.0"
  JPgrid_NX=1500
  JPgrid_NY=1200
  JPgrid_WIDE_HALO_WIDTH=6

  DT_ATMOS="18"
  LAYOUT_X="30"
  LAYOUT_Y="40"
  BLOCKSIZE=30

  if [ "$QUILTING" = "TRUE" ]; then
    WRTCMP_write_groups="1"
    WRTCMP_write_tasks_per_group=$(( 1*LAYOUT_Y ))
    WRTCMP_output_grid="rotated_latlon"
    WRTCMP_cen_lon="-75.0"
    WRTCMP_cen_lat="28.5"
    WRTCMP_lon_lwr_left="-20.0"
    WRTCMP_lat_lwr_left="-15.5"
    WRTCMP_lon_upr_rght="20.0"
    WRTCMP_lat_upr_rght="15.5"
    WRTCMP_dlon="0.02"
    WRTCMP_dlat="0.02"
  fi

 fi
 ;;
#
```

4. Edit 'config.community.sh'.

```
cd {HOMErrfs}/ush/
vim config.community.sh
```

- ESG (JP) grid:

```
EXPT_SUBDIR="jch_ecoast_esg"
RUN_ENVIR="community"
PREDEF_GRID_NAME="JCH_ECOAST_3km"
GRID_GEN_METHOD="JPgrid"
RUN_TASK_MAKE_GRID="TRUE"
RUN_TASK_MAKE_OROG="TRUE"
RUN_TASK_MAKE_SFC_CLIMO="TRUE"
```

Note) Unless the pre-generated surface climatology files exist, 'RUN_TASK_MAKE_SFC_CLIMO' should be set to "TRUE".

- GFDL grid:

```
EXPT_SUBDIR="jch_ecoast_gfdl"
RUN_ENVIR="community"
PREDEF_GRID_NAME="JCH_ECOAST_3km"
GRID_GEN_METHOD="GFDLgrid"
RUN_TASK_MAKE_GRID="TRUE"
RUN_TASK_MAKE_OROG="TRUE"
RUN_TASK_MAKE_SFC_CLIMO="TRUE"
```

5. Load the machine-specific modules on HPC:

   - On Hera:

```
module use -a /contrib/miniconda3/modulefiles
module load miniconda3
conda activate regional_workflow
```

6. Generate a workflow:

```
cd {HOMErrfs}/ush/
cp config.community.sh config.sh
./generate_FV3SAR_wflow.sh
```

7. Run the workflow:

```
cd {EXPTDIR}
./launch_FV3SAR_wflow.sh
tail -n 30 log.launch_FV3SAR_wflow
```

8. The grid and mosaic files should be found in:

```
cd {EXPTDIR}/grid/
# or
cd {EXPTDIR}/fix_sar/
```

Note) Once the orography files are obtained from Section 4.2, the grid and orography files can be plotted by the supporting tool shown in Section 10.2.6.

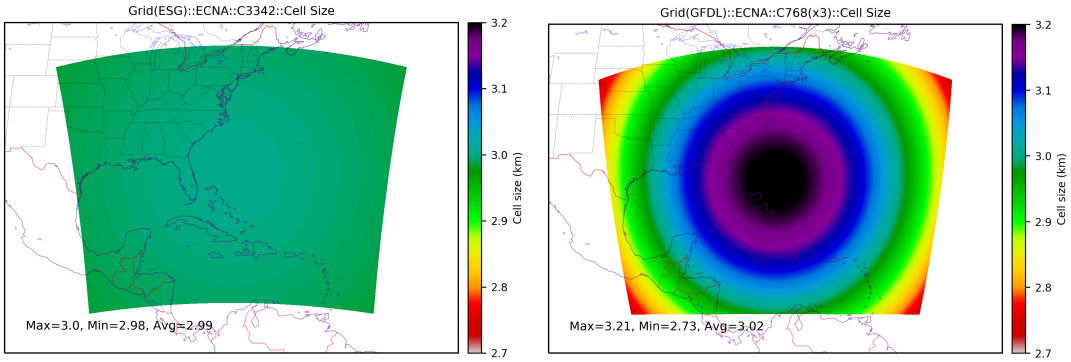

Figure 4.5: Comparison of cell sizes between ESG and GFDL

4.2 Orography

Once the grid-generation task described in Section 4.1 is complete, the orography files can be obtained by running the regional workflow again:

```
cd {EXPTDIR}
./launch_FV3SAR_wflow.sh
tail -n 30 log.launch_FV3SAR_wflow
```

The flowchart of orography generation is shown in Figure 4.6. The list of executables used for the orography-generation job is as follows:

1. orog.x
2. filter_topo
3. shave.x

The input files required for this job are as follows:

1. {CRES}_grid.tile7.halo6.nc
2. {CRES}_mosaic.halo6.nc
3. Topography files (‘{FIXgsm}/../fix_orog/’):
   (a) thirty.second.antarctic.new.bin
   (b) landcover30.fixed

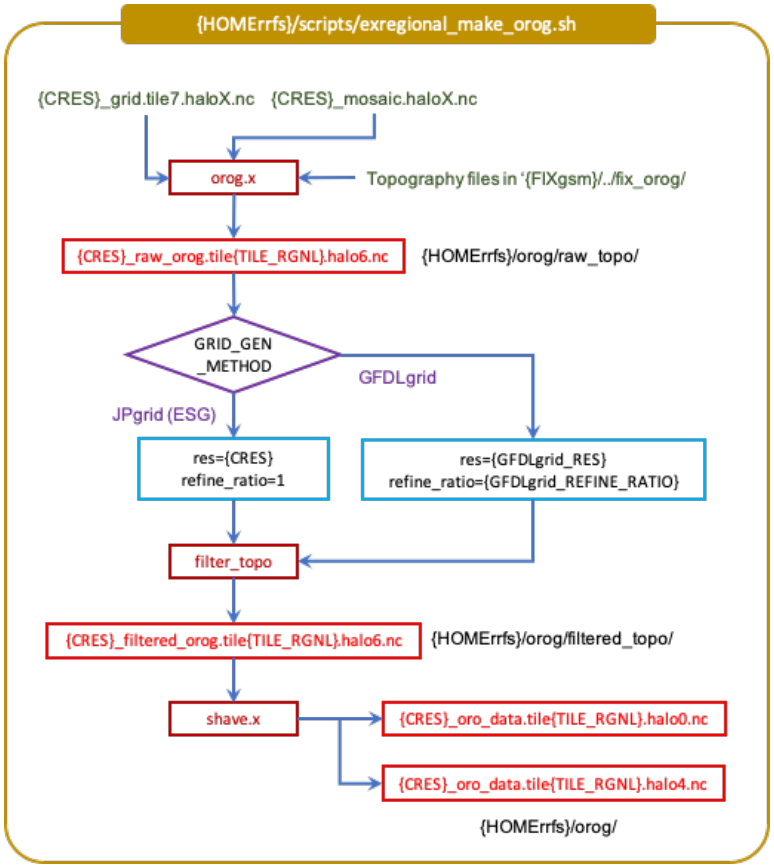

Figure 4.6: Flowchart of 'exregional_make_orog.sh'

(c) gmted2010.30sec.int

The orography files should be found in:

```
cd {EXPTDIR}/orog/
# or
cd {EXPTDIR}/fix_sar/
```

Note) Once the orography files are obtained from Section 4.2, the grid and orography files can be plotted by the supporting tool shown in Section 10.2.6.

4.3 Regional Static (Fix) Fields: Surface Climatology

Once the grid- and orography-generation tasks described in Sections 4.1 and 4.2 are complete, the regional surface climatology files can be obtained by running the regional workflow again:

```
cd {EXPTDIR}
./launch_FV3SAR_wflow.sh
tail -n 30 log.launch_FV3SAR_wflow
```

Namelist of the executable 'sfc_climo_gen':

No.	Variable	File name
1	input_facsf_file	facsf.1.0.nc
2	input_substrate_temperature_file	substrte_temperature.2.6x1.5.nc
3	input_maximum_snow_albedo_file	maximum_snow_albedo.0.05.nc
4	input_snowfree_albedo_file	snowfree_albedo.4comp.0.05.nc
5	input_slope_type_file	slope_type.1.0.nc
6	input_soil_type_file	soil_type.statsgo.0.05.nc
7	input_vegetation_type_file	vegetation_type.igbp.0.05.nc
8	input_vegetation_greenness_file	vegetation_greenness.0.144.nc
9	mosaic_file_mdl	{FIXsar}/{CRES}_mosaic.halo4.nc
10	orog_dir_mdl	{FIXsar}
11	orog_files_mdl	{CRES}_oro_data.tile6.halo4.nc
12	halo	4
13	maximum_son_albedo_method	bilinear
14	snowfree_albedo_method	bilinear
15	vegetation_greenness_method	bilinear

Table 4.9: Namelist of 'sfc_climo_gen'.

The regional surface clomatology files should be found in:

```
cd {EXPTDIR}/sfc_climo/
# or
cd {EXPTDIR}/fix_sar/
```

Note) The surface climatology files can be plotted by the supporting tool shown in Section 10.2.7.

Chapter 5

UFS_UTILS: GFDL Grid and Static Fields without Workflow

5.1 Structure of Pre-processing

The structure of pro-processing for gird, mosaic, orography, and static surface climatology files is shown in Figure 5.1. This process can be achieved by running the script 'driver_grid.[machine].sh' located in the directory of '{TOP_dir}/UFS_UTILS/driver_scripts/'. This script first sets not only grid specifications such as resolution, grid type, stretch factor, refinement ratio, and indexes of the regional domain on the global coordinates, but also paths to the home and working directories. Then it submits another script ('fv3gfs_driver_grid.sh') running the sub-scripts for grid, orography, topography filtering, halo files, and static surface climatology fields one by one.

```
cd {TOP_dir}/UFS_UTILS/ush/
vim fv3gfs_driver_grid.sh
```

where

No.	Variable	Description
1	exec_dir	Directory for executables
2	topo	Name and path of the directory where the input files for orography are located ('$home_dir/fix/fix_orog')
3	ntiles	Number of tiles (=1)
4	tile	Tile number (=7)
5	name	(temporary) Name of the parent directory for grid and orography files

6	grid_dir	Path to the directory for grid and mosaic files (‘/$name/grid’)
7	orog_dir	Path to the directory for orography files (‘/$name/orog’)
8	filter_dir	Path to the directory for topography filtering (same as ‘orog_dir’)

Table 5.1: Variables of the script for creating grid and static fields.

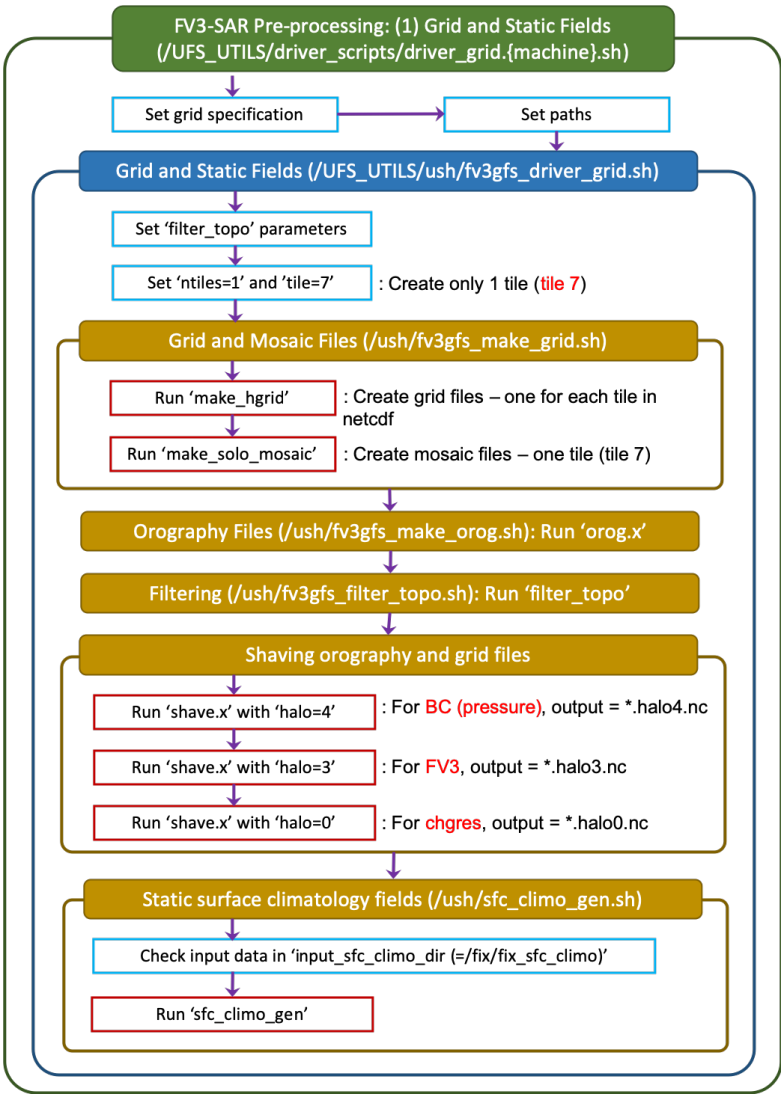

Figure 5.1: Structure of pre-processing for grid and static fields

5.2 Grid Generation

A high-resolution regional grid can be created from a low-resolution global grid by grid refinement. Refinement is a multiplicative factor named 'refine_ratio' in the scripts. Make sure that **the refined grid and its orography files keep using the resolution of the original low-resolution global grid on their file names** to avoid inaccurate filtering. For example, if a regional domain was created from a C768 domain (13*km*) with 'refine_ratio=3', its grid and orography files should be named 'C768_grid.tile7.haloX.nc' and 'C768_oro_data.tile7.haloX.nc', respectively, even though their resolution (3*km*) is three times as fine as their original domain.

Grid and mosaic files contain the longitudes and latitudes of a regional *FV3* super-grid, and other records that describe the model grid.

```
cd {TOP_dir}/UFS_UTILS/ush/
vim fv3gfs_make_grid.sh
```

1. Grid resolution of the original grid:

Table 5.2: Grid resolution of the original grid:

Resolution ('res')	C96	C192	C384	C768	C1152	C3072
Cell size (*km*)	100	50	25	13	8.5	3.2

2. Grid file by 'make_hgrid':

   (a) Namelist of 'make_hgrid': Table 4.3

   (b) Additional variables for the grid-generation script:

No.	Variable name	Description
1	outdir	Working directory
2	exec_dir	Path to the directory where the executable is located
3	excutable	Grid generation program ('make_hgrid')
4	grid_dir	Directory where a grid file is created ($TMPDIR/grid))

Table 5.3: Additional variables for grid generation.

Note) The variables of the namelist for the specific regional domain can be easily found by *fv3grid* as described in Section 10.1.

3. Mosaic file by 'make_solo_mosaic':

No.	Variable name	Description
1	num_tiles	Number of tiles (=1)
2	dir	Working directory
3	mosaic	Output file name (=C{res}_mosaic)
4	tile_file	Input grid file name (=C{res}_grid.tile7.nc)

Table 5.4: Variables for mosaic generation.

- **Note)** The number of tiles is 1 for a regional grid (tile number=7), 6 for a global grid (tile number=1–6), and 7 for a nest case (tile number: 1–6 for a parent, 7 for a child).

5.3 Orography Generation

An orography file contains land mask, terrain, and gravity wave drag fields.

```
cd {TOP_dir}/UFS_UTILS/ush/
vim fv3gfs_make_orog.sh
```

where

No.	Variable name	Description
1	$nargv	Grid type (=7 for a cubed-sphere grid, specified in 'fv3gfs_driver_grid.sh')
2	excutable	Program for creating orography files (='orog.x')
3	indir	Directory for input files (same as '$topo' in 'fv3gfs_dirver_grid.sh')

Table 5.5: Variables of the script for orography.

- Input files:

No.	Input file	Description	Fort
1	C{res}_grid.tile7.nc	Grid file	
2	thirty.second.antarctic.new.bin	Grumbine 30" Antarctica orog. (GICE)	fort.15

3	landcover30.fixed	UMD 30" lake mask	fort.10
4	gmted2010.30sec.int		fort.235

Table 5.6: Input files and fort numbers for orography.

- Output file: 'oro.C{res}.tile7.nc'

No.	Variable	Description
1	geolon	Longitudes
2	geolat	latitudes
3	slmsk	Sea-land mask
4	land_frac	Land fraction for each cell
5	orog_raw	Raw data of orography
6	orog_filt	Filtered data of orography
7	stddev	
8	convexity	
9	oa1–4	Orographic asymmetry parameters (Kim and Arakawa, 1995)
10	ol1–4	Orographic convexity parameters (Kim and Arakawa, 1995)
11	theta	
12	gamma	
13	sigma	
14	elvmax	

Table 5.7: Variables included in the output file.

The source code of 'orog.x' can be found in:

```
cd {TOP_dir}/UFS_UTILS/sorc/orog.fd/
vim mtnlm7_oclsm.f
```

5.4 Filtering Topography

```
cd {TOP_dir}/UFS_UTILS/ush/
vim fv3gfs_filter_topo.sh
```

where

No.	Variable name	Description
1	stretch	Grid stretch factor
2	refine_ratio	Grid refinement ratio
3	excutable	Program for filtering topography (=‘filter_topo’)
4	mosaic_grid	Mosaic file
5	topo_file	Orography file

Table 5.8: Variables of the script for orography.

- Parameters:

No.	Parameter	Grid resolution					
		C96	C192	C384	C768	C1152	C3072
1	cd4	0.12	0.15	0.15	0.15	0.15	0.15
2	peak_fac	1.1	1.05	1.0	1.0	1.0	1.0
3	max_slope	0.12	0.12	0.12	0.12	0.16	0.30
4	n_del2_weak	8	12	12	16	20	24

Table 5.9: Parameters for filtering topography.

- Input files:

No.	Input file	Description
1	C{res}_grid.tile7.nc	Grid file
2	C{res}_mosaic.nc	Mosaic file
3	oro.C{res}.tile7.nc	Orography file

Table 5.10: Input files for filtering topography.

- Output file: ‘oro.C{res}.tile7.nc’

5.5 Halo Files for Boundary, *FV3*, and *chgres_cube*

Three types of orography and grid files (halo0, halo3, and halo4) are created by the executable 'shave.x' for initial and boundary fields:

```
cd {TOP_dir}/UFS_UTILS/ush/
vim fv3gfs_driver_grid.sh
```

where

No.	Variable	Description
1	npts_cgx	Number of grid points (*i*-index, latitudinal)
2	npts_cgy	Number of grid points (*j*-index, longitudinal)
3	halo	Lateral boundary halo size (=3)
4	halop1	halo+1 (=4)

Table 5.11: Variables for halo files.

Steps:

1. Create a domain with 4 boundary rows/columns for pressure (halo4).
   - C{res}_oro_data.tile7.halo4.nc
   - C{res}_grid.tile7.halo4.nc
2. Create a domain with 3 boundary rows/columns for lateral boundary conditions of *FV3* (halo3).
   - C{res}_oro_data.tile7.halo3.nc
   - C{res}_grid.tile7.halo3.nc
3. Create a domain with 0 boundary row/column for *chgres* (halo0).
   - C{res}_oro_data.tile7.halo0.nc
   - C{res}_grid.tile7.halo0.nc

5.6 Regional Static (Fix) Fields

Time-independent surface fields such as soil and vegetation types for a regional domain are created by the executable 'sfc_climo_gen':

```
cd {TOP_dir}/UFS_UTILS/ush/
vim sfc_climo_gen.sh
```

where

No.	Variable	Description
1	input_sfc_climo_dir	Directory for input files (=$home_dir/fix/fix_sfc_climo)
2	FIX_FV3	$out_dir
3	SAVE_DIR	$sout_dir/fix_sfc
4	mosaic_file_mdl	$mosaic_file
5	orog_dir_mdl	$FIX_FV3
6	orog_files_mdl	Orography file (='C{res}_oro_data.tile7.nc')
7	maximum_snow_albedo_method	Interpolation method (default="bilinear")
8	snowfree_albedo_method	Interpolation method (default="bilinear")
9	vegetation_greenness_method	Interpolation method (default="bilinear")

Table 5.12: Variables for static surface fields.

- Input files:

No.	Variable	Input file	Description
1	input_facsf_file	facsf.1.0.nc	Fractional coverage for strong/weak zenith angle dependent albedo
2	input_substrate_temperature_file	substrate_temperature.2.6x1.5.nc	Soil substrate temp.
3	input_maximum_snow_albedo_file	maximum_snow_albedo.0.05.nc	Maximum snow albedo
4	input_snowfree_albedo_file	snowfree_albedo.4comp.0.05.nc	Snow-free albedo
5	input_slope_type_file	slope_type.1.0.nc	Slope type
6	input_soil_type_file	soil_type.statsgo.0.05.nc	Soil type
7	input_vegetation_type_file	vegetation_type.igbp.0.05.nc	Vegetation type
8	input_vegetation_greenness_file	vegetation_greenness.0.144.nc	Vegetation greenness

Table 5.13: Input files for filtering topography.

- Output files:

No.	Output file
1	C{res}.facsf.1.0.nc
2	C{res}.substrate_temperature.2.6x1.5.nc
3	C{res}.maximum_snow_albedo.0.05.nc
4	C{res}.snowfree_albedo.4comp.0.05.nc
5	C{res}.slope_type.1.0.nc
6	C{res}.soil_type.statsgo.0.05.nc
7	C{res}.vegetation_type.igbp.0.05.nc
8	C{res}.vegetation_greenness.0.144.nc

Table 5.14: Output files for filtering topography.

Note) The 'sfc_climo_gen' program only runs with an MPI task count that is a multiple of six. This is a requirement of the ESMP libraries. Large grids may require tasks spread across multiple nodes.

Chapter 6

UFS_UTILS: Initial and Lateral Boundary Fields

6.1 External Model Data for IC/LBC in Workflow

Machine-specific parent directories of the external model data for IC ('EXTRN_MDL_FILES_SYSBASEDIR _ICS') and LBC ('EXTRN_MDL_FILES_SYSBASEDIR_ICS') are specified in '{HOMErrfs}/ ush/set_extrn_mdl_params.sh' as shown in Table 6.1. On Hera, the recent two-day data are provided in the directory. You should check if the data that you want to simulate exist in the directory. If you want to run the regional workflow for other dates, these paths should be modified to point the data set.

- 'nco' mode: '{COMINgfs}' for both IC and LBC
- 'community' mode:

Model	Machine	Type	Data directory ('{sysbasedir}')
FV3GFS	Hera	IC	/scratch1/NCEPDEV/rstprod/com/gfs/prod
		LBC	/scratch1/NCEPDEV/rstprod/com/gfs/prod
	Orion	IC	-
		LBC	-
	WCOSS_C	IC	/gpfs/hps/nco/ops/com/gfs/prod
		LBC	/gpfs/hps/nco/ops/com/gfs/prod

RAPX	Hera	IC	/scratch2/BMC/public/data/gsd/rap/full/wrfnat
		LBC	/scratch2/BMC/public/data/gsd/rap/full/wrfnat
	Orion	IC	-
		LBC	-
	WCOSS_C	IC	-
		LBC	-
HRRRX	Hera	IC	/scratch2/BMC/public/data/gsd/hrrr/conus/wrfnat
		LBC	/scratch2/BMC/public/data/gsd/hrrr/conus/wrfnat
	Orion	IC	-
		LBC	-
	WCOSS_C	IC	-
		LBC	-

Table 6.1: External model data for IC and LBC

The file names of the external model data are specified in '{HOMErrfs}/ush/get_extrn_mdl_file_dir_info.sh' as:

Model	Type	Format	File name ('fns_on_disk' / 'fns_in_arcv')
FV3GFS	IC ('ANL')	nemsio	gfs.t{HH}z.{'atm'/'sfc'}anl.nemsio
		grib2	gfs.t{HH}z.pgrb2.0p25.f000
	LBC ('FCST')	nemsio	gfs.t{HH}z.atmf{fcst_hhh}.nemsio
		grib2	gfs.t{HH}z.pgrb2.0p25.f{fcst_hhh}
RAPX	IC ('ANL')	-	{YY}{DDD}{HHH}{mn}{fcst_hh}{fcst_mn}
	LBC ('FCST')	-	{YY}{DDD}{HHH}{mn}{fcst_hh}{fcst_mn}
HRRRX	IC ('ANL')	-	{YY}{DDD}{HHH}{mn}{fcst_hh}{fcst_mn}
	LBC ('FCST')	-	{YY}{DDD}{HHH}{mn}{fcst_hh}{fcst_mn}

Table 6.2: File name of the external model data

The final system directory, which means the full path to the external model data, based on Table 6.1 are specified as:

Model	Full path to the external model data (‘{sysdir}’)
FV3GFS	{sysbasedir}/gfs.{YYYYMMDD}/{HH}
RAPX	{sysbasedir}
HRRRX	{sysbasedir}

Table 6.3: System directories of the external model data

6.2 Global Time-dependent Data

6.2.1 GFS Data: GRIB2

1. 0.25° data (last 10days only): ‘gfs.t{HH}z.pgrb2.0p25.f{XXX}’

```
https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.{YYYYMMDD}/{HH}/
```

   Download on HPC:

```
cd [directory where data will be downloaded]
wget -c https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.{
    YYYYMMDD}/{HH}/[file name]
```

2. 0.5° data: ‘gfs_4_{YYYYMMDD}_00{HH}_{XXX}.grb2’

```
https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-
    forcast-system-gfs/
# Select ‘Data Access Links’ at ‘GFS Forecasts/004-Domain’
```

   Download from ‘HTTPS’ on HPC:

```
cd [directory where data will be downloaded]
wget -c https://nomads.ncdc.noaa.gov/data/gfs4/{YYYYMM}/{YYYYMMDD}/[file
    name]
```

3. 1.0° data: ‘gfs_3_{YYYYMMDD}_00{HH}_{XXX}.grb2’

```
https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-
    forcast-system-gfs/
# Select ‘Data Access Links’ at ‘GFS Forecasts/003-Domain’
```

Download from 'HTTPS' on HPC:

```
cd [directory where data will be downloaded]
wget -c https://nomads.ncdc.noaa.gov/data/gfs-avn-hi/{YYYYMM}/{YYYYMMDD}/[
    file name]
```

6.2.2 GFS Data: NEMSIO

1. T1534 gaussian (last 10 days only)

```
https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.{YYYYMMDD}/{HH}/
```

   - Atmospheric fields: 'gfs.t{HH}z.atmanl.nemsio'
   - Surface fields: 'gfs.t{HH}z.sfcanl.nemsio'

   Download on HPC:

```
cd [directory where data will be downloaded]
wget -c https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.{
    YYYYMMDD}/{HH}/[file name]
```

6.3 *chgres_cube*

The program *chgres_cube* creates initial data using history or restart data from another FV3 run (GRIB2), or the NEMS version of the spectral GFS (NEMSIO) for a 'cold start' of the forecast model. It converts atmospheric, surface, and nst data. The configuration of *chgres_cube* for a Regional Domain is as follows:

- References

   1. /UFS_UTILS/sorc/chgres_cube.fd/program_setup.f90
   2. /UFS_UTILS/ush/chgres_cube.sh
   3. github.com/NOAA-EMC/UFS_UTILS/blob/release/public-v1/docs/source/chgres_cube.rst

No.	Variable name	Description	Usage (data type)
1	mosaic_file_target_grid	Mosaic file (path and name) for the target grid	GRIB2/NEMSIO
2	fix_dir_target_grid	Directory (path and name) containing pre-computed fixed data on the target grid (e.g. soil type)	GRIB2/NEMSIO
3	orog_dir_input_grid	Directory (path and name) containing the input grid orography files. Only used for 'restart' or 'history' input type	GRIB2/NEMSIO
4	orog_files_input_grid	Input grid orography files. Only used for 'restart' or 'history' input type	GRIB2/NEMSIO
5	orog_dir_target_grid	Directory (path and name) containing orography files for the input grid.	GRIB2/NEMSIO
6	orog_files_target_grid	Orography files (name) for the target grid	GRIB2/NEMSIO
7	vcoord_files_target_grid	Vertical coordinate definition file (path and name)	GRIB2/NEMSIO
8	data_dir_input_grid	Directory (path and name) containing atm or sfc files on the input grid	GRIB2/NEMSIO
9	grib2_file_input_grid	File name of GRIB2 input data	GRIB2
10	varmap_file	Directory (path and name) containing the variable mapping (VARMAP) table	GRIB2
11	atm_files_input_grid	File name of input atmospheric data Not used for INPUT_TYPE="restart"	NEMSIO
12	sfc_files_input_grid	File name of input surface data	NEMSIO
13	cycle_mon	Cycle month	GRIB2/NEMSIO
14	cycle_day	Cycle day	GRIB2/NEMSIO
15	cycle_hour	Cycle hour	GRIB2/NEMSIO
16	convert_atm	Convert atmospheric data when 'true'.	GRIB2/NEMSIO
17	convert_sfc	Convert surface data when 'true'.	GRIB2/NEMSIO
18	convert_nst	Convert nst data when 'true'.	NEMSIO
19	input_type	Input data type: • 'restart' - Tiled warm restart files (netcdf) • 'history' - Tiled history files (netcdf) • 'gaussian_nemsio' - Gaussian NEMSIO files • 'gaussian_netcdf' - Gaussian NetCDF files	GRIB2/NEMSIO

		• 'grib2' - FV3GFS GRIB2 files • 'gfs_gaussian_nemsio' - Spectral GFS NEMSIO files • 'gfs_spectral' - Spectral GFS sigio/sfcio files	
20	tracers	Names of atmospheric tracers to be processed. These names will be used to identify the tracer records in the output files	NEMSIO
21	tracers_input	Names of atmospheric tracers in the input file. In most cases, they are different from those of tracers	NEMSIO
22	regional	For regional target grids: • '0' : uniform, stretch, or nest (for global domain) • '1' : Creating initial and lateral boundary files from atmospheric/surface data • '2' : Creating lateral boundary file only	GRIB2/NEMSIO
23	halo_bndy	Number of row/columns of lateral halo where pure lateral boundary conditions are applied (regional target grids only)	GRIB2/NEMSIO
24	halo_blend	Number of row/columns of blending halo where model tendencies and lateral boundary tendencies are applied. (regional target grids only)	GRIB2/NEMSIO

Table 6.4: Namelist of *chgres_cube* for a regional domain.

Notes)

- Difference of 'input_type' in the namelist of *chgres_cube* between the latest version of UFS_UTILS ('EMC') and the version used in SRW App ('NCAR').

No.	Description	Latest version	SRW App
1	Tiled warm restart files	restart	restart
2	Tiled history files	history	history
3	Gaussian NEMSIO files	gaussian_nemsio	gaussian
4	Gaussian NetCDF files	gaussian_netcdf	-
5	FV3GFS GRIB2 files	grib2	grib2
6	Spectral GFS NEMSIO files	gfs_gaussian_nemsio	gfs_gaussian

7	Spectral GFS sigio/sfcio files	gfs_spectral	gfs_spectral

Table 6.5: Difference of the 'input_type' options in the namelist of *chgres_cube*

- Some caveats in initializing with GRIB2 data (from Reference 3)

  1. GRIB2 data do not contain the fields required for the NSST (Near Sea Surface Temperature) scheme.
     - In GFS, we use the foundation temperature ('Tref') and add a diurnal warming/-cooling layer using NSST. This is the surface temperature that is passed to the atmospheric boundary layer. This is a critical feature especially for DA. GRIB2 files only store 'Tsfc' and it is set as the foundation temperature. So the diurnal heating/cooling is baked into the initial condition. This can be critical if the model is initialized when the ocean is warm and the diurnal warming is at the peak.
     - A user has two options: 1) to use a spin-up cycle to spin up NSST by setting 'nstf_name=[2,1,0,0,0] in 'input.nml', or 2) to run the model without any NSST option by setting 'nstf_name=[0,0,0,0,0] in 'input.nml'.
  2. Relatively low-resolution data, compared to the NCEP operation GFS, may result in inaccurate initialization especially for the surface fields.
  3. Sea/lake ice thickness and column temperatures are not available. Nominal values of $1.5m$ and $265K$ are used.
  4. Soil moisture is evaluated using bilinear interpolation. It may result in inaccurate latent/sensible heat fluxes.
  5. Ozone is not available at all isobaric levels. Missing levels are set to a nominal value defined in the variable mapping (VARMAP) file.
  6. It is not guaranteed to use old GFS data (older than GFS v14).

6.4 Initial Conditions: Cold Start

Creates initial conditions (ICs and BC at the initial time) for a regional domain using global GFS data:

```
cd {HOMEfv3}/scripts/
vim exregional_make_ic.sh
```

where {HOMEfv3} is the home directory of the regional_workflow package.

Set the links to use the grid, orography, and static field files with **FOUR (4)** halos:

```
# Grid
ln -sf {FIXsar}/C{res}_grid.tile7.halo4.nc {FIXsar}/C{res}_grid.tile7.nc
# Orography
ln -sf {FIXsar}/C{res}_oro_data.tile7.halo4.nc {FIXsar}/C{res}_oro_data.tile7.nc
# Static fields
ln -sf {FIXsar}/C{res}.[variables].tile7.halo4.nc {FIXsar}/C{res}.[variables].
    tile7.nc
```

- List of the static fields:

No.	Variable name	Transferred name
1	vegetation_greenness.tile7.halo4.nc	C{res}.vegetation_greenness.tile7.nc
2	soil_type.tile7.halo4.nc	C{res}.soil_type.tile7.nc
3	slope_type.tile7.halo4.nc	C{res}.slope_type.tile7.nc
4	substrate_temperature.tile7.halo4.nc	C{res}.substrate_temperature.tile7.nc
5	facsf.tile7.halo4.nc	C{res}.facsf.nc
6	maximum_snow_albedo.tile7.halo4.nc	C{res}.maximum_snow_albedo.tile7.nc
7	snowfree_albedo.tile7.halo4.nc	C{res}.snowfree_albedo.tile7.nc
8	vegetation_type.tile7.halo4.nc	C{res}.vegetation_type.tile7.nc

Table 6.6: Static fields transferred to the INPUT directory.

- Other input files:

  1. C{res}_mosaic.nc (mosaic file)
  2. global_hyblev.l{LEVS}.txt (vertical coordinate definition file)
  3. GFSphys_var_map.txt (only for GRIB2)

Create namelist and run *chgres_cube*:

- Namelist for *chgres_cube* (GRIB2):

No.	Variable	Input
1	mosaic_file_target_grid	"{FIXsar}/C{res}_mosaic.nc"
2	fix_dir_target_grid	"{FIXsar}"
3	orog_dir_target_grid	"{FIXsar}"
4	orog_files_target_grid	"C{res}_oro_data.tile7.halo4.nc"
5	vcoord_file_target_grid	"{FIXam}/global_hyblev.l{LEVS}.txt"
6	data_dir_input_grid	"gfs.{ymd}/{hour}"
7	grib2_file_input_grid	"gfs.t{cyc}z.pgrb2.0p25.f{hour_name}"
8	varmap_file	"{UFS_UTILS}/parm/varmap_tables/GFSphys_var_map.txt"
9	input_type	"grib2"
10	cycle_mon	{month}
11	cycle_day	{day}
12	cycle_hour	{cyc}
13	convert_atm	**.true.**
14	convert_sfc	**.false.**
15	convert_nst	**.false.**
16	regional	**1**
17	halo_bndy	4
18	halo_blend	10

Table 6.7: Namelist for *chgres_cube* with GRIB2 data.

- Namelist for *chgres_cube* (NEMSIO):

No.	Variable	Input
1	mosaic_file_target_grid	"{FIXsar}/C{res}_mosaic.nc"
2	fix_dir_target_grid	"{FIXsar}"
3	orog_dir_target_grid	"{FIXsar}"
4	orog_files_target_grid	"C{res}_oro_data.tile7.halo4.nc"
5	vcoord_file_target_grid	"{FIXam}/global_hyblev.l{LEVS}.txt"
6	data_dir_input_grid	"{ANLDIR}"
7	atm_files_input_grid	"gfs.t{cyc}z.atmanl.nemsio"

8	sfc_files_input_grid	"gfs.t{cyc}z.sfcanl.nemsio"
9	cycle_mon	{month}
10	cycle_day	{day}
11	cycle_hour	{cyc}
12	convert_atm	**.false. (.true.)**
13	convert_sfc	**.true.**
14	convert_nst	**.true.**
15	input_type	"gaussian_nemsio"
16	tracers	"sphum", "liq_wat", "o3mr", "ice_wat", "rainwat", "snowwat", "graupel"
17	tracers_input	"spfh", "clwmr", "o3mr", "icmr", "rwmr", "snmr", "grle"
18	regional	**1**
19	halo_bndy	4
20	halo_blend	10

Table 6.8: Namelist for *chgres_cube* with NEMSIO data.

Note) The list of 'tracers' and 'tracers_input' can be found in Table 2.38.
Note) 'regional=1' because both initial and boundary conditions are evaluated at the initial time.

Rename output files and move them to the INPUT directory for *FV3*:

```
mv gfs_ctrl.nc {INPdir}/.
mv gfs.bndy.nc {INPdir}/gfs_bndy.tile7.000.nc
mv out.atm.tile1.nc {INPdir}/gfs_data.tile7.nc
mv out.sfc.tile1.nc {INPdir}/sfc_data.tile7.nc
```

- Output files:

  1. gfs_ctrl.nc
  2. gfs.bndy.nc (boundary condition at the initial time)
  3. out.atm.tile1.nc (initial conditions of time-dependent fields inside the regional domain)
  4. out.sfc.tile1.nc (initial conditions of static fields inside the regional domain)

6.5 Lateral Boundary Conditions

Creates lateral boundary conditions every 3 hours after initialization for a regional domain using global GFS data:

```
cd {HOMEfv3}/scripts/
vim exregional_make_bc.sh
```

where {HOMEfv3} is the home directory of the regional_workflow package.

Set the links to use the grid, and orography files with **FOUR (4)** halos:

```
# Grid
ln -sf {FIXsar}/C{res}_grid.tile7.halo4.nc {FIXsar}/C{res}_grid.tile7.nc
# Orography
ln -sf {FIXsar}/C{res}_oro_data.tile7.halo4.nc {FIXsar}/C{res}_oro_data.tile7.nc
```

- Other input files:

  1. C{res}_mosaic.nc (mosaic file)
  2. global_hyblev.l{LEVS}.txt (vertical coordinate definition file)

Create namelist and run *chgres_cube*:

- Namelist for *chgres_cube* (GRIB2):

No.	Variable	Input
1	mosaic_file_target_grid	"{FIXsar}/C{res}_mosaic.nc"
2	fix_dir_target_grid	"{FIXsar}"
3	orog_dir_target_grid	"{FIXsar}"
4	orog_files_target_grid	"C{res}_oro_data.tile7.halo4.nc"
5	vcoord_file_target_grid	"{FIXam}/global_hyblev.l{LEVS}.txt"
6	data_dir_input_grid	"gfs.{ymd}/{hour}"
7	grib2_file_input_grid	"gfs.t{cyc}z.pgrb2.0p25.f{hour_name}"
8	varmap_file	"{UFS_UTILS}/parm/varmap_tables/GFSphys_var_map.txt"
9	input_type	"grib2"
10	cycle_mon	{month}

11	cycle_day	{day}
12	cycle_hour	{cyc}
13	convert_atm	**.true.**
14	convert_sfc	**.false.**
15	convert_nst	**.false.**
16	regional	**2**
17	halo_bndy	4
18	halo_blend	10

Table 6.9: Namelist for *chgres_cube* with GRIB2 data.

- Namelist for *chgres_cube* (NEMSIO):

No.	Variable	Input
1	mosaic_file_target_grid	"{FIXsar}/C{res}_mosaic.nc"
2	fix_dir_target_grid	"{FIXsar}"
3	orog_dir_target_grid	"{FIXsar}"
4	orog_files_target_grid	"C{res}_oro_data.tile7.halo4.nc"
5	vcoord_file_target_grid	"{FIXam}/global_hyblev.l{LEVS}.txt"
6	data_dir_input_grid	"{INIDIR}"
7	atm_files_input_grid	"gfs.t{cyc}z.atmf{hour_name}.nemsio"
8	sfc_files_input_grid	"gfs.t{cyc}z.sfcanl.nemsio"
9	cycle_mon	{month}
10	cycle_day	{day}
11	cycle_hour	{cyc}
12	convert_atm	**.true.**
13	convert_sfc	**.false.**
14	convert_nst	**.false.**
15	input_type	"gaussian_nemsio"
16	tracers	"sphum", "liq_wat", "o3mr", "ice_wat", "rainwat", "snowwat", "graupel"
17	tracers_input	"spfh", "clwmr", "o3mr", "icmr", "rwmr", "snmr", "grle"
18	regional	**2**

19	halo_bndy	4
20	halo_blend	10

Table 6.10: Namelist for *chgres_cube* with NEMSIO data.

Note) 'regional=2', 'convert_sfc=.false.', and 'convert_nst=.false.' for boundary conditions.

Rename output files and move them to the INPUT directory for *FV3*:

```
mv gfs.bndy.nc {INPdir}/gfs\_bndy.tile7.{hour_name}.nc
```

- Output files:

  1. gfs.bndy.nc (boundary condition at the certain hour)

Chapter 7

Step by Step to a New Case without Workflow (GFDL Grid)

7.1 Build Pre-processing Programs and FV3GFS model

In order to run FV3GFS as a stand-alone regional domain, a user must first get the fv3gfs pre-processing programs and scripts, and the fv3gfs model code.

1. UFS_UTILS:

   Create a directory to hold the source code. It is referred to as "**{TOP_DIR}** ".

```
mkdir {TOP_DIR (e.g.: fv3sar)}
```

   Clone UFS_UTILS from GitHub.

```
cd {TOP_DIR}
git clone -b develop --recursive https://github.com/NOAA-EMC/UFS_UTILS
```

   Build program:

```
cd {TOP_DIR}/UFS_UTILS/
sh build_all.sh > utl.log
```

   Note) In the above 'utl.log' file, you can check the versions of modules such as Intel and NetCDF. Make sure that the corresponding versions of modules are loaded in the scripts. For example, the corresponding NetCDF module can be loaded as follows:

- On Hera:

```
module use /scratch2/NCEPDEV/nwprod/NCEPLIBS/modulefiles
module load netcdf_parallel/4.7.4
```

- On Orion:

```
module load intel/2020 impi/2020
module load netcdf_parallel/4.7.2-parallel

ulimit -s unlimited
ulimit -a
```

Note) To avoid memory-related issues, the stack size should be set as above.

- On WCOSS:

```
module use /usrx/local/nceplibs/dev/NCEPLIBS/modulefiles
module load netcdf_parallel/4.7.4
```

Link the global 'fix' directory:

```
cd {TOP_DIR}/UFS_UTILS/fix
vim link_fixdirs.sh
# Check if 'FIX_DIR' is set up correctly for the specific machine (see below
    ).
ksh link_fixdirs.sh [run_env] [machine]
```

where [run_env] is 'emc' or 'nco', and [machine] is 'dell', 'hera', or 'orion'.

- Path to 'FIX_DIR':

No.	Machine	Path to 'fix'
1	WCOSS (Mars, Dell)	/gpfs/dell2/emc/modeling/noscrub/emc.glopara/git/fv3gfs/fix
2	Hera	/scratch1/NCEPDEV/global/glopara/fix
3	Orion	/work/noaa/fv3-cam/emc.glopara/fix

Table 7.1: Path to the global 'fix' directory.

2. UFS weather model:

Clone the model from GitHub.

```
cd {TOP_DIR}
git clone -b develop --recursive https://github.com/ufs-community/ufs-
    weather-model
```

Build programs:

```
cd {TOP_DIR}/ufs-weather-model/tests
sh compile.sh {TOP_DIR}/ufs-weather-model/FV3 [platform] ''32BIT=Y'' 32bit
[clean_before] [clean_after] >& make.out.32bit
```

where [platform] is 'wcoss_dell_p3, 'hera.intel', or 'orion.intel'. The last two optional arguments [clean_before] and [clean_after] control whether or not to run make clean to remove temporary files. The default values are 'YES'. Specifying 'NO' will skip the cleaning step, which will speed up repeating compilation, which is useful during debugging.

7.2 Grid, Orography, and Static Fields

The first step in generating initial conditions for a forecast is to execute a script that generates gird information and specific time-independent fields in the NetCDF format such as:

- 'mosaic' and 'grid' files: lat/lon and other info describing the model grid.
- 'oro' files: land mask, terrain and gravity wave drag fields.
- Surface climo. fields: soil type, vegetation greenness, and albedo.

```
# Driver script to create a cubic-sphere based model grid:
cd {TOP_DIR}/UFS_UTILS/driver_scripts
vim driver_grid.[machine].sh
```

This directory contains scripts for each individual system (cray/dell/hera/orion). Edit the script for the appropriate system and make sure these are set:

```
export res=768
export gtype='regional'
export stretch_fac=2.0
export target_lon=-45.0
export target_lat=-20.0
```

```
export refine_ratio=3
export istart_nest=599
export jstart_nest=627
export iend_nest=1240
export jend_nest=1240
export halo=3
```

where

No.	Variable name	Description
1	res	Grid resolution
2	gtype	Grid type ('uniform' and 'stretch' for the global domain, 'nest' for a nest domain in two-way nesting, or 'regional' for a regional domain)
3	stretch_fac	Stretching factor for the grid
4	target_lon	Center longitude of the highest resolution tile
5	target_lat	Center latitude of the highest resolution tile
6	refine_ratio	Grid refinement ratio
7	istart_nest	Lowest *i*-index of the parent-tile super-grid corresponding to the regional (nested) domain
8	jstart_nest	Lowest *j*-index of the parent-tile super-grid corresponding to the regional (nested) domain
9	iend_nest	Highest *i*-index of the parent-tile super-grid corresponding to the regional (nested) domain
10	jend_nest	Highest *j*-index of the parent-tile super-grid corresponding to the regional (nested) domain
11	halo	Lateral boundary halo for FV3

Table 7.2: Variables for grid specification.

- **Note)** The above variables for the specific regional domain can be easily found by *fv3grid* as described in Section 10.1.
- **Note)** For a regional domain, the 'gtype' must be 'regional' only.
- **Note)** 'halo=3' is a fundamental characteristic of FV3 though we need the 4th boundary row for its pressure value. Every MPI task subdomain has a halo of 3 rows. The *chgres* generates

4 rows of boundary data to allow for a computational halo of 3 rows that are needed at those outermost integration points.

The default directory where the output files will be created is defined by the variable 'out_dir':

```
export TMPDIR=/work/noaa/fv3-cam/$LOGNAME/test/fv3_grid.$gtype
export out_dir=/work/noaa/fv3-cam/$LOGNAME/test/${domain}_C${res}
```

Then, submit the 'driver_grid' script:

- On Hera:

```
sbatch driver_grid.hera.sh
```

- On Orion:

```
sbatch driver_grid.orion.sh
```

- On WCOSS (Dell):

```
bsub < driver_grid.dell.sh
```

Once the job finished, the 'out_dir' should include the following NetCDF files where some ('*.halo4.nc') are required for creating initial and boundary files in Sections 7.3 and 7.4 :

1. In '{out_dir}/':
   - C{res}_grid.tile7.halo[0,3,4].nc
   - C{res}_mosaic.nc
   - C{res}_oro_data.tile7.halo[0,3,4].nc
2. In '{out_dir}/fix_sfc/':
   - C{res}.facsf.tile7.halo[0,4].nc
   - C{res}.maximum_snow_albedo.tile7.halo[0,4].nc
   - C{res}.slope_type.tile7.halo[0,4].nc
   - C{res}.snowfree_albedo_type.tile7.halo[0,4].nc
   - C{res}.soil_type.tile7.halo[0,4].nc
   - C{res}.substrate_temperature.tile7.halo[0,4].nc
   - C{res}.vegetation_greenness.tile7.halo[0,4].nc
   - C{res}.vegetation_type.tile7.halo[0,4].nc

7.3 Initial Conditions

Initial hourly atmospheric and surface data for a regional domain are created by running the script in the 'regional_scripts' directory:

```
cd {HOMEreg}/regional_scripts
# atmospheric (GRIB2)
vim run.regional_chgres_{domain}.atm.[machine]
# surface (NEMSIO)
vim run.regional_chgres_{domain}.sfc.[machine]
```

where {HOMEreg} is the parent directory where 'regional_scripts' and 'regional_files' are located.

Edit the script for the appropriate resolution, domain and case names, and date and time of initial conditions (YYYYMMDDHH):

```
res=768
domain=''brz''
CASE=${domain}_C${res}
DATE=2020041900
```

where

No.	Variable name	Description
1	res	Grid resolution
2	domain	Domain name
3	CASE	Name of the directory where the grid and orography files, and 'fix_sfc' sub-directory (={domain}_C{res})
4	DATE	Date and time of data for initial conditions

Table 7.3: Variables in the script for IC.

Note) The 'DATE' should be corresponding to the data that are set in the namelist for *chgres* ('data_dir_input_grid') in the script (line 64).

Set the 'BASEDIR' to the 'UFS_UTILS' directory containing '/exec/chgres_cube', and the 'FIXDIR' to the parent directory of {CASE}:

```
BASEDIR=/work/noaa/fv3-cam/$LOGNAME/UFS_UTILS
```

```
FIXDIR=/work/noaa/fv3-cam/$LOGNAME/test
```

Check that the required input files exist in the directory '$FIXDIR/$CASE/' and its subdirectory 'fix_sfc'. Make sure that the variable '$CASE' should be the same directory as 'out_dir' in the 'driver_gird' script (Section 7.2).

```
# grid and orography
cd {FIXDIR}/{CASE}/
# static fields
cd fix_sfc
```

Set a path to a working directory ($WORKDIR):

```
WORKDIR=/work/noaa/fv3-cam/$LOGNAME/test/chgres_fv3
```

Check that the input files for *chgres* exist in the right directories:

- NEMSIO:

```
vcoord_file_target_grid="$BASEDIR/fix/fix_am/global_hyblev.l65.txt"
data_dir_input_grid="/work/noaa/fv3-cam/chjeon/Data/gfs_nemsio/gfs.{ymd}/{
    hour}"
```

- GRIB2:

```
vcoord_file_target_grid="$BASEDIR/fix/fix_am/global_hyblev.l65.txt"
data_dir_input_grid="/work/noaa/fv3-cam/chjeon/Data/gfs_grb2/gfs.{ymd}/{hour}
    "
varmap_file=''$BASEDIR/parm/varmap_tables/GFSphys_var_map.txt''
```

Note) The recent GFS data can be downloaded from the NCEP's NOMADS server as shown in Section 6.2

Check the numbers of 'regional', 'halo_bndy', and 'halo_blend':

```
regional=1
halo_bndy=4
halo_blend=10
```

Then, submit the script:

```
sbatch run.regional_chgres_{domain}.atm.[machine]
sbatch run.regional_chgres_{domain}.sfc.[machine]
```

Check that output files exist in the case directory:

```
cd /work/noaa/fv3-cam/chjeon/test/{CASE}/
```

Output files:

- gfs_ctrl.nc
- gfs_bndy.tile7.000.nc
- gfs_data.tile7.nc
- sfc_data.tile7.nc

Note) The output files can be plotted with the python scripts in Section 10.2 ('gfs_bndy.tile7.000 .nc' and 'gfs_data.tile7.nc' – 'plot_fv3_icbc.py', and 'sfc_data.tile7.nc' – 'plot_fc3_icsfc.py').

7.4 Lateral Boundary Conditions

Boundary data for a regional domain are created:

```
cd {HOMEreg}/regional_scripts
vim run.regional_chgres_forBC_{domain}.[machine]
```

Edit the script for the appropriate resolution and simulation time (YYYYMMDDHH) as well as working directories:

```
res=768
CASE=BRZ_C$res
DATE=2020041900
BASEDIR=/work/noaa/fv3-cam/chjeon/UFS_UTILS
FIXDIR=/work/noaa/fv3-cam/chjeon/test
WORKDIR=/work/noaa/fv3-cam/chjeon/test/chgres_fv3_forBC
chour=3
end_hour=24
bc_interval=3
```

where

No.	Variable name	Description
1	res	Grid resolution
2	DATE	Date and time of data for lateral boundary conditions
3	BASEDIR	Path and name of the parent directory of 'UFS_UTILS'
4	FIXDIR	Path and name of the parent directory of {CASE}
5	WORKDIR	Path and name of the working directory
6	chour	Hours from the initial to the time for the next boundary file. It should be the same as 'bc_interval'. It is a relative number. The forecast always creates the output starting at 00.
7	end_hour	Hours for the last boundary files
8	bc_interval	Time interval between boundary files

Table 7.4: Variables for file relocation.

Check the numbers of 'regional', 'halo_bndy', and 'halo_blend':

```
regional=2
halo_bndy=4
halo_blend=10
```

Then, submit the script:

- On Hera:

```
sbatch run.regional_chgres_forBC_{domain}.hera
```

- On Orion:

```
sbatch run.regional_chgres_forBC_{domain}.orion
```

Check that output files exist in the case directory:

```
cd /work/noaa/fv3-cam/chjeon/test/{CASE}/
```

Output files:

- gfs_bndy.tile7.003.nc
- gfs_bndy.tile7.006.nc

- gfs_bndy.tile7.009.nc
- gfs_bndy.tile7.012.nc

Note) The output files can be plotted with the python script in Section 10.2 ('plot_fv3_icbc.py').

7.5 Relocation of Input Files

This will copy all the necessary fix, txt, and INPUT files for a run. It will also copy 'model_configure' and 'input_nml' files for c768 and c96 as well as sample run scripts.

```
cd {HOMEreg}/regional_scripts
vim setup.regional_runfiles_{domain}.sh
```

Modify the paths and resolution.

```
res=768
CASE=BRZ_C${res}
RUN_CASE=run_${CASE}
INPUTDIR=/work/noaa/fv3-cam/$LOGNAME/test
RUNDIR=/work/noaa/fv3-cam/$LOGNAME/test/${RUN_CASE}
UFS_UTILS_DIR=/work/noaa/fv3-cam/$LOGNAME/UFS_UTILS
```

where

No.	Variable name	Description
1	res	Grid resolution
2	CASE	Case name ('{domain}+C{res}'). The input data will be created in the directory of the same name
3	RUN_CASE	Name of a case-run directory ('run_{CASE}')
4	INPUTDIR	Path and name of the directory where the input and case-run sub-directories are located
5	RUNDIR	Path and name of the case-run directory
6	UFS_UTILS_DIR	Path to the 'UFS_UTILS' directory

Table 7.5: Variables for file relocation.

Check the data files for the target year in '/fix/fix_am/fix_co2_proj/' (Line 63):

```
# for i in 2019 2020
```

Run the script:

```
sh setup.regional_runfiles_{domain}.sh
```

7.6 Forecast Run

The 'diag_table' is empty so this must be populated with the required diagnostic/prognostic fields. In addition, the 'model_configure' files have quilting turned off.

Move to the case run directory:

```
cd {RUNDIR}
```

where {RUNDIR} is the case-run directory.

Select a file for the 'diag_table':

- Without I/O: 'diag_table.empty'

```
vim diag_table.empty
# The first line is just a descriptor.
# Edit the second line to the same dates in model_configuration
2020 04 19 00 0 0
```

- With I/O: 'diag_table.full'

```
vim diag_table.full
# The first line is just a descriptor.
# Edit the second line to the same dates in model_configuration
2020 04 19 00 0 0
```

- Same as a regional workflow: 'diag_table.tmp'

```
vim diag_table.tmp
# The first line is just a descriptor.
# Edit the second line to the same dates in model_configuration
2020 04 19 00 0 0
```

Check the configuration for FV3 (see Table 2.48):

```
vim model_configure.c{res}.{PE}.{CASE}
# Modify configuration as needed
start_year: 2020
start_month: 04
start_day: 19
start_hour: 00
nhours_fcst: 12

cen_lon:  -45.0
cen_lat: -20.0
lon1: -30.0
lat1: -10.0
lon2: 30.0
lat2: 10.0
```

Check the input namelist file for FV3 (see Section **??**):

```
vim input.nml.{CASE}
# Modify as needed
layout = 24,54
npx = 964
npy = 922
npz = 64
target_lat = -20.0
target_lon = -45.0
stretch_fac = 2.0
bc_update_interval = 3

# For near-boundary blending (under 'fv_core_nml')
regional_bcs_from_gsi = .false.
write_restart_with_bcs = .false.
nrows_blend = 10
```

Note) 'target_lat', 'target_lon', and 'stretch_fac' can be found in 'driver_grid.[machine].sh' (Section 7.2).

Check that the 'BASEDIR' has the machine-specific fv3 executable in 'ufs-weather-model':

```
vim run.c{res}.{case}
```

```
export BASEDIR=/scratch2/NCEPDEV/fv3-cam/$LOGNAME
# Modify the case run directory
WRKDIR=/scratch2/NCEPDEV/stmp1/$LOGNAME/${RUN_CASE}
```

Check links of the input files to the right name for *FV3*:

```
vim run.c{res}.{case}
ln -sf model_configure.c{res}.{PE}.{domain} model_configure
ln -sf input.nml.{domain} input.nml
ln -sf diag_table.tmp diag_table.   # same as workflow
```

Submit a job for a case:

- On Hera:

```
sbatch run.c{res}.{domain}
```

- On Orion:

```
sbatch run.c{res}.{domain}
```

Note) The executable used for this case run is '*fv3_32bit.exe*' that is located in '$BASEDIR/ufs-weather-model/tests/'. You should compile and replace it for additional development tests.

Output files:

1. grid_spec.nc
2. atmos_static.nc
3. dynfHHH.nc
4. phyfHHH.nc
5. time_stamp.out
6. logfHHH
7. logfile.xxxxxx.out

Note) If 'quilting=.false.' in 'model_configure', the result files of (3) and (4) will be replaced with 'fv3_history' and 'fv3_history2d' as specified in 'diag_table' instead of 'dyn' and 'phy'.

Chapter 8

Unified Post Processor (UPP)

8.1 Converting into GRIB2

Ref.) upp.readthedocs.io/en/ufs-v1.0.0/InputsOutputs.html

- On Hera:

```
cd /home/Chan-hoo.Jeon/fv3sar/regional_scripts
vim post.regional_{domain}_{res}.hera
# Check variables
sh post.regional_{domain}_{res}.hera
```

- On Orion:

```
cd /home/chjeon/regional_scripts
vim post.regional_{domain}_{res}.orion
# Check variables
sh post.regional_{domain}_{res}.orion
```

where

No.	Variable name	Description
1	fhr	Hour of the input data (2 digits, 'XX')
2	domain	Domain name
3	RUN	Case name
4	INPUT_DATA	Path to the directory where output files are created

5	PARMfv3	Path to the directory where UPP input files are located

Table 8.1: Variables in the script of UPP.

1. Input files

   (a) 'itag' namelist: created in the script

   - Path and name of the *FV3* **pressure-level** output file ({DIR}/dynf0XXX.nc)
   - Format of the *FV3* output (netcdf, binarynemsio)
   - Format of *UPP* output (grib2)
   - Forecast valid time in *FV3* format (not the model start time but the forecast time desired to be post-processed; YYYY-MM-DD_HH:00:00)
   - Name of the dynamic core
   - Path and name of the *FV3* **surface** output file ({DIR}/phyf0XXX.nc)
   - Name of configuration file ('postxconfig-NT.txt')

   (b) GRIB2 control file: 'postxconfig-NT.txt'

   - GRIB2 output table ('dtcenter.org/sites/default/files/community-code/upp-grib2-table_0.pdf')
     i. BGRD3DXX.tmXX

No.	Field description	Type of level	Short name	nlvl
1	Temperature on model surface	hybrid	t	npz
2	Temperature on pressure surface	isobaricInhPa	t	4
3	Temperature in boundary layer	pressureFromGroundLayer	t	6
4	Dew point temperature in boundary layer	pressureFromGroundLayer	dpt	6
5	Specific humidity on model surface	hybrid	q	npz
6	Specific humidity on pressure surface	isobaricInhPa	q	4
7	Specific humidity in boundary layer	pressureFromGroundLayer	q	6
8	Cloud mixing ratio on model surface	hybrid	clwmr	npz
9	Rain mixing ratio on model surface	hybrid	rwmr	npz
10	Snow mixing ratio on model surface	hybrid	snmr	npz
11	Turbulent kinetic energy on model surface	hybrid	tke	npz

12	U component of wind on model surface	hybrid	u	npz
13	U component of wind on pressure surface	isobaricInhPa	u	4
14	U wind at flight levels	heightAboveSea	u	10
15	U wind in boundary layer	pressureFromGroundLayer	u	6
16	V component of wind on model surface	hybrid	v	npz
17	V component of wind on pressure surface	isobaricInhPa	v	4
18	V wind at flight levels	heightAboveSea	v	10
19	V wind in boundary layer	pressureFromGroundLayer	v	6

Table 8.2: Grib2 fields of 'BGRD3D' produced by Unipost (selected).

ii. BGDAWPXX.tmXX

No.	Field description	Type of level	Short name	nlvl
1	Temperature on pressure surface	isobaricInhPa	t	46
2	Temperature at specific height	heightAboveGround	t	14
3	Temperature in boundary layer	pressureFromGroundLayer	t	6
4	Relative humidity on pressure surface	isobaricInhPa	rh	46
5	Dew point temperature on pressure surface	isobaricInhPa	dpt	46
6	Specific humidity on pressure surface	isobaricInhPa	q	46
7	Specific humidity at specific height	heightAboveGround	q	14
8	Specific humidity in boundary layer	pressureFromGroundLayer	q	6
9	U component of wind on pressure surface	isobaricInhPa	u	46
10	U component of wind at specific height	heightAboveGround	u	14
11	U component of wind in boundary layer	pressureFromGroundLayer	u	6
12	U component of storm motion	heightAboveGroundLayer	ustm	1
13	V component of wind on pressure surface	isobaricInhPa	v	46
14	V component of wind at specific height	heightAboveGround	v	14
15	V component of wind in boundary layer	pressureFromGroundLayer	v	6
16	V component of storm motion	heightAboveGroundLayer	vstm	1
17	Vertical velocity in boundary layer	pressureFromGroundLayer	w	6
18	Absolute vorticity in boundary layer	isobaricInhPa	absv	6

19	Turbulent kinetic energy on pressure surface	isobaricInhPa	tke	46
20	Cloud mixing ratio on pressure surface	isobaricInhPa	clwmr	46
21	Ice water mixing ratio	isobaricInhPa	icmr	46
22	Rain mixing ratio on pressure surface	isobaricInhPa	rwmr	46
23	Snow mixing ratio on pressure surface	isobaricInhPa	snmr	46
24	2M temperature	heightAboveGround	2t	1
25	2M dew point temperature	heightAboveGround	2d	1
26	2M relative humidity	heightAboveGround	2r	1
27	10M *u* component wind	heightAboveGround	10u	1
28	10M *v* component wind	heightAboveGround	10v	1
29	10M wind speed	heightAboveGround	10si	1
30	Total cloud cover	“unknown”	tcc	1
30	Total precipitation	surface	tp	1
31	Storm surface runoff	surface	ssrun	1
32	Sea surface temperature	surface	sst	1
33	Derived radar reflectivity	hybrid	refd	2
34	Maximum/Composite radar reflectivity	“unknown”	refc	1
35	Derived radar reflectivity backscatter from rain	“unknown”	refzr	1
36	Derived radar reflectivity backscatter from ice	“unknown”	refzi	1
37	Derived radar reflectivity	isothermal	refd	1

Table 8.3: Grib2 fields ‘BGDAWP’ produced by Unipost (selected).

In the above tables, ‘nlvl’ is the total number of layers, and ‘npz’ is the number of vertical layers in a case.

(c) Look-up table: ‘eta_micro_lookup.dat’

(d) Coefficient files for satellite

2. Regridding

If necessary, the *UPP* output can be interpolated onto a different grid with *wgrib2*.

(a) General Format:

- Latitude-longitude grid

```
-new_grid latlon lon0:nlon:dlon lat0:nlat:dlat outfile
```

where

No.	Variable	Description	Format
1	lon0	Longitude of the first grid point	In degree
2	nlon	Total number of grid points in the longitudinal direction	Integer
3	dlon	Longitudinal distance between two adjacent grid points	Float
4	lat0	Latitude of the first grid poin	In degree
5	nlat	Total number of grid points in the latitudinal direction	Integer
6	dlat	Latitudinal distance between two adjacent grid points	Float

Table 8.4: Variables for a longitude-latitude grid.

- Lambert conic grid

```
-new_grid lambert:lov:latin1:latin2 lon0:nx:dx lat0:ny:dy outfile
```

where

No.	Variable	Description	Format
1	lov	Longitude where y axis is parallel to meridian	In degrees
2	latin1	First latitude from pole which cuts the secant cone	In degrees
3	latin2	Second latitude from pole which cuts the secant cone	In degrees
4	lon0	Longitude of the first grid point	In degrees
5	lat0	Latitude of the first grid point	In degrees
6	nx	Total number of grid points along x	Integer
7	ny	Total number of grid points along y	Integer
8	dx	Longitudinal distance between two adjacent grid points	In meters

9	dy	Latitudinal distance between two adjacent grid points	In meters

Table 8.5: Variables for Lambert conic grid.

- Polar stereographic grid

```
-new_grid nps(or sps):lov:lad lon0:nx:dx lat0:ny:dy outfile
```

where

No.	Variable	Description	Format
1	nps / sps	North / South polar stereographic	
2	lov	Longitude where y axis is parallel to meridian	In degrees
3	lad	Latitude where dx and dy are specified	60(nps) / -60(sps)
4	lon0	Longitude of the first grid point	In degrees
5	lat0	Latitude of the first grid point	In degrees
6	nx	Total number of grid points along x	Integer
7	ny	Total number of grid points along y	Integer
8	dx	Longitudinal distance at 'lad'	In meters
9	dy	Latitudinal distance at 'lad'	In meters

Table 8.6: Variables for Polar stereographic grid.

(b) Winds:

The wind directions are defined in advance for interpolation.

```
-new_grid_winds grid(or earth)
```

where

No.	Option	Description
1	grid	u-wind goes from grid (i, j) to $(i+1, j)$
2	earth	u- and v-winds go eastward and northward, respectively

Table 8.7: Option for wind directions.

(c) Interpolation:

The default interpolation is bilinear, but it can be set by '-new_grid_interpolation'.

```
-new_grid_interpolation neighbor
```

3. Output files

No.	Output file	Transferred name
1	BGDAWP{fhr}.{tmmark}	{RUN}.tz.{domain}.natprs.f{fhr}.{tmmark}.grib2
2	BGRD3D{fhr}.{tmmark}	{RUN}.tz.{domain}.natlev.f{fhr}.{tmmark}.grib2

Table 8.8: Output files of UPP.

Chapter 9

Code Development

9.1 Compiling the Source Code of *FV3* after Modification

Once the code is modified, it should be compiled again:

```
cd ~/ufs-weather-model/tests/
sh compile.sh ${PATHTR} ${BUILD_TARGET} ${MAKE_OPT} ${BUILD_NAME} ${clean_before}
     ${clean_after} >& log.txt
```

where

No.	Variable name	Description
1	PATHTR	Path to the FV3 source code
2	BUILD_TARGET	Machine specific name
3	MAKE_OPT	Compile option for 32 bit (“32BIT=Y”)
4	BUILD_NAME	Sub-name of the executable (‘fv3_(?).exe’)
5	clean_before	Flag for cleaning the entire code before compiling (YES or NO)
6	clean_after	Flag for cleaning the entire code after the compilation is done (YES or NO)

Table 9.1: Variables for compiling.

- On Hera:

```
sh compile.sh ../FV3 hera.intel "32BIT=Y" 32bit NO NO >& log.txt
```

- On WCOSS:

```
sh compile.sh ../FV3 wcoss_dell_p3 "32BIT=Y" 32bit NO NO >& log.txt
```

Above will produce the executable called *fv3_32bit.exe* in the 'tests' directory. Copy it to the work directory where a case runs:

```
cp fv3_32bit.exe $WORKDIR
```

9.2 Initial and Lateral Boundary Fields

Almost all modification for regional IC/LBC can be found at

```
cd ~/ufs-weather-model/FV3/atmos_cubed_sphere/model/
vim fv_regional_bc.F90
```

Sometimes it happens at

```
cd ~/ufs-weather-model/FV3/atmos_cubed_sphere/tools/
vim external_ic.F90
```

9.3 Near-boundary Blending

The blending works by not only generating data for the boundary region but by inserting additional 'boundary rows' into the BC files that overlap the outermost rows of the integration domain. The *chgres* code in 'UFS_UTILS' can add those extra rows in the boundary files. When running the forecast set the 'input.nml' variable 'nrows_blend' to the number of rows you want to blend within the outer rows of the integration domain.

9.3.1 Blending in a Forecast Run

1. Check that *FV3*, which is installed in your system, supports the 'blending' option:

   ```
   cd ~/ufs-weather-model/FV3/atmos_cubed_sphere/model/
   grep -n -i blend fv_control.F90 fv_arrays.F90
   ```

2. Create IC/LBC fields including blending layers (Section 6).

- Run 'chgres_cube' with 'halo_blend':

```
regional = 1 (or 2)
halo_bndy =4
halo_blend = 10
```

3. Turn on the blending option in the configuration 'input.nml' (Section **??**).

```
(under '&fv_core_nml')
    regional_bcs_from_gsi = .false.   (for DA blending)
    write_restart_with_bcs = .false.   (for DA blending)
    nrows_blend = 10
```

Note) Make sure that 'halo_blend' is equal to 'nrows_blend'.

9.3.2 Blending in a DA Run

1. Two variables in 'input.nml' need to be used: (1) write_restart_with_bcs (.true. / .false.), and (2) regional_bcs_from_gsi (.true. / .false.).

   - 'write_restart_with_bcs':
     Typically the model writes out restart files that contain only the integration domain in which case set this flag 'false'. However, in order to include the boundaries in the assimilation, those boundary rows should be included in the restart files so set the first flag to 'true'. This will make restart files write the boundary rows in addition to writing out normal core and tracers.
   - 'regional_bcs_from_gsi':
     The second flag says to treat all the BC files in the usual way when it is 'false'. The 'usual' means that the variables are remapped in the vertical from the external forecast's layer distribution to the forecast's, and the winds are rotated from geographic lat/lon to the forecast grids' orientation. When the assimilation writes out restart files, though the fields put into the new post-GSI, BC files are already on the proper levels and are oriented properly so the model is told to **NOT** remap or rotate by setting this flag to 'true'.

2. To create and write the new larger restart files, which include the blending boundary layers, for the GSI, the code needs original restart files to get all the specifications and field names in the '/INPUT' subdirectory for NetCDF:

(a) Copy any normal-sized 'fv_core.res.tile1.nc' and 'fv_tracer.res.tile1.nc' into the '/INPUT' subdirectory before starting the forecast, basically as templates.

Note) Make sure that all three dimensions in those template restart files are the values you want to use.

(b) Change their names from 'tile1' to 'temp'. Otherwise, FMS will get confused.

(c) The forecast's normal and enlarged restart files will show up in the '/RESTART' subdirectory of the working directory as usual. The enlarged restart file names will end with '.tile1_new.nc'.

3. After your run writes out the two enlarged restart files for the data assimilation, you also need 'sfc_data.nc' and 'grid_spec.nc' files that have been similarly enlarged. To get them, run the following serial code:

```
prep_for_regional_DA.F90
```

- Input files:

No.	Original name	Transferred name	Description
1	sfc_data.tile7.nc	sfc_data_orig.nc	Output of IC
2	grid_spec.nc	grid_spec_orig.nc	Output of a forecast run
3	grid.tile7.halo3.nc	grid.tile7.halo3.nc	Output of 'grid_driver' (pre-processing)

Table 9.2: Input files for enlarged restart files

- Output files:

No.	Original name	Transferred name
1	sfc_data_new.nc	sfc_data.nc
2	grid_spec_new.nc	grid_spec.nc

Table 9.3: Output files for enlarged restart files

Note) This job assumes that the original smaller files have been renamed 'sfc_data_orig.nc' and 'grid_spec_orig.nc' and it then generates the larger files called 'sfc_data_new.nc' and 'grid_spec_new.nc'.

4. The data assimilation runs on the enlarged core and tracers restart files and the enlarged 'sfc_data.nc' and 'grid_spec.nc' files. When it is finished, the updated data must be put back into the files the model will read in the forecast.

```
move_DA_update_data.F90
```

- Input files:

No.	File name	Description
1	fv_core.res.tile1_new.nc	Enlarged core restart file
2	fv_tracer.res.tile1_new.nc	Enlarged tracers restart file
3	gfs_bndy.tile7.HHH.nc	Original BC file valid at the time (HHH) of the restart (no DA changes)

Table 9.4: Input files for updated BC files

- Run:

  Currently the move program takes 'HHH' as a command line argument. i.e. if the BC file is 'gfs_bndy.tile7.000.nc' then to execute the program:

```
./a.out 000
```

  When the code executes it will copy all the interior data from the updated enlarged restart files back into the original-sized restart files that were written prior to running the data assimilation (both normal and enlarged restart files are written when 'write_restart_with_bcs = .true.'). And a new BC file is generated called 'gfs_bndy.tile7.HHH_gsi.nc' that is taken from the boundary region of the updated enlarged restart files and it has a different structure from the original BC files.

- Output files:

No.	File name	Description
1	gfs_bndy.tile7.HHH_gsi.nc	

Table 9.5: Output files for updated BC files

5. When you start up the next free forecast following the DA update set:

```
regional_bcs_from_gsi = .true.
```

So the model will use 'gfs_bndy.tile7.HHH_gsi.nc' as the starting BC file (with no vertical remap or wind rotation) but then will read the following BC file valid at the end of your boundary update interval with its original name (no '_gsi' in it) and will do the remap and wind rotation as usual.

9.3.3 Modification of the source code

The code can be cloned from GitHUB:

```
git clone https://github.com/TomBlack-NOAA/GFDL_atmos_cubed_sphere.git -b tracers
```

The files under 'atmos_cubed_sphere' that are changed for the blending and for including the regional boundary in the data assimilation are shown in Table 9.1. Here are two versions of two files needed to switch between the NAM's 60-layer vertical structure and the 65 layers.

1. For 65 layers:

```
cp external_ic.F90_65lyrs external_ic.F90
cp fv_eta.F90_65lyrs fv_eta.F90
```

2. For NAM's 60 layers:

```
cp external_ic.F90_NAM_lyrs external_ic.F90
cp fv_eta.F90_NAM_lyrs fv_eta.F90
```

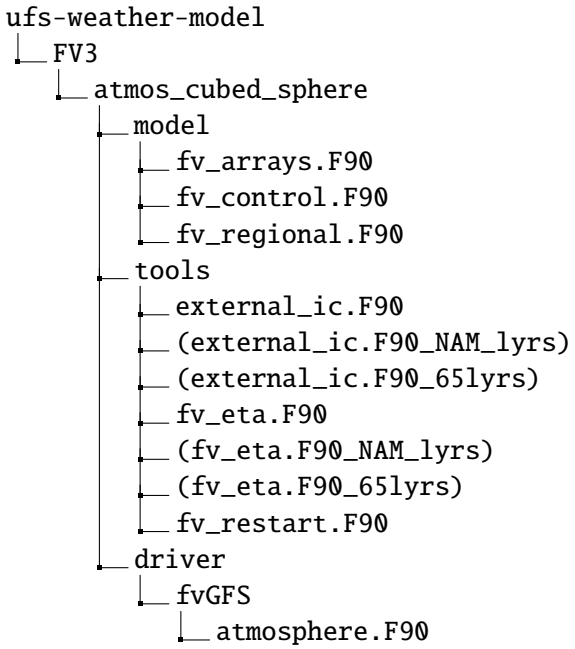

Figure 9.1: Structure of directory

9.4 *chgres_cube*

9.4.1 Compiling the Program

chgres_cube requires cmake 3.12 or higher. It can be built as part of the NCEPLIBS unified build system.

1. Install NCEPLIBS:

2. Compile 'chgres_cube':

```
# Check paths to hdf5, compiler, mpi, and cmake
cd (path to NCEPLIBS)
source bin/setenv_nceplibs.sh
# Set cmake compiler: -export FC=ifort
cd (where you checked out UFS_UTILS)
mkdir build
cd build
```

```
cmake .. -DCMAKE_INSTALL_PREFIX=(path to the directory where the code will
    be installed)
make -j 8
make install
```

9.4.2 Running the Program Stand Alone

1. Locate input files:

   - Input files:
     (a) Grid file: C{res}_grid.tile7.nc
     (b) Mosaic file: C{res}_mosaic.nc
     (c) Orography file: C{res}_oro_data.tile7.nc
     (d) Surface climatology files (in the sub-directory 'fix_sfc):
         i. C{res}.facsf.tile7.nc (fractional coverage for strong/weak zenith angle dependent albedo)
         ii. C{res}.maximum_snow_albedo.tile7.nc (maximum snow albedo)
         iii. C{res}.slope_type.tile7.nc (slope type)
         iv. C{res}.snowfree_albedo.tile7.nc (snow-free albedo)
         v. C{res}.soil_type.tile7.nc (soil type)
         vi. C{res}.substrate_temperature.tile7.nc (soil substrate temperature)
         vii. C{res}.vegetation_greenness.tile7.nc (vegetation greenness)
         viii. C{res}.vegetation_type.tile7.nc (vegetation type)
     (e) Vertical coordinate file: global_hyblev.l{LEVS}.txt
     (f) GFS data

2. Set the namelist for the experiment:

   See Section 6.3

3. Link the namelist to Fortran unit number 41:

```
ln -fs your-namelist-file ./fort.41
```

4. Load required runtime libraries:

- On Hera:

```
module purge
module load intel
module load impi
module load netcdf
module list
```

5. Run the program with an MPI task count that is a multiple of six. This is an ESMF library requirement when processing a six-tiled global grid.

```
time srun -l ./chgres_cube.exe
```

6. Check the output files:

   (a) out.atm.tile1.nc: Atmospheric 'cold start' file

   (b) out.sfc.tile1.nc: Surface/NSST(Near Surface Temperature) 'cold start' file

   (c) gfs_bndy.nc: Lateral boundary condition files

9.4.3 Code Structure

1. 'chgres.F90': Main driver routine
2. 'program_setup.F90'
   - Reads program namelist.
   - Computes soil parameters
   - Reads the variable mapping (VARMAP) table.
3. 'model_grid.F90'
   - Sets up the ESMF grid objects for the input data grid and target FV3 grid.
4. 'static_data.F90'
   - Read static surface climatological data for the target FV3 grid.
5. 'write_data.F90'

- Writes the tiled and header files.

6. 'input_data.F90'
   - Contains routines to read atmospheric and surface data from GRIB2 and NEMSIO files.
7. 'utils.f90'
   - Contains utility routines such as error handling.
8. 'grib2_util.F90'
   - Converts from RH to specific humidity.
   - Converts from omega to dzdt which is required for GRIB2 input data.
9. 'atmosphere.F90'
   - Processes atmospheric fields.
   - Horizontally interpolates from input to the target FV3 grid using ESMF regridding.
   - Adjust surface pressure according to terrain differences between input and target grids.
   - Vertically interpolates to the vertical levels on the target FV3 grid.
   - Main routines:
     (a) 'read_vcoord_info': Reads a vertical coordinate definition file.
     (b) 'newps': Computes adjusted surface pressure at new terrain heights.
     (c) 'newpr1': Computes 3-dimensional pressure at the adjusted surface pressure.
     (d) 'vintg': Vertically interpolates atmospheric fields to the target FV3 grid.
10. 'surface.F90'
    - Processes land, sea/lake ice, open water/NSST (Near Sea Surface Temperature) fields.
    - Assumes the FV3 run will use the Noah LSM-based land data as input.
    - NSST fields are not available when using GRIB2 input data.
    - Main routines:
      (a) 'interp': Horizontally interpolates fields from the input to target FV3 grids.
      (b) 'calc_liq_soil_moisture': Computes liquid portion of the total soil moisture.

(c) ‘adjust_soilt_for_terrain’: Adjust soil temperature for large differences between the input and target FV3 grids.

(d) ‘rescale_soil_moisture’: Adjust the total soil moisture for differences between soil types on the input and target FV3 grids.

(e) ‘roughness’: Sets roughness length at land and sea/lake ice. At land, a vegetation type-based lookup table is used.

(f) ‘qc_check’: Checks consistency.

11. ‘search_util.f90’

- Searches for the nearest valid land/non-land data where the input and target FV3 land-mask differ.

Chapter 10

Supporting Tools

10.1 Grid Coordinates: *fv3grid*

The coordinates of global (parent) and regional (child, nest) grids can be depicted by *fv3grid*. This tool is useful to check the location of a regional domain on the global domain as well as on the rotated domain.

10.1.1 Regional Domain

Parameters for creating a new regional domain can be found as:

1. Pop up a display window of *fv3grid*:

   - On Hera

   ```
   /scratch2/NCEPDEV/fv3-cam/Dusan.Jovic/dbrowse/fv3grid
   ```

   Note) To pop up a display window on the terminal, 'ssh' should accompany with '-Y' as

   ```
   'ssh -Y Chan-hoo.Jeon@hera-rsa.rdhpcs.noaa.gov'.
   ```

   Note) If your local machine is Mac, you should install 'XQuartz' in advance to pop up a display window.

2. Set the parameters for Tile 6 of the global grid:

No.	Variable name	Description
1	resolution C	Grid resolution
2	stretch factor	Stretching factor for the global grid
3	target lon	Center longitude of Tile 6 (in degrees)
4	target lat	Center latitude of Tile 6 (in degrees)

Table 10.1: Variables for a global grid.

Note) It is recommended that the 'target lon' and 'target lat' have the same values with the center of the regional domain for convenience.

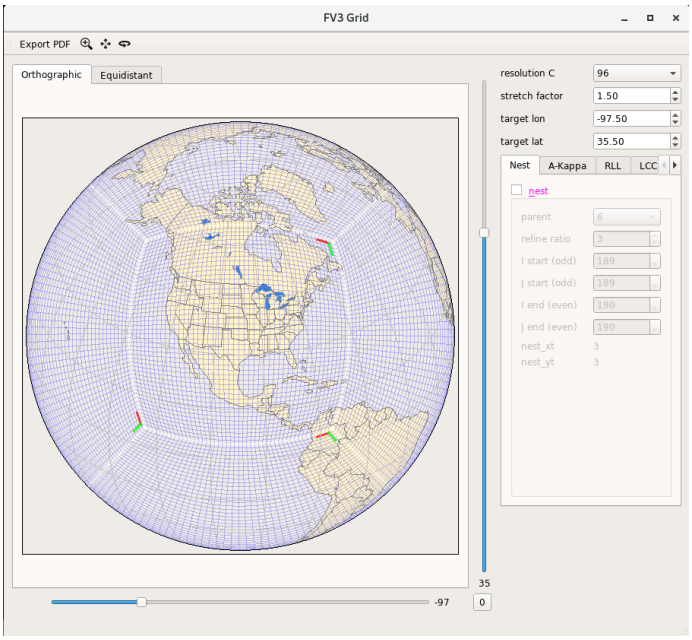

Figure 10.1: Change of resolution and stretch factor

3. Check the box of 'nest' for a regional grid, and set the parameters:

No.	Variable name	Description
1	parent	Number of the parent tile where the regional grid is located (=6)
2	refine ratio	Grid refinement ratio

3	I start (odd)	Lowest *i*-index of the parent-tile super-grid corresponding to the regional domain
4	J start (odd)	Lowest *j*-index of the parent-tile super-grid corresponding to the regional domain
5	I end (even)	Highest *i*-index of the parent-tile super-grid corresponding to the regional domain
6	J end (even)	Highest *j*-index of the parent-tile super-grid corresponding to the regional domain

Table 10.2: Variables for a regional grid.

Notes)

- To make the domain symmetry along the centerline, 'I end'=2×'resolution C'-('I start'-1), and 'J end'=2×'resolution C'-('J start'-1).
- The 'resolution C' means that the size of the parent tile 6 is {2×'resolution C'}×{2×'resolution C'} for the *fv3* super-grid or {'resolution C'}×{'resolution C'} for a general grid.

10.1.2 Rotated Domain for 'output_grid'

Parameters for plotting extent in 'model_configure' can be found as:

1. Pop up a display window of *fv3grid*:
   - On Hera:

     ```
     /scratch2/NCEPDEV/fv3-cam/Dusan.Jovic/dbrowse/fv3grid
     ```

2. Set the parameters for Tile 6 in the global and regional domains as shown in Section 10.1.1:
3. Click the 'RLL' tab on the right of the window, and check the box of 'Rotated Lat Lon':
4. Adjust the parameters to put the blue box inside the regional domain in purple:

No.	Variable name	Description
1	tlm0d	Longitude of the center of the rotated domain (in degrees)
2	tph0d	Latitude of the center of the rotated domain (in degrees)
3	wbd	Longitudinal distance from the center to the bottom-left corner of the rotated domain (in degrees)
4	sbd	Latitudinal distance from the center to the bottom-left corner of the rotated domain (in degrees)

Table 10.3: Variables for a RLL grid.

Note) 'tlm0d' and 'tph0d' do not need to be the same as 'target_lon' and 'target_lat', respectively. However, it is recommended to make them the same each other.

Note) The top-right corner of the rotated domain (box in blue) is automatically set as the same distance from the center as 'wbd' and 'sbd'.

10.1.3 A–Kappa (α–κ) Domain for an ESG (JP) Grid

1. Pop up a display window of *fv3grid*:

   - On Hera:

   ```
   /scratch2/NCEPDEV/fv3-cam/Dusan.Jovic/dbrowse/fv3grid
   ```

2. Click the 'nest' check box in the 'Nest' tap, and set up the GFDL domain for comparison if necessary:

3. Click the 'A-Kappa' check box in the 'A-Kappa' tap, and adjust the grid parameters for an ESG (JP) grid:

10.2 Plotting with *Python*

10.2.1 'Natural Earth' Data for Background

Download shape files:

```
www.naturalearthdata.com/downloads/
```

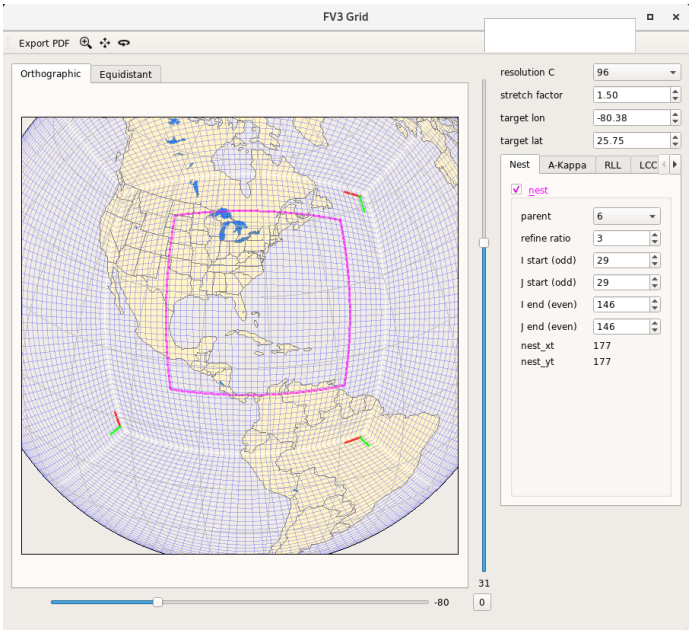

Figure 10.2: New regional region of interest

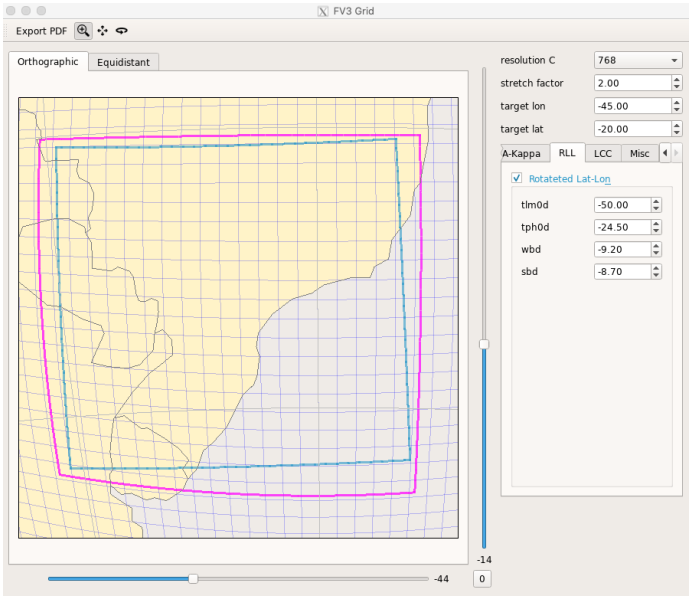

Figure 10.3: Parameters on the rotated grid

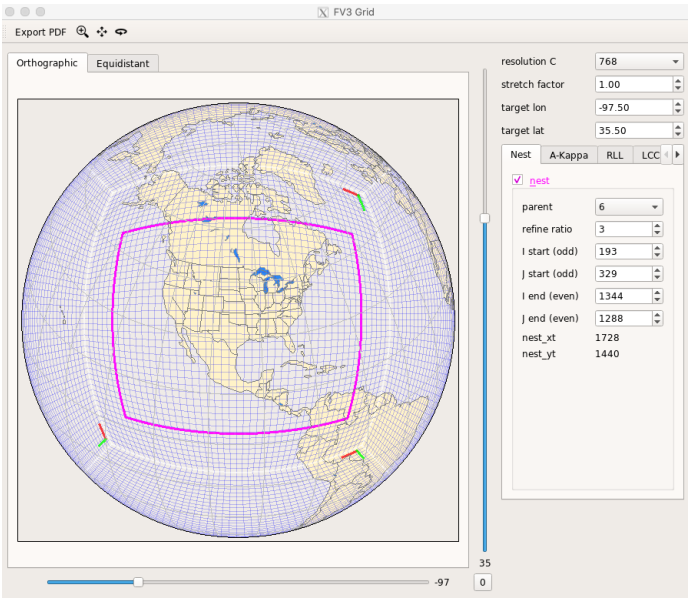

Figure 10.4: Parameters of a GFDL grid for camparison

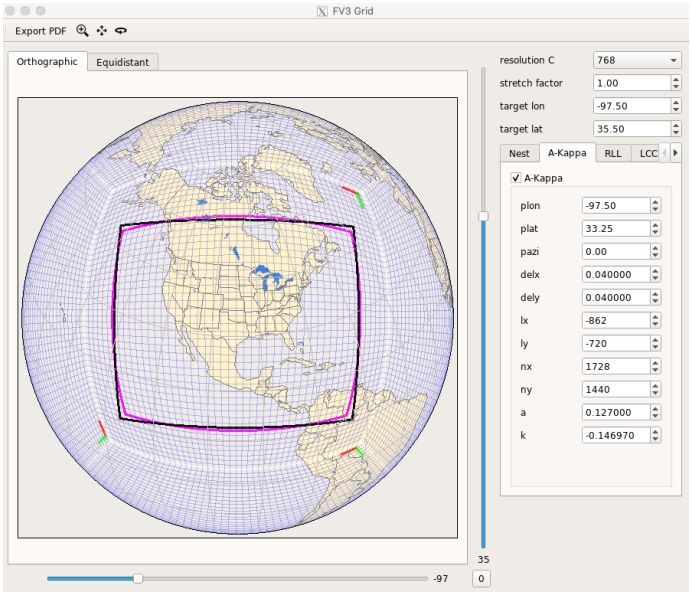

Figure 10.5: Parameters for an ESG (JP) grid

The default scale (resolution) of background attributes built in *python* is 1:110m. This is good enough for the global domain. However, the medium scale (1:50m) or large scale (1:10m) data are

recommended for a regional domain as shown in Figure 10.6.

- For 'Cultural':

```
www.naturalearthdata.com/downloads/50m-cultural-vectors/
# Click 'Download all 50m cultural thememes'
```

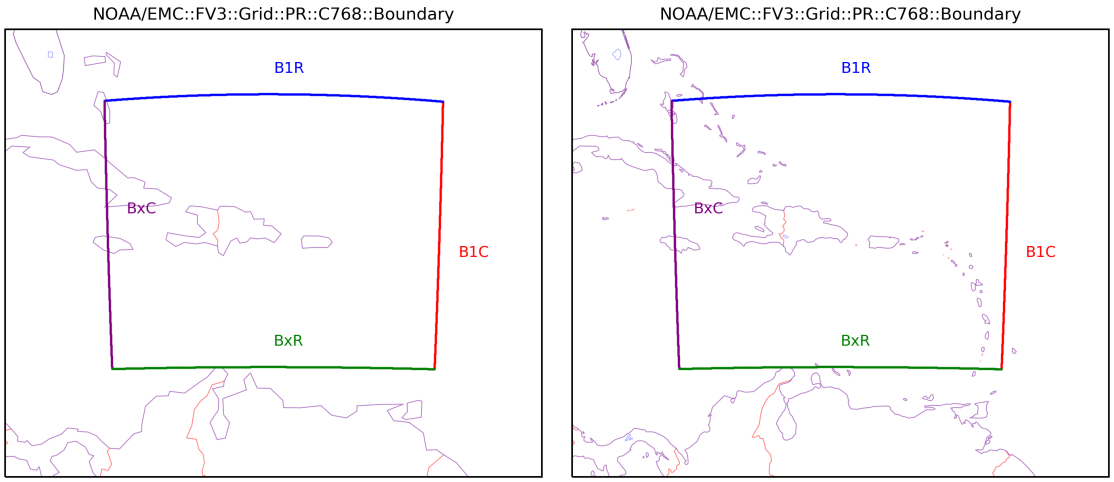

Figure 10.6: Background scales: 1/110m vs. 1/50m

- For 'Physical':

```
www.naturalearthdata.com/downloads/50m-physical-vectors/
# Click 'download all 50m physical themes'
```

Put the files into the specific structures of directories as below:

```
$data_dir/shapefiles/natural_earth/physical (or cultural)
```

Or, the files can be found as follows:

- On Hera:

```
/scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/tools/NaturalEarth/
```

- On Orion:

```
/home/chjeon/tools/NaturalEarth/
```

- On WCOSS:

```
/u/Chan-Hoo.Jeon/tools/NaturalEarth/
```

Set the path in the script:

- On Hera:

```
cartopy.config['data_dir']='/scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/tools/
    NaturalEarth'
```

- On Orion:

```
cartopy.config['data_dir']='/home/chjeon/tools/NaturalEarth'
```

- On WCOSS:

```
cartopy.config['data_dir']='/u/Chan-Hoo.Jeon/tools/NaturalEarth'
```

10.2.2 Cartopy Background Image

Cartopy provides the 'background_img()' method to add background images in a convenient way. You can specify custom background images.

1. Download background images such as Natural Earth data and put them in the specific directory.

2. Set up the environment variable for the path of the directory that contains the background images:

```
{
os.environ["CARTOPY_USER_BACKGROUNDS"]="/scratch2/NCEPDEV/fv3-cam/Chan-hoo.
    Jeon/tools/NaturalEarth/raster_files"
}
```

3. The arguments of 'name' and 'resolution' for the 'background_img()' are specified in the configuration file 'images.json' that should be located in the above directory. An example of the 'images.json' file is as follows:

```
{
    "NE": {
    "__comment__": "Natural␣Earth␣raster␣file",
    "__source__": "www.naturalearthdata.com/downloads",
    "__projection__": "PlateCarree",
    "high": "NE1_50M_SR_W.tif"}
}
```

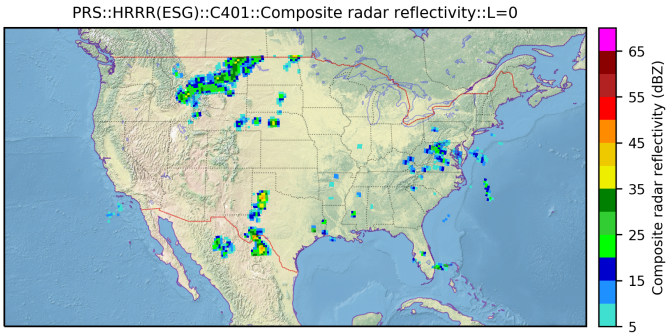

Figure 10.7: Background image: Natural Earth data

4. Call 'background_img()' in the script as follows:

```
ax.background_img(name='NE', resolution='high')
```

Note) A low-resolution background image can be easily called by 'stock_img()' in a script.

10.2.3 Modules

If a HPC, such as Hera, provides a module of 'Anaconda/Miniconda', you can load modules for running python scripts as follows:

- On Hera:

```
module load intel
module use -a /contrib/anaconda/modulefiles
module load anaconda/latest
```

Note) If 'Miniconda (or Anaconda)' is installed in the home directory, you do not need to load the above modules for running post-processing python scripts.

10.2.4 Installation of Miniconda (Anaconda)

If a HPC, such as Orion, does not provide a module of 'Anaconda' for python packages, you need to install either 'Miniconda' or 'Anaconda' in your home directory.

1. Download the installer from the Miniconda repository:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

2. Verify the installer hashes:

```
sha256sum Miniconda3-latest-Linux-x86_64.sh
# Compare the hash code to that on the website ('https://conda.io/projects/
    conda/en/latest/user-guide/install/linux.html')
```

3. Install Miniconda:

```
bash Miniconda3-latest-Linux-x86_64.sh
(enter)
(space bar)
(yes)
(enter)
(enter)
```

4. Update the path

```
vim ~/.bashrc
# insert
export PATH=/home/$LOGNAME/miniconda3/bin:$PATH
# close the '.bashrc' file
source ~/.bashrc
```

5. Test the installation

```
conda list
```

6. Install the packages required for running the post-processing python scripts:

```
conda install cartopy
conda install xarray
conda install netcdf4
conda install dask
conda install -c conda-forge pygrib
```

10.2.5 Python Scripts on GitHub

The python scripts described in this section can be cloned from GitHub as:

```
git clone https://github.com/chan-hoo/FV3LAM_plot_python.git
```

The list of the scripts in the above repository is as follows:

No.	Script name	Description
1	plot_fv3lam_grid_oro.py	Grid and orography files
2	plot_fv3lam_grid.py	Grid file alone
3	plot_fv3lam_static.py	Regional static field files
4	plot_fv3lam_fixam.py	Global static field files
5	plot_fv3lam_icbc.py	Time-dependent IC/LBC fields
6	plot_fv3lam_icsfc.py	Initial surface climatology fields
7	plot_fv3lam_co2his.py	Historical CO_2 data
8	plot_fv3lam_gridspec.py	Output 'grid_spec.nc' file
9	plot_fv3lam_atmstat.py	Output 'atmos_static.nc' file
10	plot_fv3lam_dynf.py	Output 'dynffXXX.nc' file
11	plot_fv3lam_phyf.py	Output 'phyfXXX.nc' file
12	plot_fv3lam_bgrd3d.py	Output 'BGRD3D' GRIB2 file
13	plot_fv3lam_bgdawp.py	Output 'BGDAWP' GRIB2 file
14	plot_fv3lam_comp2f.py	Comparison of two NetCDF (GRIB2) files
15	plot_fv3lam_mrms.py	MRMS radar data for reflectivity
16	plot_fv3lam_ani_comref.py	Animation of hourly comparison of GRIB2 and radar data

Table 10.4: Python scripts for plotting input and output files.

10.2.6 Grid and Orography

The python script 'plot_fv3sar_grid_oro.py' creates figures for grid and orography in a regional domain. The script is based on Python 3.7, and Python libraries of 'matplotlib', 'xarray', and 'cartopy' are applied.

1. Input files
   - C{res}_gird.tile7.halo4.nc (grid)
   - C{res}_oro_data.tile7.halo4.nc (orography)
2. Output files
   - fv3_grid_{domain}_C{res}.png (grid)
   - fv3_grid_{domain}_C{res}_bndr.png (boundary of domain)
   - fv3_grid_{domain}_C{res}_dxy.png (cell size)
   - fv3_grid_{domain}_C{res}_crnr_RXCX.png (grid at the four corners)
   - fv3_orog_{domain}_C{res}_{variables}.png (orography variables)
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_grid_oro.py
# Run the script
python3 plot_fv3sar_grid_oro.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input NetCDF files are located
3	domain	Domain name
4	res	Grid resolution (C+resolution)
5	gtype	Grid type ('ESG', 'GFDL')
6	refine	Refinement ratio
7	fnm_in_base	Basic form of the input NetCDF files
8	n_skip	Number of grid points in lows/column skipped for plotting

9	out_fig_dir	Path to the directory where output files are created
10	out_[*]_title	Title of the figures
11	out_[*]_fname	Output file name of the figures
12	back_res	Resolution of background natural earth data (‘50m’ or ‘110m’)

Table 10.5: Namelist in the script for plotting gird and orography.

4. Output:
   For example, the following files for the CONUS (C96) domain are created in the output directory:

   (a) Grid points for every ‘n_skip’ rows/columns (‘fv3_grid_CONUS_C96.png’)

   - For a clear view, grid points are shown at every ‘n_skip’-th row and column from ($i = 2, j = 2$) for the super-grid of *fv3* in red, and ($i = 1, j = 1$) for the typical grid such as the grid of orography in green. This is because the super-grid includes not only a grid point at the center of a cell but also grid points at the corner of a cell.
   - Default values of ‘n_skip’: 10 for ‘CONUS’, 50 for ‘AK’, 20 for ‘PR’, and 50 for others.

   (b) Grid points along the boundary (‘fv3_gird_CONUS_C96_bndr.png’)

   - This figure only shows the grid points along the four boundaries.
     i. B1C: Along the first column ($j = 1$) in red
     ii. B1R: Along the first row ($i = 1$) in blue
     iii. BxC: Along the last column ($j = x$) in purple
     iv. BxR: Along the last row ($i = x$) in green

   (c) Grid cell sizes (‘fv3_grid_CONUS_C96_dxy.png’)

   (d) Grid points at the four corners (‘fv3_grid_CONUS_C96_crnr_[X].png’)

   - Each corner has a separate plot: R1C1, R1Cx, RxC1, and RxCx.
     i. R1C1: Around the corner ($i = 1, j = 1$), intersection between B1R and B1C
     ii. R1Cx: Around the corner ($i = 1, j = x$), intersection between B1R and BxC
     iii. RxC1: Around the corner ($i = x, j = 1$), intersection between BxR and B1C
     iv. RxCx: Around the corner ($i = x, j = x$), intersection between BxR and BxC

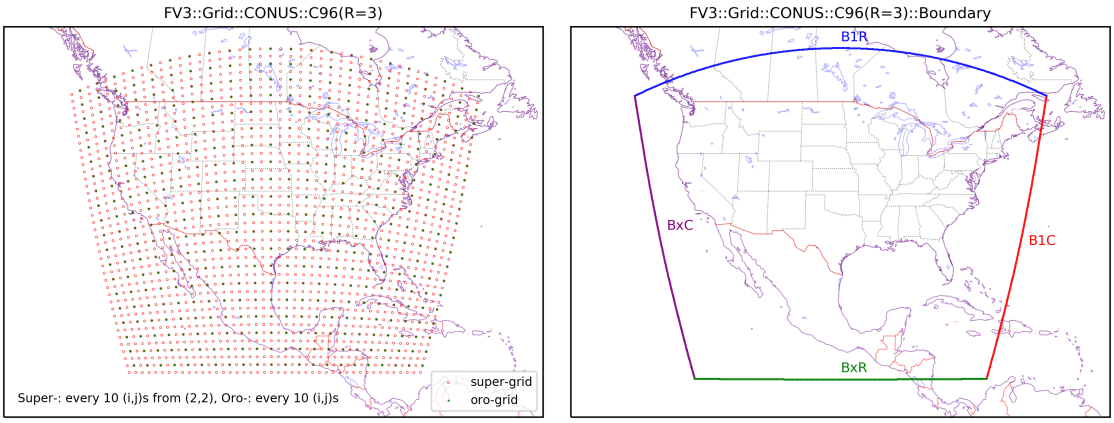

Figure 10.8: Grid points at every 10 rows and columns, and along boundaries

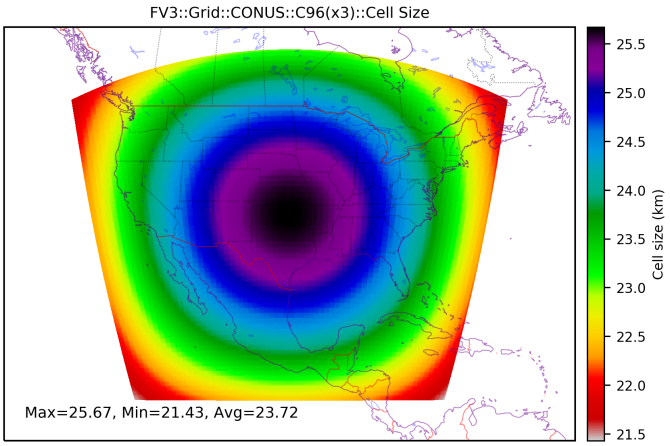

Figure 10.9: Cell size

- Each figure shows 30 rows/columns by the default. It can be adjusted by modifying the value of 'N_crn' in the script.
- They plot both the *fv3* super-grid and the typical grid ('oro-grid' on the figures).

(e) Orography (raw and filtered)

(f) Sea-land mask (sea:0, land:1), and land fraction

(g) Asymmetry and convexity parameters

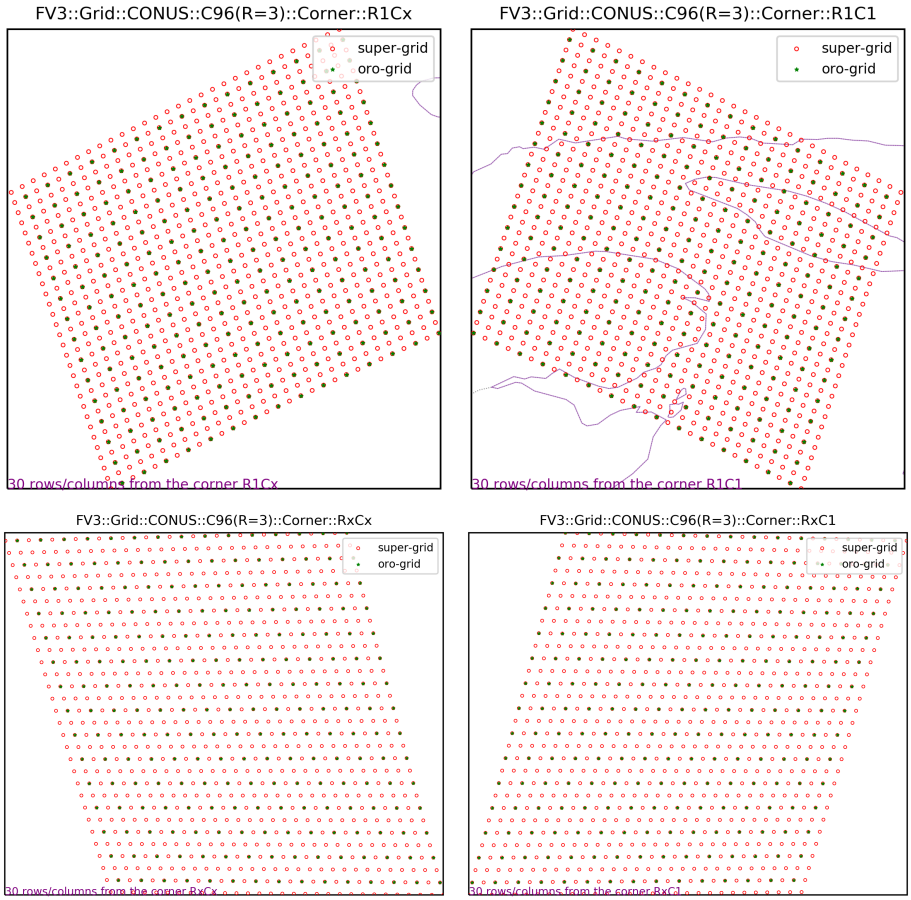

Figure 10.10: Grid points at four corners

10.2.7 Static Fields: Regional ('fix_sar')

The python script 'plot_fv3sar_static.py' creates figures for static fields in a regional domain as well as the original (global) data for comparison. The required files can be found in the namelist as described in Table 2.47.

1. Input files

   - C{res}.{variable}.tile7.halo[0 or 4].nc
   - C{res}_oro_data.tile7.halo[0 or 4].nc (orography file for coordinates)
   - Original global static field files

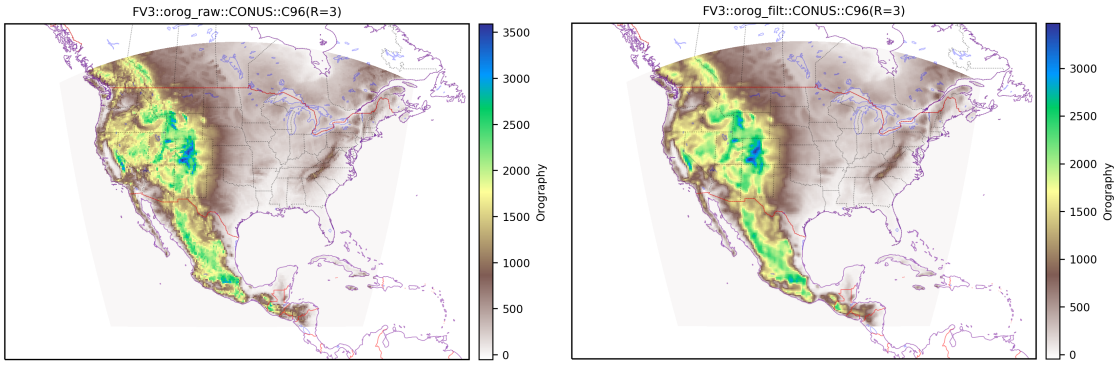

Figure 10.11: Orography

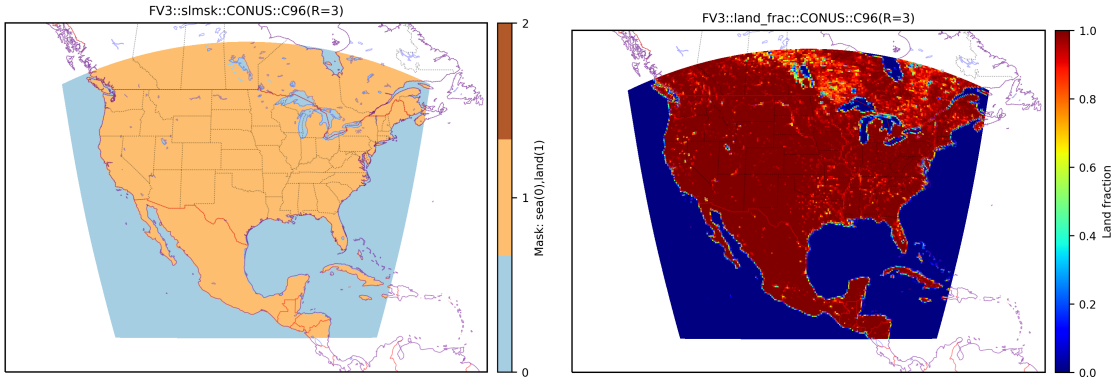

Figure 10.12: Sea-land mask, and land fraction

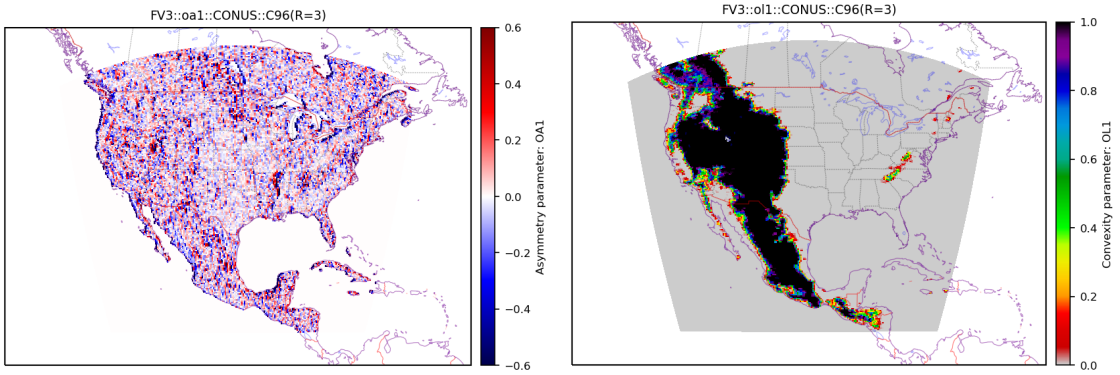

Figure 10.13: Asymmetry and convexity parameters

Note) The halo number ('0' or '4') of the orography file must correspond to that of the variable file.

2. Output files

   - fv3_static_{domain}_C{res}_{variable}.png

3. Run on HPC

   - On Hera:

```
# Check Input and Output paths and names
vim plot_fv3lam_static.py
# Interactive mode on HPC
salloc --x11=first -q debug -t 0:30:00 --nodes=1 -A fv3-cam
# Run the script
python3 plot_fv3lam_static.py
```

   - On Orion:

```
# Check Input and Output paths and names
vim plot_fv3lam_static.py
# Interactive mode on HPC
salloc --ntasks=1 --time=00:30:00
# Run the script
python3 plot_fv3lam_static.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input NetCDF files are located
3	prt_data	Path to the original (global) data files
4	domain	Domain name
5	res	Resolution (C+resolution)
6	gtype	Grid type ('ESG', 'GFDL')
7	refine	Refinement ratio
8	fnm_in_base	Basic form of the input NetCDF files
9	vars_static	Specific field name of the input files

10	out_fig_dir	Path to the directory where output files are created
11	out_title_base	Basic form of the title in the figures
12	out_fname_base	Basic form of the output file name for the fields
13	back_res	Resolution of the background plot

Table 10.6: Namelist in the script for plotting static fields.

- **Note)** Since the NetCDF files for the static fields do not contain the coordinates, the longitudes and latitudes of the grid points are imported from the orography file.
- **Note)** In case of time-limit on the system such as an interactive job on Hera (limit=30min), split 'vars_static' into two: 'snowfree_albedo' and others. Moreover, the variables in 'snowfree_albedo' should be split into two in the function of 'static_plot': 'visible' and 'near_IR'.

4. Output:

   (a) Maximum snow albedo, and substrate temperature

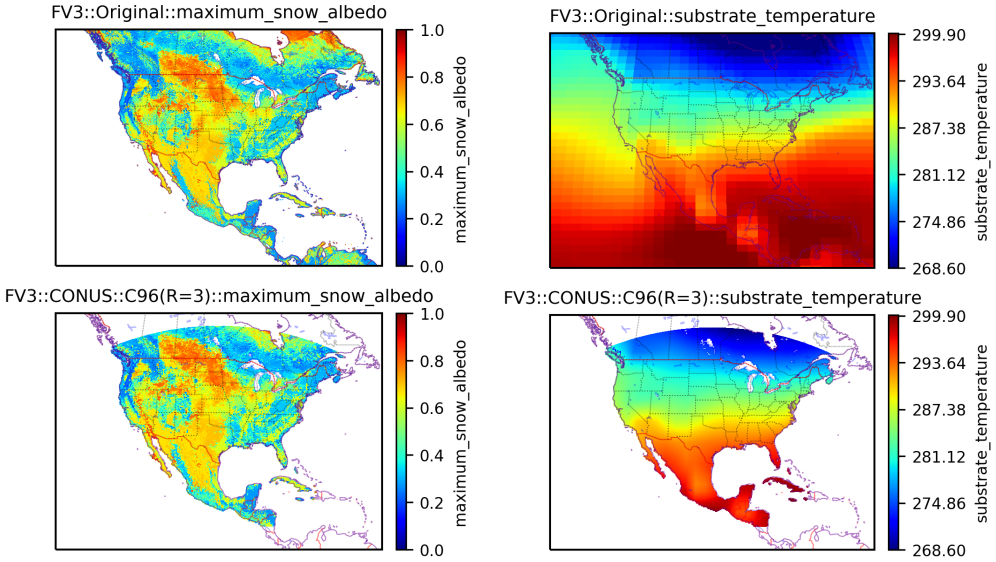

Figure 10.14: Maximum snow albedo, and Substrate temperature

   (b) Vegetation type and greenness (12 files for each month)

   (c) Snow free albedo (visible/near IR black/white sky albedo for each month)

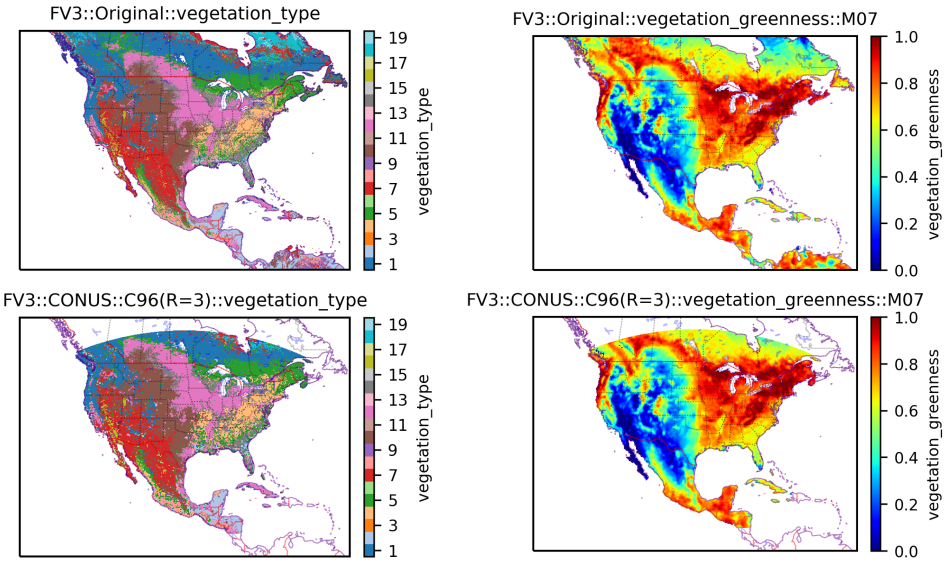

Figure 10.15: Vegetation type and greenness

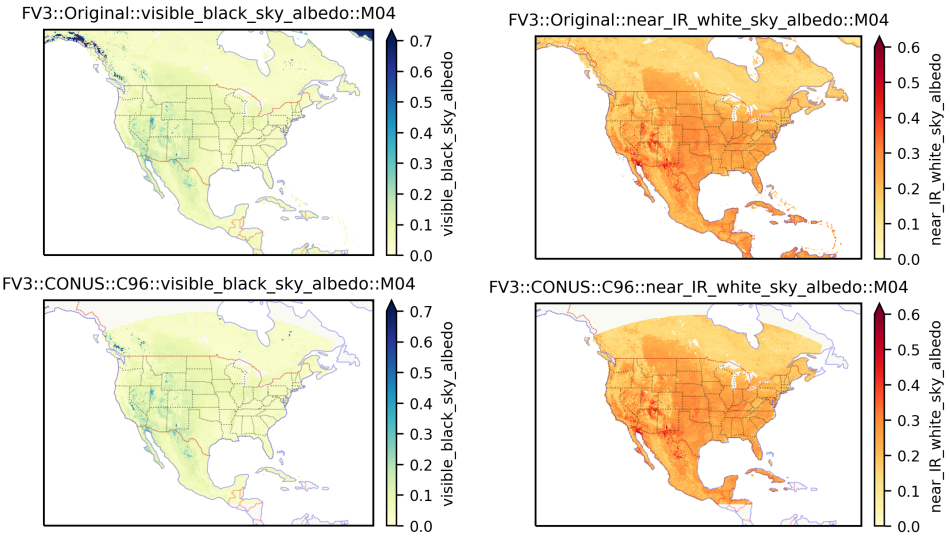

Figure 10.16: Visible black and near IR white sky albedo in April

10.2.8 Static Fields: Global ('fix_am')

The python script 'plot_fv3sar_fixam.py' creates figures for global static fields (input files, format: GRB) in the directory of 'fix_am'. The required files can be found in the namelist as described in Table 2.47.

1. Input files
   - global_glacier.2x2.grb
   - global_maxice.2x2.grb
   - global_snoclim.1.875.grb
   - global_soilmgldas.t1534.3072.1536.grb
   - CFSR.SEAICE.1982.2012.monthly.clim.grb
   - RTGSST.1982.2012.monthly.clim.grb
2. Output files
   - fv3_fixam_{variable}.png
   - fv3_fixam_{variable}_m{month}.png
   - fv3_fixam_{variable}_m{month}_L{level}.png
3. Run on HPC
   - On Hera:

```
# Check Input and Output paths and names
vim plot_fv3lam_fixam.py
# Submit interactive job for a long run-time (<30min)
salloc -q debug -t 0:30:00 --nodes=1 -A fv3-cam
# Run python script
python3 plot_fv3lam_fixam.py
```

   - On Orion:

```
# Check Input and Output paths and names
vim plot_fv3lam_fixam.py
# Interactive mode on HPC
salloc --ntasks=1 --time=00:30:00
```

```
# Run the script
python3 plot_fv3lam_fixam.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input NetCDF files are located
3	vars_fixam	Specific field name of the input files
4	out_fig_dir	Path to the directory where output files are created
5	out_title_base	Basic form of the title in the figures
6	out_fname_base	Basic form of the output file name for the fields
7	back_res	Resolution of the background plot

Table 10.7: Namelist in the script for plotting global static fields.

4. Output:

   (a) Ice cover (surface, and mean sea in January)

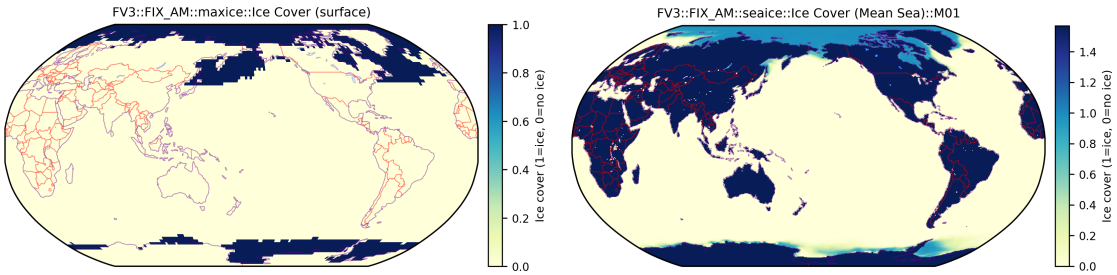

Figure 10.17: Ice cover (surface, and mean sea in January)

   (b) Sea Surface Temperature, and soilmgldas (March, L10)

10.2.9 Time-dependent IC/LBC Fields

The python script 'plot_fv3sar_icbc.py' creates figures for time-dependent initial (IC) and lateral boundary (LBC) fields on the halo and blending layers along the boundary of a regional domain as well as the original data of the fields for comparison. Since 'halo=4', '0 ≤n_blend≤ 10', and the

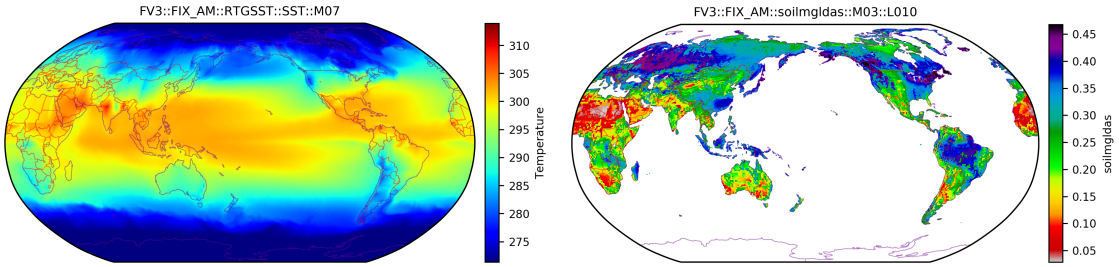

Figure 10.18: Sea surface temperature, and soilmgldas

number of grid points along the axis is more than 190, the IC/LBC fields in the files are too narrow and long to plot in the regular coordinate system. Therefore, the target areas in this script are four corners of the domain. Since *FV3* uses the Arakawa C and D grid systems, velocity components are evaluated at the centers of top, bottom, left, and right grid faces. Therefore, some of top, bottom, left, and right boundaries have one more layer (row/column) than other fields depending on which component is evaluated. The axes of top, bottom, left, and right are contracted to the middle of the axes to secure reasonable spaces for comparison between cells near the corners. However, this approach makes us look some discontinuous points. To verify the boundary fields, this script creates another file for the original GFS data. Since this GFS file contains the initial condition at t=000, it should be compared to the data of 'gfs_bndy.tile7.000.nc'. The figure shows two different coordinate systems with the same data: (1) general equi-distant spacing, and (2) contracted spacing to the middle, which is the same as the boundary plot.

1. Input files
   - gfs_bndy.tile7.[XXX].nc (for boundary plot)
   - gfs_data.tile7.nc (for initial GFS data plot in the main domain)
2. Output files
   - fv3_icbc_{domain}_C{res}_{Variable}_L{layer number}_t{time}_B{n_blend}.png (boundary)
   - fv3_icbc_{domain}_C{res}_{Variable}_L{layer number}_gfs.png (initial GFS data)
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_icbc.py
```

```
# Run the script
python3 plot_fv3lam_icbc.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input NetCDF files are located
3	fnm_in_bndr_base	Basic form of the input NetCDF files
4	bndr_step	Time steps in the file names
5	vars_bc	Variables for plotting
6	ilvl	Specific number of the vertical level (layer number) for 3-D variables
7	domain	Domain name
8	res	Resolution (C+resolution)
9	gtype	Grid type ('ESG', 'GFDL')
10	refine	Refinement ratio
11	out_fig_dir	Path to the directory where output files are created
12	out_title_base	Basic form of the title in the figure
13	out_fname_base	Basic form of the output file name
14	n_halo	Number of additional boundary arrays (halo), typically halo=4
15	n_blend	Number of blending layers

Table 10.8: Namelist in the script for plotting IC/LBC fields.

- **Note)** The namelist of the tracers can be found in Table 2.39.
- **Note)** In the figures, the dashed lines in white represent the boundary of the regional domain.
- **Note)** Total number of layers = n_halo + n_blend.

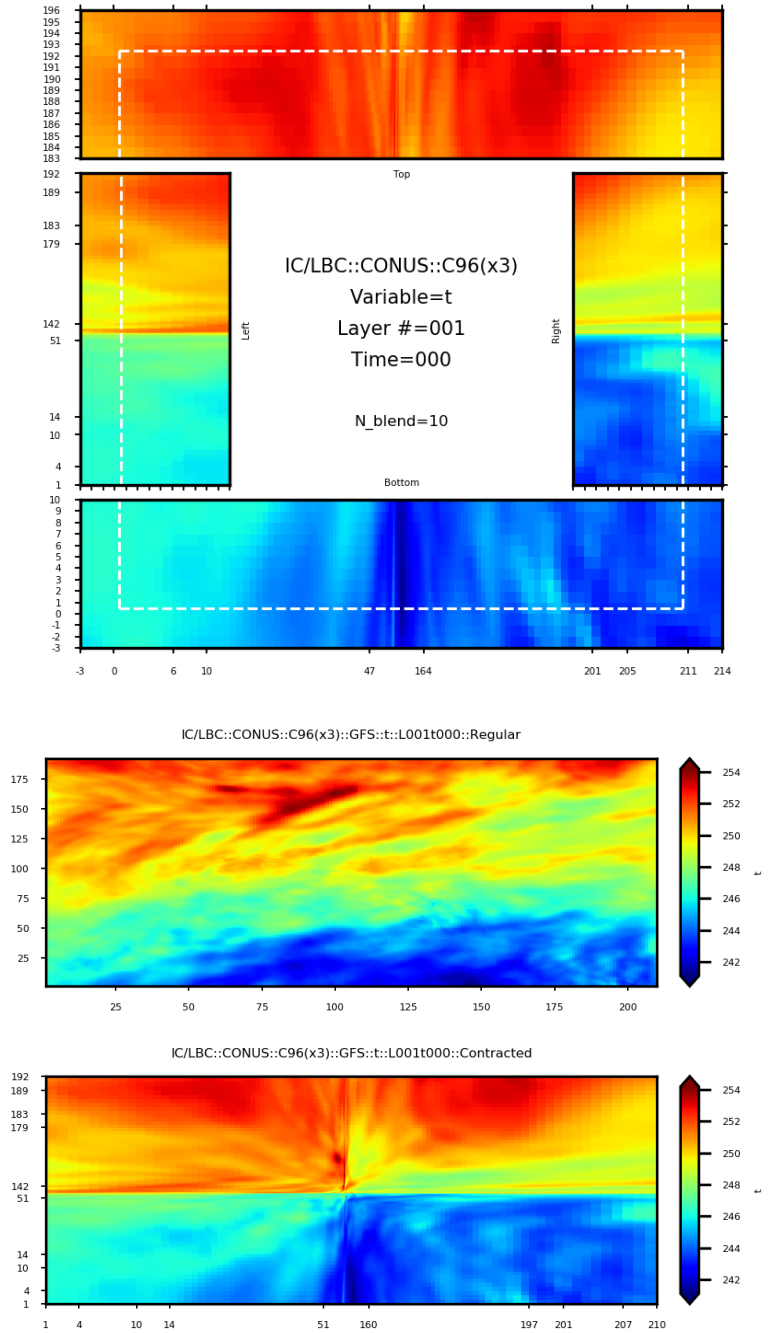

Figure 10.19: Temperature (t) along the boundary at the first layer (initial boundary condition, time=000, layer number=1) and GFS data inside the domain

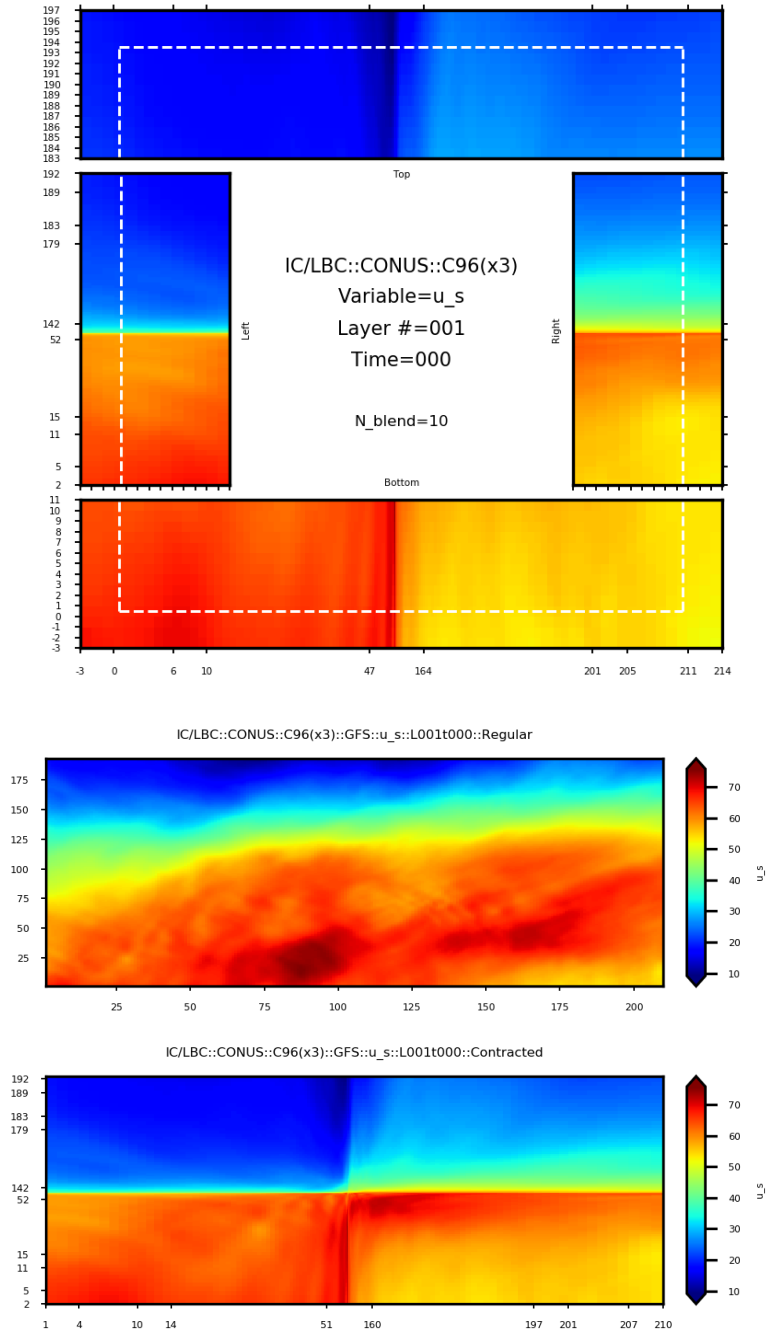

Figure 10.20: Longitudinal velocity at the south edge (u_s) along the boundary (initial boundary condition, time=000, layer number=1) and GFS data inside the domain

10.2.10 Initial Surface Climatology Fields

This script creates a figure to illustrate surface climatology fields in the input 'fix' files.

1. Input files
   - sfc_data.tile7.nc
2. Output files
   - fv3_isfc_{domain}_C{res}_{Variable}.png
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_isfc.py
# Run the script
python3 plot_fv3sar_isfc.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input NetCDF files are located
3	fnm_input	Input NetCDF file
4	domain	Domain name
5	res	Resolution (C+resolution)
6	gtype	Grid type ('ESG', 'GFDL')
7	refine	Refinement ratio
8	fnm_in_orog	Orography file
9	vars_isfc	plotting variables in the data file
10	out_fig_dir	Path to the directory where output files are created
11	out_title_base	Basic form of the title in the figure
12	out_fname_base	Basic form of the output file name
13	back_res	Background resolution

Table 10.9: Variables for plotting initial surface climatology.

No.	Variable	Meaning	No.	Variable	Description
1	alvsf	Visible black sky albedo at zenith 60 degree			
2	alvwf	Visible white sky albedo			
3	alnsf	Near-IR black sky albedo at zenith 60 degree			
4	alnwf	Near-IR white sky albedo			
5	d_conv	Thickness of free convective layer (fcl)			
6	f10m	10 meter wind speed over lowest model wind speed			
7	facsf	Fractional coverage with strong cosz dependency			
8	facwf	Fractional coverage with weak cosz dependency			
9	ffhh	Surface exchange coefficient for heat			
10	ffmm	Surface exchange coefficient for momentum			
11	qrain	Sensible heat flux due to rainfall (W)			
12	slmsk	Sea (0), land (1), ice (2) mask			
13	slc	Volume fraction of unfrozen soil moisture			
14	smc	Volume fraction of soil moisture			
15	snoalb	Maximum snow albedo over land in fraction			
16	srflag	Snow/rain flag for precipitation			
17	shdmin	Minimum areal fractional coverage of annual green vegetation			
18	shdmax	Maximum areal fractional coverage of annual green vegetation			
19	canopy	Canopy moisture content (m)	36	z_c	Sub-layer cooling thickness
20	dt_cool	Sub-layer cooling thickness	37	uustar	Boundary layer parameter
21	fice	Sea-ice fraction	38	hice	Sea-ice depth
22	q2m	Two-meter q	39	tref	Reference temperature (K)
23	slope	Surface slope type	40	xt	Heat content in DTL
24	sheleg	Snow depth (m)	41	xs	Salinity content in DTL
25	snwdph	Physical snow depth	42	xu	u-current content in DTL
26	stc	Soil temperature (K)	43	xv	v-current content in DTL
27	stype	Soil type	44	xz	DTL thickness
28	t2m	Two-meter temperature	45	zm	MXL thickness
29	tg3	Deep soil temperature	46	xtts	d(xt)/d(ts)

30	tisfc	Sea-ice surface temperature	47	xzts	d(xz)/d(ts)
31	tprcp	Total precipitation	48	c_0	Coefficient 1 for d(tz)/d(ts)
32	tsea	Sea-surface temperature	49	c_d	Coefficient 2 for d(tz)/d(ts)
33	vfrac	plant greenness	50	w_0	Coefficient 3 for d(tz)/d(ts)
34	vtype	Vegetation type	51	w_d	Coefficient 4 for d(tz)/d(ts)
35	zorl	Surface roughness (*cm*)	52	ifd	Index to start dtm run

Table 10.10: Namelist in the script for plotting initial surface climatology fields.

4. Output:

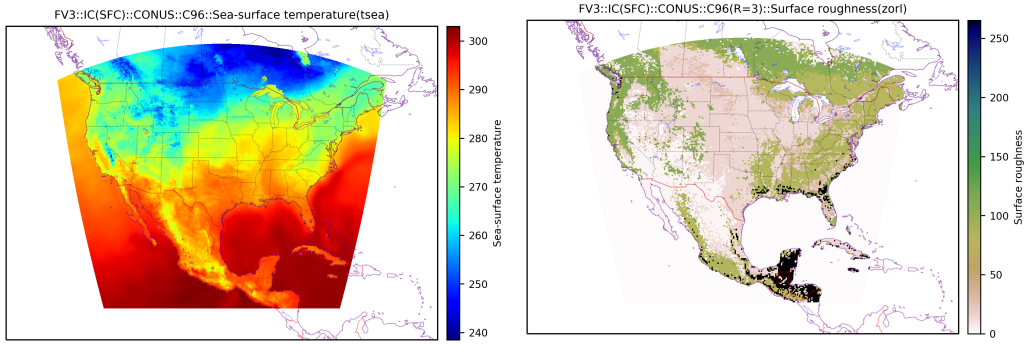

Figure 10.21: Sea-surface temperature, and roughness

10.2.11 Historical CO_2 Data

This script creates figures to illustrate monthly CO_2 data in the input 'co2historicaldata_20XX.txt' file.

1. Input files
   - co2historicaldata_20XX.txt
2. Output files
   - fv3_co2his_m{month}.png
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_co2his.py
# Run the script
python3 plot_fv3lam_co2his.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input file is located
3	fnm_in	Input text file
4	out_fig_dir	Path to the directory where output files are created
5	out_title_base	Base of the figure title
6	out_fname_base	Base of the output file name
7	back_res	Background resolution

Table 10.11: Namelist in the script for plotting 'grid_spec'.

4. Output:

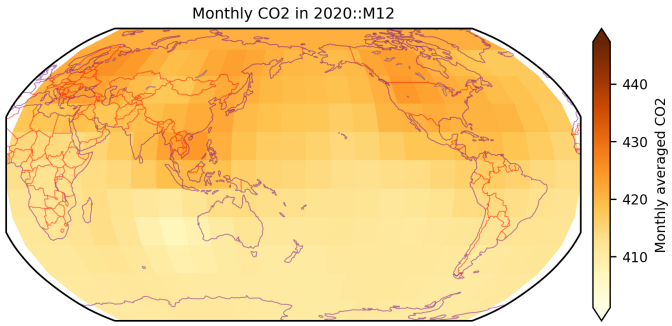

Figure 10.22: Monthly CO_2 input data

10.2.12 Output: 'grid_spec'

This script creates a figure to illustrate some grid points in the output 'gird_spec.nc' file.

1. Input files

- grid_spec.nc
- C{res}_grid.tile7.halo0.nc
- C{res}_oro_data.tile7.halo0.nc

2. Output files

   - fv3_out_grdspec_{domain}_C{res}.png

3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_gridspec.py
# Run the script
python3 plot_fv3lam_gridspec.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input NetCDF files are located
3	fnm_input	Input NetCDF file
4	domain	Domain name
5	res	Resolution (C+resolution)
6	gtype	Grid type ('ESG', 'GFDL')
7	refine	Refinement ratio
8	n_gpt	Number of grid points in plot
9	dnm_ref	Path to the directory where the grid and orography files are located
10	fnm_ref_grd	Grid file
11	fnm_ref_oro	Orography file
12	out_fig_dir	Path to the directory where output files are created
13	out_grd_title	Title of the figure
14	out_grd_fname	Output file name
15	back_res	Background resolution

Table 10.12: Namelist in the script for plotting 'grid_spec'.

4. Output:

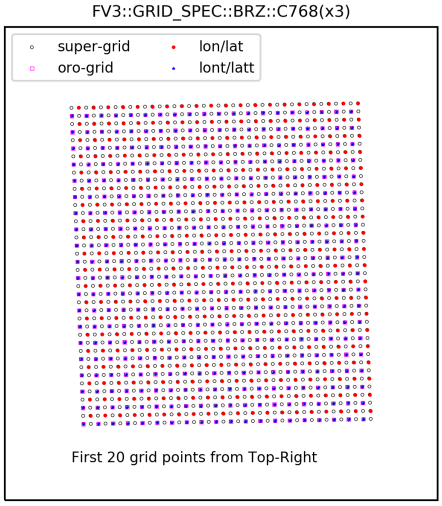

Figure 10.23: Output: grid_spec

10.2.13 Output: 'atmos_static'

1. Input files
   - atmos_static.nc
   - C{res}_oro_data.tile7.halo0.nc
2. Output files
   - fv3_out_atmfix_{domain}_C{res}_[variable].png
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_atmfix.py
# Run the script
python3 plot_fv3lam_atmfix.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_data	Path to the directory where the input NetCDF files are located
3	fnm_input	Input NetCDF file

4	domain	Domain name
5	res	Resolution (C+resolution)
6	gtype	Grid type (‘ESG’, ‘GFDL’)
7	refine	Refinement ratio
8	dnm_ref	Path to the directory where the grid and orography files are located
9	fnm_ref_oro	Orography file
10	vars_atm	Variables in the input file, which can be found in Section 2.6.3
11	out_fig_dir	Path to the directory where output files are created
12	out_title_base	Title of the figure
13	out_fname_base	Output file name
14	back_res	Background resolution

Table 10.13: Namelist in the script for plotting ‘atmos_static’.

4. Output:

(a) One-dimensional data:

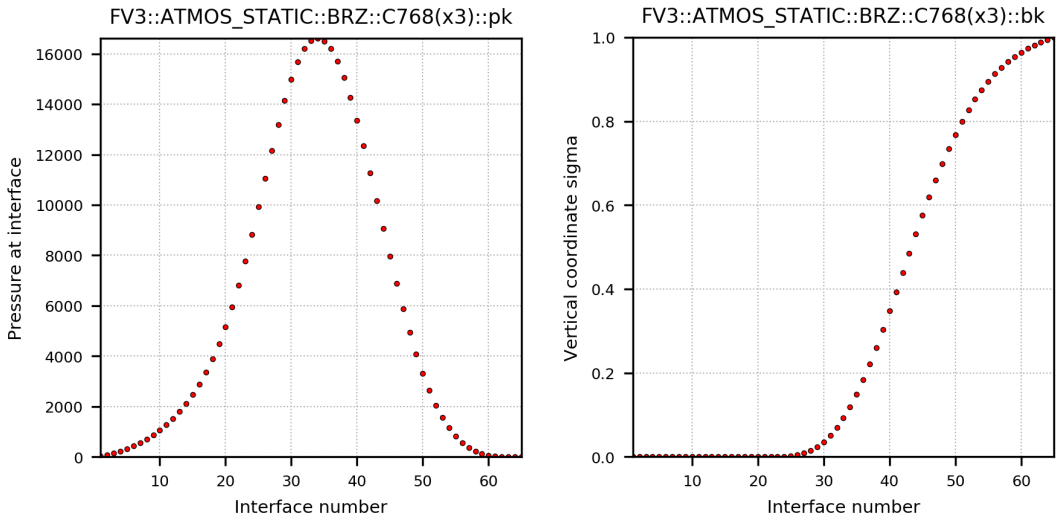

Figure 10.24: Output: atmos_static (‘pk’ and ‘bk’)

(b) Two-dimensional data:

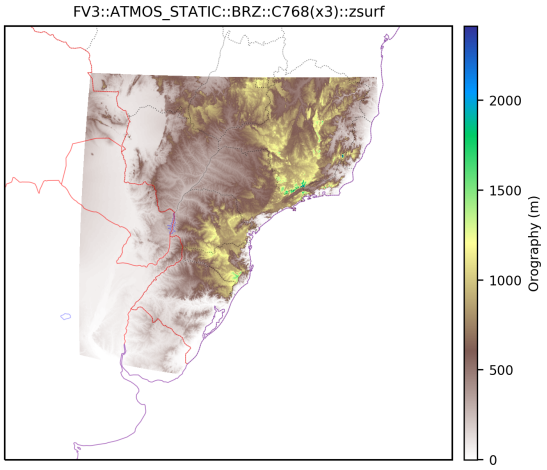

Figure 10.25: Output: atmos_static ('zsurf')

10.2.14 Output: 'dynf'

1. Input files
   - dynfXXX.nc
2. Output files
   - fv3_out_dyn_{domain}_C{res}_[variable].png
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_dynf.py
# Run the script
python3 plot_fv3lam_dynf.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_out	Path to the directory where the input NetCDF files are located
3	fnm_hr	Input file number
4	domain	Domain name
5	res	Resolution (C+resolution)

6	gtype	Grid type (‘ESG’, ‘GFDL’)
7	refine	Refinement ratio
8	vars_dyn	Variables in the input file, which can be found in Section 2.6.3
9	out_fig_dir	Path to the directory where output files are created
10	out_title_base	Title of the figure
11	out_fname_base	Output file name
12	back_res	Background resolution

Table 10.14: Namelist in the script for plotting ‘dynfXXX’.

4. Output:

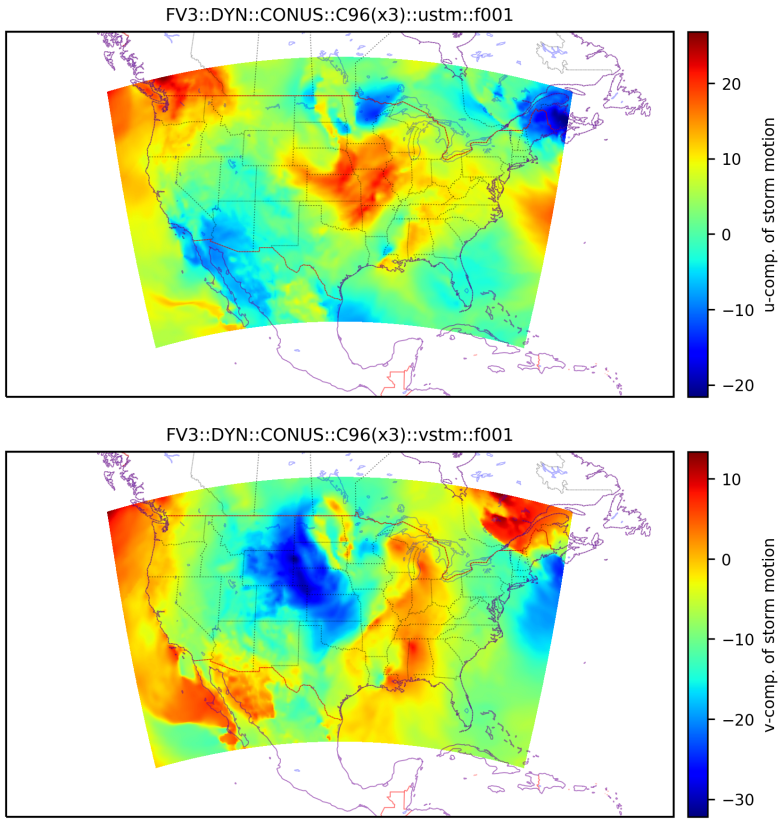

Figure 10.26: Output: dynfXXX

10.2.15 Output: 'phyf'

1. Input files

   - phyfXXX.nc

2. Output files

   - fv3_out_phy_{domain}_C{res}_[variable].png

3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_phyf.py
# Run the script
python3 plot_fv3lam_phyf.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_out	Path to the directory where the input NetCDF files are located
3	fnm_hr	Input file number
4	domain	Domain name
5	res	Resolution (C+resolution)
6	gtype	Grid type ('ESG', 'GFDL')
7	refine	Refinement ratio
8	vars_phy	Variables in the input file, which can be found in Section 2.6.3
9	out_fig_dir	Path to the directory where output files are created
10	out_title_base	Title of the figure
11	out_fname_base	Output file name
12	back_res	Background resolution

Table 10.15: Namelist in the script for plotting 'phyfXXX'.

4. Output:

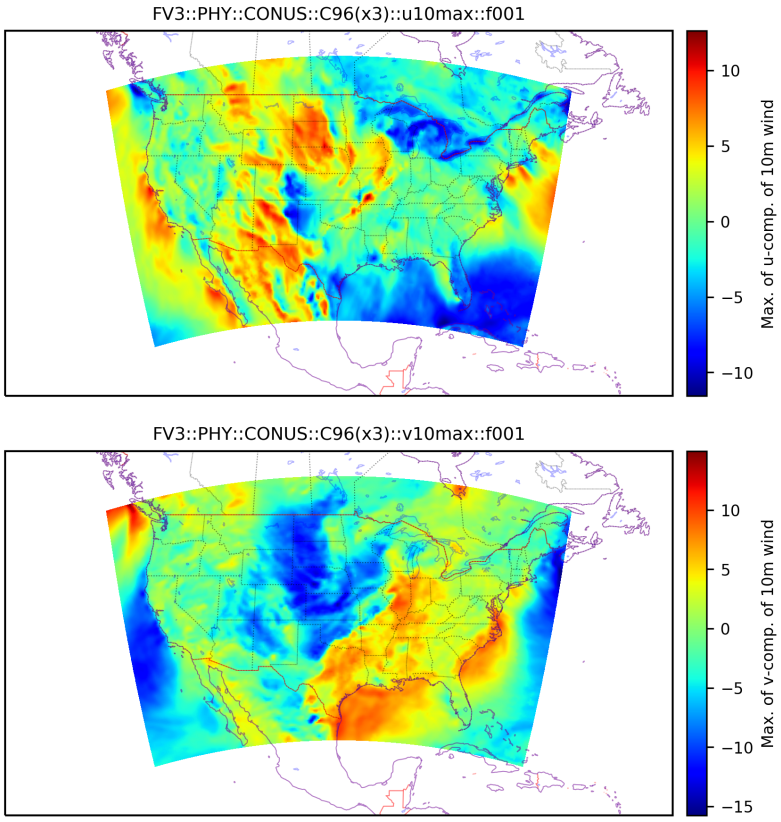

Figure 10.27: Output: phyfXXX

10.2.16 Output: 'BGRD3D (natlev.grib2)'

1. Input files
   - BGRD3DXX.tmXX (*.natlev.grib2)
2. Output files
   - fv3_out_lev_{domain}_C{res}_[variable].png
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_bgrd3d.py
# Run the script
python3 plot_fv3lam_bgrd3d.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_out	Path to the directory where the input GRIB2 file is located
3	fnm_in	Input file name
4	domain	Domain name
5	res	Resolution (C+resolution)
6	gtype	Grid type ('ESG', 'GFDL')
7	refine	Refinement ratio
8	nlvl	Total number of vertical layers in the model
9	ilvl	Layer number to be plotted
10	vars_grb2	Variables in the input file (Table 8.2)
11	out_fig_dir	Path to the directory where output files are created
12	out_title_base	Title of the figure
13	out_fname_base	Output file name
14	back_res	Background resolution ('10m', '50m', '100m')
15	back_img	Background image ('on', 'off')

Table 10.16: Namelist in the script for plotting 'lev.grib2'.

4. Output:

10.2.17 Output: 'BGDAWP (natprs.grib2)'

1. Input files

   - BGDAWPXX.tmXX (*.natprs.grib2)

2. Output files

   - fv3_out_prs_{domain}_C{res}_[variable].png

3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_bgdawp.py
# Run the script
```

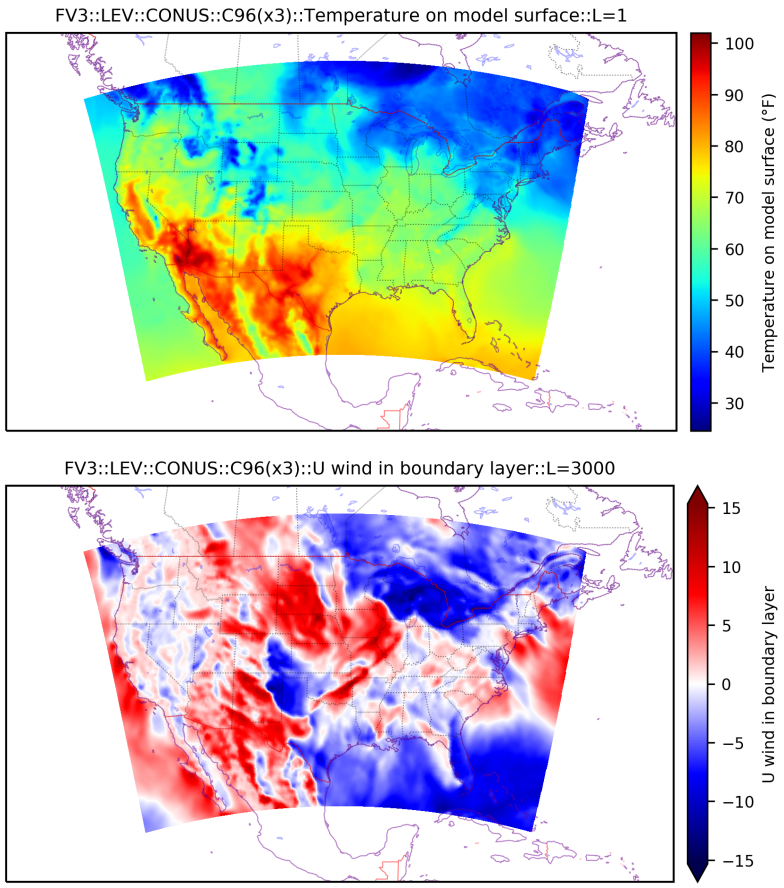

Figure 10.28: Output: BGRD3D (lel.grib2)

```
python3 plot_fv3lam_bgdawp.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_out	Path to the directory where the input GRIB2 file is located
3	fnm_in	Input file name
4	domain	Domain name
5	res	Resolution (C+resolution)
6	gtype	Grid type (‘ESG’, ‘GFDL’)
7	refine	Refinement ratio

8	ilvl	Layer number to be plotted
9	vars_grb2	Variables in the input file (Table 8.3)
10	out_fig_dir	Path to the directory where output files are created
11	out_title_base	Title of the figure
12	out_fname_base	Output file name
13	back_res	Background resolution (‘10m’, ‘50m’, ‘100m’)
14	back_img	Background image (‘on’, ‘off’)

Table 10.17: Namelist in the script for plotting ‘prs.grib2’.

4. Output:

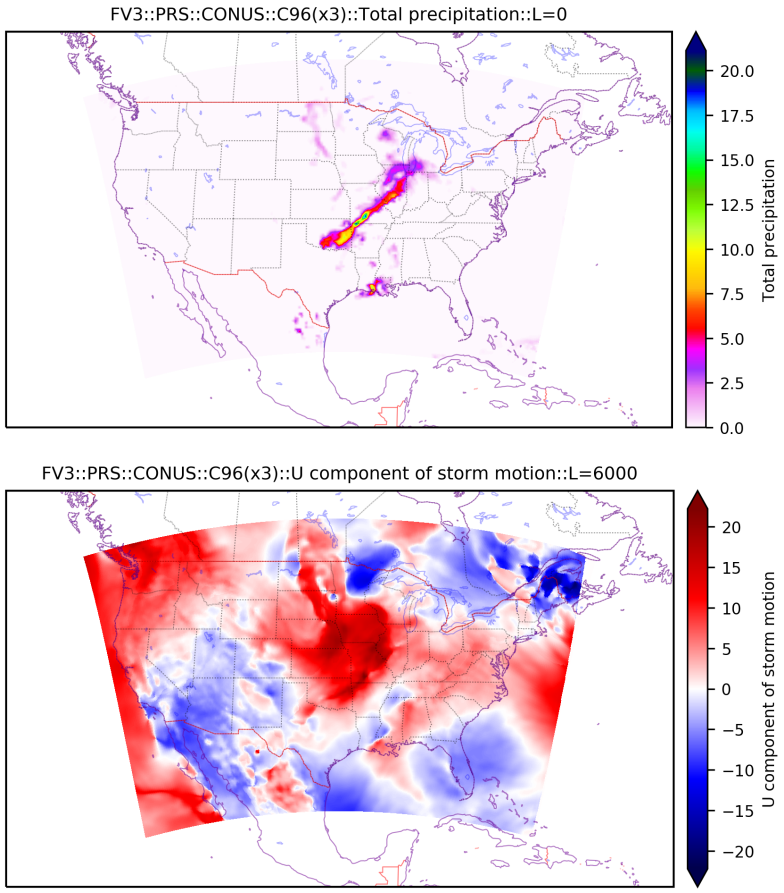

Figure 10.29: Output: BGRD3D (lel.grib2)

10.2.18 Output: Comparison of Two NetCDF (GRIB2) Files

This script creates a figure to illustrate a comparison between two NetCDF or GRIB2 files on the same grid.

1. Input files
   - Two NetCDF or GRIB2 files on the same grid.
2. Output files
   - fv3_out_comp_{domain}_C{res}_[variable].png
3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_comp2f.py
# Run the script
python3 plot_fv3lam_comp2f.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting
2	dnm_in1	Path to the directory where the first input file is located
3	dnm_in2	Path to the directory where the second input file is located
4	fnm_in1	Name of the first Input file
5	fnm_in2	Name of the second Input file
6	domain	Domain name
7	res	Resolution (C+resolution)
8	gtype	Grid type (‘ESG’, ‘GFDL’)
9	refine	Refinement ratio
10	vars_phy	Variables in the input file
11	grb_name	Name of the variable in GRIB2
12	grb_typlvl	Type of level of the variable in GRIB2
13	ilvl	Layer number to be plotted
14	cmp_lbl	Additional text for comparison in title and label
15	out_fig_dir	Path to the directory where output files are created

16	out_title_base	Title of the figure
17	out_fname_base	Output file name
18	back_res	Background resolution

Table 10.18: Namelist in the script for plotting comparison.

4. Output:

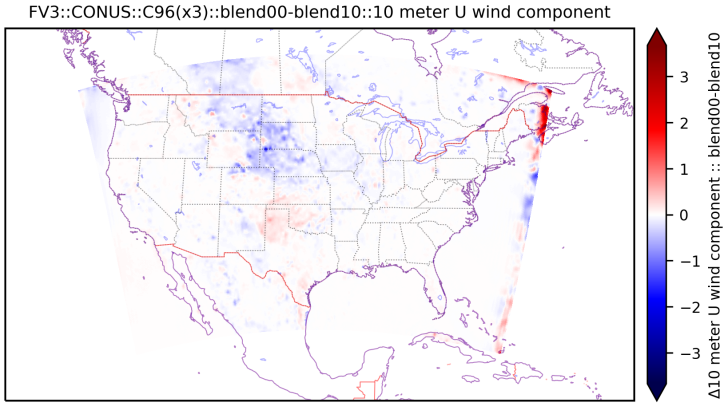

Figure 10.30: Output: comparison

10.2.19 MRMS: Composite Reflectivity Radar Data

1. Input files

   - QCComposite_00.50_[date]_[time]_new.grib2 (Composite reflectivity)
   - ref3D_[date]_[time]_new.grib2 (26 layer reflectivity)

   Notes)

   - The MRMS radar data can be obtained from HPSS as shown in Section C.2.
   - The original GRIB2 file should be converted to complex packing by *wgrib2* to avoid an API error using *Python*.

2. Output files

   - fv3_mrms_comRefl_{domain}_[date]_[time].png (Composite reflectivity)

- fv3_mrms_xzRefl_{domain}_[date]_[time].png (Cross-sectional reflectivity)
- fv3_mrms_3dRefl_{domain}_[date]_[time].png (3-D reflectivity)

3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_mrms.py
# Run the script
python3 plot_fv3lam_mrms.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting (‘hera’)
2	s_date	Date of the input data (YYYYMMDD)
3	s_time	Time of the input data (HHmm)
4	dnm_data	Path to the directory where the input GRIB2 file is located
5	fnm_in_com	Input file name of composite reflectivity
6	fnm_in_3d	Input file name of 3-D reflectivity
7	domain	Domain name
8	plt_com	Flag for plotting composite reflectivity (‘yes’ or ‘no’)
9	plt_crx	Flag for plotting cross-sectional reflectivity (‘yes’ or ‘no’)
10	crx_lon / crx_lat	Longitude/latitude of the target point located at the center of the cross sections (in degrees)
11	crx_dist	Distance from the target point for the limits of the cross sections (in degrees)
12	plt_3d	Flag for plotting 3-dimensional reflectivity (‘yes’ or ‘no’)
13	out_fig_dir	Path to the directory where output files are created
14	out_title_com	Title of the output figure of the composite reflectivity
15	out_fname_com	Output file name of the composite reflectivity
16	out_fname_crx	Output file name of the cross-sectional reflectivity
17	out_fname_3d	Output file name of the 3-dimensional reflectivity
18	back_res	Background resolution (‘10m’, ‘50m’, ‘100m’)
19	back_img	Background image (‘on’, ‘off’)

Table 10.19: Namelist in the script for plotting ‘mrms_rada_reflectivity.grib2’.

4. Output:

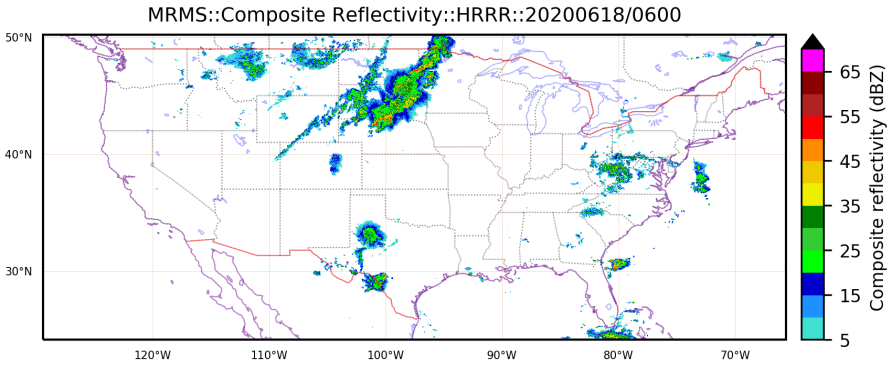

Figure 10.31: MRMS radar: composite reflectivity

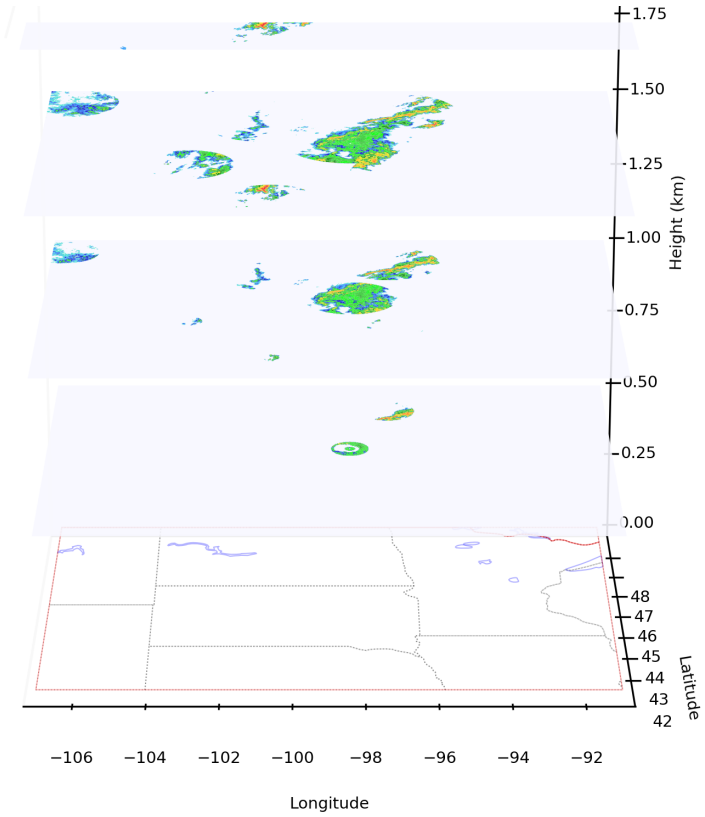

Figure 10.32: MRMS radar: 3-dimensional reflectivity

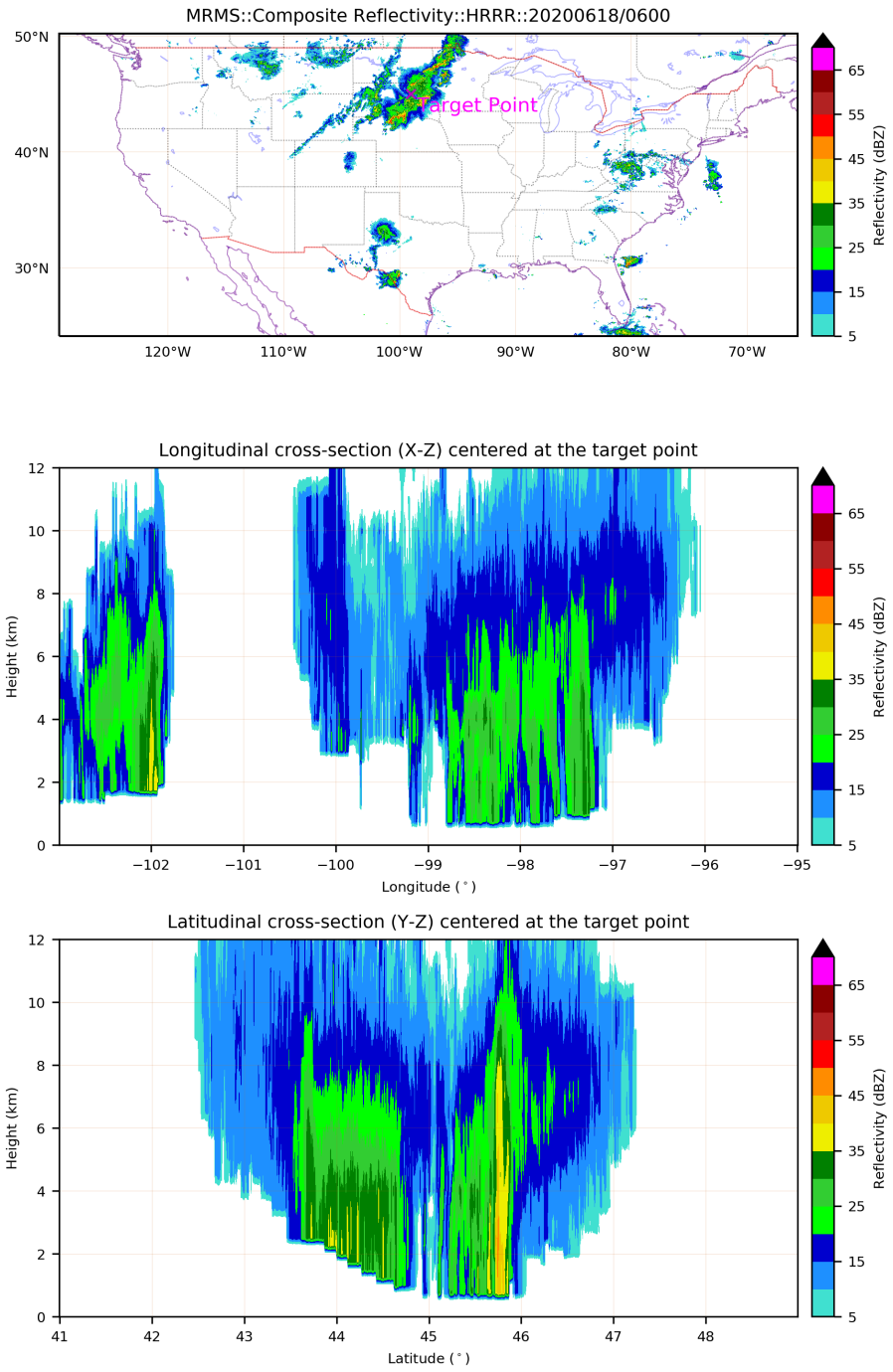

Figure 10.33: MRMS radar: cross-sectional reflectivity

10.2.20 Animation: Hourly Comparison of Composite Reflectivity

1. Input files

   - QCComposite_00.50_[date]_[time]_new.grib2 (radar data files)
   - {domain}.tXXz.bgdawpXX.tmXX.grib2 (FV3-LAM result files)

   Notes)

   - The MRMS radar data can be obtained from HPSS as shown in Section C.2.
   - The original GRIB2 file should be converted to complex packing by *wgrib2* to avoid an API error using *Python*.

2. Output files

   - fv3_out_ani_refl_comp_{domain}_[date].gif

3. Run on HPC

```
# Check Input and Output paths and names
vim plot_fv3lam_ani_comref.py
# Run the script
python3 plot_fv3lam_ani_comref.py
```

No.	Variable name	Description
1	machine	HPC machine for plotting (‘hera’)
2	domain	Domain name
3	res	Domain resolution
4	s_date	Date in the data directory name (YYYYMMDD)
5	plot_hrs	Plotting hours for animation
6	s_time	Time of the input data (HHmm)
7	dnm_data_mdl	Path to the directory where the output GRIB2 files are located
8	dnm_data_dat	Path to the directory where the radar data are located
9	svar	Output field name
13	out_fig_dir	Path to the directory where output files are created
14	out_title_sub1	Sub-title of the sub-plot 1 for output data
15	out_title_sub2	Sub-title of the sub-plot 2 for radar data

16	back_res	Background resolution (‘10m’, ‘50m’, ‘100m’)
17	back_img	Background image (‘on’, ‘off’)

Table 10.20: Namelist in the script for animation comparing output data.

4. Output:

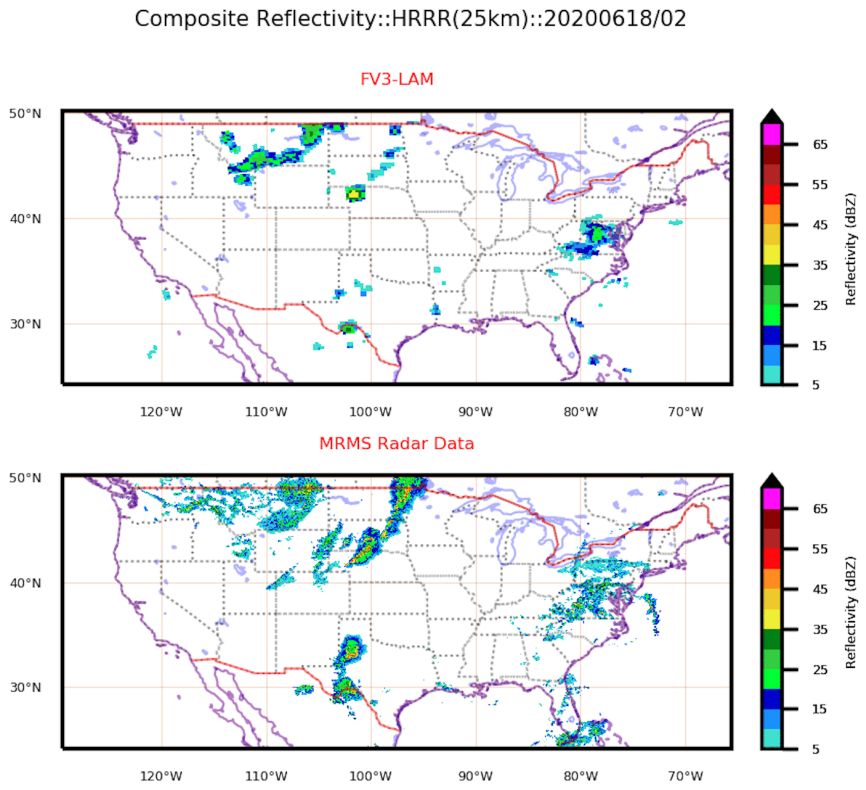

Figure 10.34: Animation: hourly comparison with the MRMS rada data

Appendix A

GitHub

A.1 Flowchart for GitHub 'Pull Request (PR)'

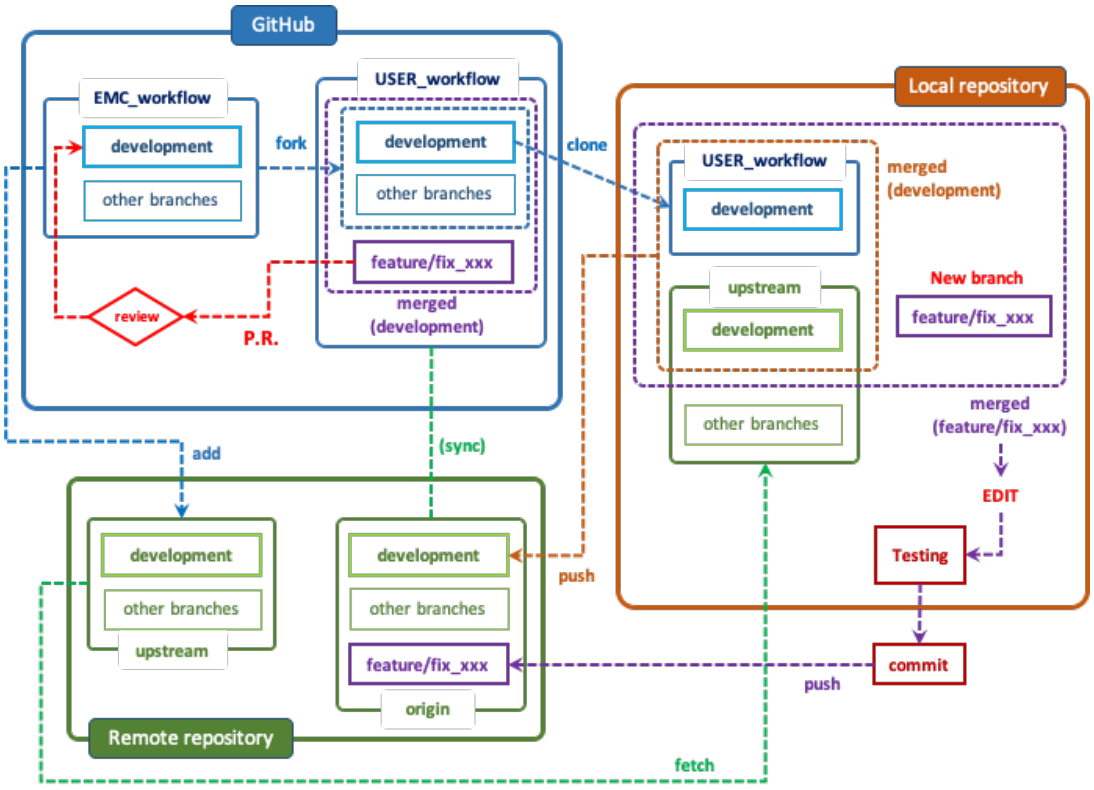

Figure A.1: Flowchart for Pull Request (PR)

A.2 Creating a New Repository Fork

1. In a web browser, go to the GitHub page of the 'NOAA-EMC/regional_workflow' repository.

```
https://github.com/NOAA-EMC/regional_workflow
```

2. Press the 'Fork' button on the top right. The URL of your fork will be

```
https://github.com/{GITHUBUSER}/regional_workflow
```

3. On a terminal window of HPC, clone your fork in the '{HOMEdir}' directory (local repository) and check out the 'develop' branch:

```
cd {HOMEdir}
git clone -b develop --recursive https://github.com/{GITHUBUSER}/
    regional_workflow
```

   where '{HOMEdir}' is the parent directory where the 'develop' branch of the regional workflow will be cloned.

4. When you clone your fork, git will create a remote (i.e. a pointer to a repository location) named 'origin' that points to your fork. To check the list of remotes that are defined in your clone, issue the command:

```
cd {HOMEdir}/regional_workflow/
git remote -v
```

   - If you have not created any new remotes since cloning your fork, the output will be as follows:

```
origin https://github.com/{GITHUBUSER}/regional_workflow.git (fetch)
origin https://github.com/{GITHUBUSER}/regional_workflow.git (push)
```

     This shows that there is a remote named 'origin' that points to your fork. It has two URLs for fetching (pulling) and pushing.

   - The authoritative 'NOAA-EMC/regional_workflow' can be pointed by the 'origin' fetching URL or another remote such as 'upstream' in this case. However, using the 'origin' is more restrictive than the other approach.

5. Create a new remote named 'upstream' that points to the authoritative 'NOAA-EMC/regional_workflow' repository:

```
cd {HOMEdir}/regional_workflow/
git remote add upstream https://github.com/NOAA-EMC/regional_workflow
git remote -v
```

   - The output should be as follows:

```
origin   https://github.com/$GITHUBUSER/regional_workflow.git (fetch)
origin   https://github.com/$GITHUBUSER/regional_workflow.git (push)
upstream        https://github.com/NOAA-EMC/regional_workflow.git (fetch
    )
upstream        https://github.com/NOAA-EMC/regional_workflow.git (push)
```

A.3 Updating the Branch in the Clone

You should update the 'develop' branch in your clone (local repository) as well as in your fork in order to have the latest changes in the 'develop' branch of the authoritative 'NOAA-EMC/regional_workflow' repository.

- When you create new feature branches to address new issues, you start off with the latest version of 'develop' in the authoritative repository in those branches.
- Updating the 'develop' branch in your clone entails merging the latest version of the 'develop' branch of the 'NOAA-EMC/regional_workflow' into the 'develop' branch of your fork.

1. Fetch the contents of the 'upstream' repository (i.e. the 'NOAA-EMC/regional_workflow' repository) to your local disk:

```
cd {HOMEdir}/regional_workflow/
git fetch upstream
```

   This will fetch all the branches from the 'NOAA-EMC/regional_workflow' repository and put them on your local disk. However it will not automatically check any of them out.

   - You can see which branch you are currently on (branch with an asterisk):

```
git branch -vva
```

There should be a list of remote branches which include the one you want to merge into your local 'develop' branch.

2. Merge the 'remote/upstream/develop' branch that you fetched in Step 1 into your local 'develop' branch:

```
git checkout develop
git merge upstream/develop
```

Your local 'develop' branch will now be an exact copy of the latest version of the 'develop' branch of the 'NOAA-EMC/regional_workflow' repository.

- It is recommended not to make your own changes to your local 'develop' branch.
- Any changes you make to address issues should be made in a separate feature branch.
- Keeping your local 'develop' branch unchanged (except for occasionally updating it with the latest from the authoritative branch) allows you
  (a) to merge the latest changes from other developers into your feature branch.
  (b) to quickly create a new and up-to-date feature branch whenever you need to address a new issue.

3. Update the 'develop' branch in your fork. You can do this by pushing your updated local 'develop' branch to your fork.

```
git checkout develop
git push
(GitHub Username)
(GitHub Password)
```

- In case of 'warning: push.default is unset.'

```
git push
(Warning)
git config --global push.default matching
git config --global push.default simple
git push
```

A.4 Creating a New Feature Branch

1. Make sure that you are on the local 'develop' branch:

```
cd {HOMEdir}/regional_workflow/
git checkout develop
```

2. Create a new feature branch named 'fix_something':

```
git checkout -b feature/fix_something
```

   You will now be on this new branch with the command:

```
git branch -vva
```

   At this point, the 'feature/fix_something' is only a local branch, i.e. it only exists in your clone. It does not exist in your fork (nor in the authoritative repository).

3. Push the local 'feature/fix_something' branch to your fork:

```
git push -u origin feature/fix_something
(GitHub Username)
(GitHub Password)
```

   Now you can see the new branch 'feature/fix_something' in your fork in the 'branches' tab on 'https://github.com/{GITHUBUSER}/regional_workflow'.

A.5 Making Changes to the Feature Branch

1. Update your local 'develop' branch with the latest version of the 'develop' branch of the authoritative repository as shown in A.3.

2. Make sure that your are on the 'feature/fix_something' branch:

```
cd {HOMEdir}/regional_workflow/
git checkout feature/fix_something
```

3. Merge your local 'develop' branch into your local 'feature/fix_something' branch:

```
git merge develop
```

This command will report merge conflicts, in which case you will have to resolve them manually. If you have set up a visual mergetool in your git configuration file, you can use the 'git mergetool' command to visually resolve the conflict.

4. Edit/add/delete files as necessary in your local 'feature/fix_something' branch in order to address the issue.

5. After making your changes, you will likely need to run one or more tests (e.g. workflow end-to-end tests, regression tests) to ensure that your modifications do not break the code under various configurations.

6. Commit the changes you made to your local 'feature/fix_something' branch.

```
git add file1 file2 file3 ...
git commit
```

This will open up a text editor window in which you can enter a commit message. The commit will occur once you close the editor window.

7. In order to completely address the issue (e.g. after running tests on the first set of changes), you may have to make additional changes to your local 'feature/fix_something' branch. For this purpose, repeat the steps 1 to 6 as necessary until the issue has been addressed without breaking the code under various configurations.

8. Push your local 'feature/fix_something' branch to your fork:

```
git push
(GitHub Username)
(GitHub Password)
```

Your fork will now have a version of the 'feature/fix_something' branch with all of your commits.

A.6 Creating a Pull Request (PR)

Create a pull request to get the changes you made in your 'feature/fix_something' branch into the 'develop' branch of the authoritative repository.

1. On your web browser, go to your GitHub fork (https://github.com/{GITHUBUSER}/regional_workflow) and switch to the 'feature/fix_something' branch.

2. Create a pull request ('New pull request' button):

No.	Variable	Target / example
1	Base repository	NOAA-EMC/regional_workflow
2	Base branch	develop
3	Head repository	{GITHUBUSER}/regional_workflow
4	Head branch	feature/fix_something

Table A.1: Set-up for change comparison.

3. Fill in the message for the PR with information on what the PR is for and what tests you have run, etc.

4. Choose reviewers.
Note) We have placed a file called 'CODEOWNERS' in {HOMEdir} that specifies the GitHub users to notify for various types of files, directories, etc in the code base (GitHub feature). GitHub will look in this file and generate an initial set of reviewers for you. If necessary, you can add additional reviewers manually using the gear icon on the upper-right of the PR page.

5. Submit the Pull Request (PR).

6. Wait to hear back from the reviewers.

7. Once your PR is accepted and merged, to reduce clutter you should delete both your local feature branch and its copy in your fork.

   - To delete your local branch:

   ```
   git branch -d feature/fix_something
   ```

   - To delete the 'feature/fix_something' branch on your fork, you can use one of two methods:

     (a) Delete from within your clone on your local machine.

     ```
     git push origin -d feature/fix_something
     ```

(b) Delete using your web browser:

i. Go to the URL of your fork (https://github.com/ {GITHUBUSER}/regional_workflow).

ii. Click on the '# branches' tab on the head bar (not the 'Branch: *Name*').

iii. Find your feature branch in the list, and click on the red trash can icon on the same line.

A.7 Deleting the Forked Repository

1. Open the repository and navigate to settings (top right):

```
github.com/{GITHUBUSER}/regional_workflow/settings
```

2. Scroll to the bottom of the 'Setting' page.

3. Click on the 'Delete this repository' in Danger Zone.

4. Type '{GITHUBUSER}/regional_workflow' to confirm.

5. Click on the red bar on the bottom.

Appendix B

Workflow Manager: *Rocoto*

Ref.) https://github.com/christopherwharrop/rocoto/wiki/documentation

B.1 How the *Rocoto* Engine Works

Rocoto:

- is a 'Ruby' program that interfaces to the underlying batch system.
- is designed to run weather and climate workflows (not general purpose).
- does all the book keeping necessary to submit tasks when dependencies are satisfied and tracks the progress of the workflow.
- automatically resubmit failed tasks up to a maximum number of attempts, and can recover from system outages.
- has no control over when jobs start to run.
- runs one instance of the workflow for a set of user-defined 'circles'. A 'circle' usually corresponds to a model analysis or initialization time.
- users must define their workflows with the XML language.

1. Executing workflows:
    - The 'rocotorun' command is used to run a workflow:

```
rocotorun -w /path/to/workflow/xml/file -d /path/to/workflow/database/
    file
```

where

- The '-w' flag specifies the name of the workflow definition file. This must be an XML file.
- The '-d' flag specifies the name of the database file that is to be used to store the state of the workflow. The database file is a binary file created and used only by *Rocoto* and need not exit prior to the first time the command is run.

- It is very important to understand that the running process is iterative. Each time the above command is executed, *Rocoto* performs the following actions:
  (a) Read the last known state of the workflow from the database file specified by the '-d' flag.
  (b) Query the batch system to acquire the current state of the workflow.
  (c) Take actions based on new state information (resubmit crushed jobs or submit next jobs)
  (d) Save the current state of the workflow to the file specified by the '-d' flag
  (e) Quit

2. Checking the status of workflows

- 'rocotostat': query the status of a set of cycles and tasks.

```
rocotostat -w /path/to/workflow/xml/file -d /path/to/workflow/database/
    file [-c YYYYMMDDHHMM, [YYYYMMDDHHMM, ...]] [-t taskname, [taskname,
     ...]] [-s] [-T]
```

  - The '-c' option allows you to select specific cycles to query.
  - The '-t' option allows you to select specify tasks.
  - The '-T' option sorts the output by taskname rather than by cycle.
  - The '-s' option will provide a summary report of the status of the cycles themselves rather than information about the tasks.

- 'rocotocheck': query detailed information about a specific task for a specific cycle.

```
rocotockeck -w /path/to/workflow/xml/file -d /path/to/workflow/database/
    file -c YYYYMMDDHHMM -t taskname
```

3. Forcing tasks to run

   - 'rocotoboot': force a task to be submitted, regardless of whether or not dependencies are satisfied or throttling violations would occur.

```
rocotoboot -w /path/to/workflow/xml/file -d /path/to/workflow/database/
    file -c YYYYMMDDHHMM -t taskname
```

4. Rerunning several tasks in the workflow

   - 'rocotoewind': undo the effect of having run some jobs, and mark those jobs as not having run.

```
rocotoewind -w /path/to/workflow/xml/file -d /path/to/workflow/database/
    file -c YYYYMMDDHHMM -t taskname
```

B.2 *Rocoto* XML Language

1. XML Header

   Every XML file must have this text at the top:

```
<?xml version="1.0"?>
<!DOCTYPE workflow
[
]>
```

2. ENTITY

   An 'ENTITY' represents a value that is used in lots of places in the XML.

```
<!ENTITY CYCLE_THROTTLE "4">
<!ENTITY GTYPE "@[GTYPE:-regional]">
```

   The 'ENTITY' is referenced by the systex, '*&{ENTITY_NAME}*;'.

```
<!ENTITY HOME_DIR = "chan-hoo.jeon/fv3sar/">
<!ENTITY LOG = "$HOME_DIR;/log">
```

3. Tag: <workflow>

   This is the main tag for defining workflows. Everything except for the header and ENTITY definitions must be contained within the <workflow> tags.

```
<workflow realtime="F" scheduler="moabtorque" cyclelifespan="0:01:00:00"
    cyclethrottle="5" taskthrottle="5">

# Everything else goes in here

</workflow>
```

   (a) **realtime** : define whether or not the workflow is to be run in realtime ("T" or "True"), or in retrospective mode ("F" or "False"). The difference between realtime and retrospective mode is in how cycles are activated. In realtime mode, cycles are activated based on the current time of day. In retrospective mode, there is no time dependency that must be dhered to. Thus cycles will be activated immediately, in chronological order.

   (b) **scheduler** : must be set to one of "sge", "lsf" (for WCOSS), "torque", "moabtorque" (for Jet), or "moab" (for Gaea). This attribute tells *Rocoto* which batch system to use when managing the workflow. It is recommended that the keep time be set to a minimum of 24 hours.

   (c) **cyclelifespan** : specifies how long a cycle can be active before it expires in the format of "DD:HH:MM:SS". Wall-clock-time requests are automatically capped such that they will not exceed the time when a cycle expires.

   (d) **cyclethrottle** : limits how many cycles may be active at one time. Cycles are active if they have not expired and not all tasks for the cycle have completed successfully.

   (e) **corethrottle** : limits the total number of cores that may be consumed by jobs submitted to the batch system. Any job that is submitted to the batch system, whether it is running or queued, counts against this total.

   (f) **taskthrottle** : limits how many tasks may be active at one time. A task is active if a job has been submitted for it and that job has not finished. Any job that is submitted to the batch system, whether it is running or queued, counts against this total.

4. Tag: <log>

   This defines the path and name of *Rocoto* log files. In general, it is best to define the name of the log to be dynamically dependent on the cycle being processed.

```
<log><cyclestr>&LOG;/workflow_@Y@m@d@H.log</cyclestr></log>
```

5. Tag: <cyclestr>

   It is often necessary to refer to the various components of the current cycle time when defining various aspects of the workflow. The <cyclestr> tag uses flags to represent the various time components of the current cycle. These flags will be replaced with the appropriate date/time component of the current cycle being processed. They can be used in any combination to represent any date/time string desired inside a <cyclestr> ... </cyclestr> block.

   - **offset**: represents offset before or after the current cycle. The format of the offset attribute is 'dd:hh:mm:ss'. Leading field whose values are 0 do not need to be specified.

```
<cyclestr offset="1:00:00">@Y@m@d@H.log</cyclestr>
```

6. Tag: <cycledef>

   The <cycledef> tag defines the set of cycles the workflow is to be run on. *Rocoto* uses the union of all sets of cycles specified by the <cycledef> tags to create the overall cycle pool. If there is overlap between definitions, it will not cause cycles to be run twice. There are two methods to specify cycles. You can mix and match the two different ways to specify cycles as needed.

   (a) **The start-stop-step method**: Specify a start cycle, an end cycle, and an increment. The format of the start and stop cycles is 'yyyymmddhhmm'. The format of the increment is 'dd:hh:mm:ss'.

```
<cycledef>200607060900 201304230900 06:00:00</cycledef>
```

   (b) **The crontab-like method** : Uses a crontab-like format to represent groups of cycles. Six fields such as minute, hour, day, month, year, and weekday must be defined. Each field can be a single value, a range of values, a comma separated list of values, etc.

Table B.1: <cyclestr> tags:

Flag	Time component
@a	Abbreviated weekday name (e.g. “Sun”)
@A	Full weekday name (e.g. “Sunday”)
@b	Abbreviated month name (e.g. “Jan”)
@B	Full month name (e.g. “January”)
@c	Preferred local date and time representation (e.g. “Thu Jul 5 11:27:59 2012”)
@d	Day of month (01–31)
@H	Hour of the day, 24 hour clock (00–24)
@I	Hour of the day, 12 hour clock (01–12)
@j	Day of the year (001–365)
@m	Month of the year (01–12)
@M	Minute of the hour (00–59)
@p	Meridian indicator (“AM” or “PM”)
@P	Meridian indicator (“am” or “pm”)
@s	Number of seconds since January 1, 1970 00:00:00 UTC
@S	Second of the minute (00–59)
@U	Week number of the current year, starting with the first Sunday as the first day
@W	Week number of the current year, starting with the first Monday as the first day
@w	Day of week (Sunday is 0, 0–6)
@x	Preferred representation for the date alone (“07/05/12”)
@X	Preferred representation for the time alone (“11:34:58”)
@y	Year without century (00–99)
@Y	Year with century
@Z	Time zone name

- **group**: Defines distinct sets of cycles because some tasks should only be run for certain subsets of cycles. Multiple <cycledef> tags may be assigned to the same group, but a <cycledef> tag may not be assigned to more than one group.

```
<cycledef group="15min">* /15 * * * 2006-2013 *></cycledef>
```

where an asterisk (*) denotes a shorthand for all values of the field. The above example defines a set of cycles consisting of every 15 minutes of every hour for every day and

every month of the years 2006–2013.

7. Tag: <task>

   It defines the computations that you want to run. Every task must have a unique name defined.

   (a) **cycledefs** : (optional). If set, its value must be a comma separated list of <cycledef> tag group names. If it is not set, the task will be run for every cycle in the general pool of cycles.

   (b) **maxtries** : (optional). When *Rocoto* detects that a task has failed, it will attempt to resubmit it. The maxtries attribute limits the number of times a task can be retried.

   (c) **throttle** : (version 1.1 and higher, optional). If set, its value must be a positive integer. The throttle limits the number of instances of the task which may be queued or running at any one time. There is one instance of the task per cycle. Therefore, the task throttle limits the number of cycles for which the task may be active at any one time.

```
<task name="wrf" cycledefs="3hourly,␣6hrlyJanFeb" maxtries="3">

</task>
```

8. Tag: <command>

   Every <task> tag must contain a <command> tag. This is the command that carries out the task's work and is the command that *Rocoto* will submit to the batch system for execution.

```
<command><cyclestr>&SCRIPTS;/wrf.ksh -c @Y@m@d@H</cyclestr></command>
```

9. Tag: <account>

   Every <task> tag should contain an <account> tag. It is optional to allow for situations where a default may be used. This defines the batch system account/project.

```
<account>&ACCOUNT</account>
```

10. Tag: <queue>

    Every <task> tag should contain a <queue> tag. This defines the batch system queue that *Rocoto* will submit the task to for execution.

```
<queue>&QUEUE_PE</queue>
```

11. Tag: <cores>

    Every <task> tag must contain either one <cores> tag or one <nodes> tag. The <cores> tag defines the number of cores that *Rocoto* will request when submitting the task for execution.

```
<cores>1</cores>
```

12. Tag: <nodes> (version 1.1 and higher)

    The <nodes> tag defines the task geometry (a list of node counts and cores per node). The <nodes> tag should only be used if the task has a requirement to use less than all of the available cores on at least one of its nodes. The <nodes> tag is primarily intended to support OpenMP/MPI hybrid codes, and MPMD codes.

```
<nodes>1:ppn=1+10:ppn12</nodes>
```

    This example requests 1 core on the first node and 12 cores on each of the next 10 nodes.

13. Tag: <walltime>

    Every <task> tag must contain a <walltime> tag. This defines the amount of wall-clock time. The requested walltime is automatically reduced so as not to exceed the amount of time remaining before the task expires.

```
<walltime>00:00:10</walltime>
```

14. Tag: <memory>

    The <memory> tag is usually only needed for serial tasks. This defines the amount of memory.

```
<memory>512M</memory>
```

15. Tag: <jobname>

    Every <task> tag should contain a <jobname> tag because it helps in visual tracking of jobs in queue status outputs. It is optional to allow for situations where a default may be used.

```
<jobname>test</jobname>
```

16. Tags: <stdout>, <stderr>, and <join>

    Every <task> tag should contain either both <stdout> and <stderr> tags or a <join> tag because it helps in tracking and finding the output of the jobs. These tags define the location of the stdout and stderr output of the job that executes the task. The <join> tag is used when you want both stdout and stderr to go to the same place. **DO NOT** set the value of <stdout> and <stderr> to the **same** thing. Instead, use **<join>** to do that.

```
<join>/home/test/test.log</join>

or

<stdout>/home/test/test.out</stdout>
<stderr>/home/test/test.err</stderr>
```

17. Tag: <envar>

    The <envar> tag is used inside the <task> tags to define environment variables. It consists of (name,value) pairs. All <envar> tags must contain a <name> tag. However, <value> tags can be absent in cases where a variable needs to be set, but does not need to be assigned a value.

```
<envar>
   <name>ANALYSIS_TIME</name>
   <value><cyclestr>@Y@m@d@H@M</cyclestr></value>
</envar>
```

18. Tag: <dependency>

    The <dependency> tags are located inside <task> tags, and used to describe the inter-dependencies of the tasks. Dependencies are defined as boolean expressions. There are three types of dependencies: (1) task dependency, (2) data dependency, and (3) time dependency. They can be combined in any combination in a boolean expression to form a task's overall dependencies. There must be exactly **one** tag of the following tags inside the <dependency> tags:

```
<dependency>
        <taskdep>
        <datadep>
        <timedep>
```

```
        w/ boolean operator tag (<and>, <or>, <not>, <nand>, <nor>, <xor>, <
    some>)
</dependency>
```

(a) Tag: <taskdep>

The task attribute of the <taskdep> tag must be set to the name of the task which must complete in order for the dependency to be satisfied. The task attribute must refer to a task that has already been defined in the XML document.

- **cycle_offset** : A task may have a dependency on a task from a different cycle. The 'cycle_offset' attribute allows users to specify an offset from the current cycle to define inter cycle dependencies. The offset can be positive or negative.
- **state** : The 'state' attribute is optional. It allows you to specify whether you want the dependency to be satisfied when a task has completed successfully, or when a task has failed and exhausted retries. It may be set to either "Succeeded" (default) or "Dead".

```
  <taskdep state="Dead" task="wrfpost_f006" cycle_offset="-6:00:00"/>
```

(b) Tag: <datadep>

Data dependencies are defined inside <dependency> tags. File dependencies are satisfied when the file in question exists, has not been modified for at least the amount of time specified by 'age', and is at least as large as the size specified by 'minsize'.

- **age** : (optional). It contains the time (dd:hh:mm:ss) that the file must not be modified before the file is considered to be available. The default is zero.
- **minsize** : (optional). It contains the minimum size for the file before it is considered to be available. The default value is zero. B(or b): byte (default), K(or k): kilobyte, M (or m): megabyte, G (or g): gigabyte.

```
  <datadep age="00:02:00" minsize="1024b" /datadep>
```

(c) Tag: <timedep>

The value between the start and end of the <timedep> tag is a time in 'yyyymmddhhmmss' format. Time dependencies are satisfied with the wall clock time is equal to or greater than the time specified. All times are calculated in GMT.

```
<timedep><cyclestr offset="&DEADLINE;">@Y@m@d@H@M@S</cyclestr> </
 timedep>
```

(d) The boolean operator tags

There are several boolean operators that may be used to compose boolean expressions of dependencies. The operators and the operands (<taskdep>, <datadep>, <timedep>) can be combined without limit.

Table B.2: Boolean operator tags:

Flag	Time component
<and>	Satisfied if all enclosed dependencies are satisfied
<or>	Satisfied if at least one of the enclosed dependencies is satisfied
<not>	Satisfied if the enclosed dependency is not satisfied
<nand>	Satisfied if at least one of the enclosed dependencies is not satisfied
<nor>	Satisfied if none of the enclosed dependencies are satisfied
<xor>	Satisfied if exactly one of the enclosed dependencies is satisfied
<some>	Satisfied if the fraction of enclosed dependencies that are satisfied exceeds some threshold ($0 \leq$ threshold ≤ 1)

Appendix C

Data Transfer from/to/between NOAA HPCs

C.1 Trusted Data Transfer Node: Between NOAA HPCs

- DTN is only accessible from some hosts **within .noaa.gov**.
- The DTN supports ssh-based authentication transfer methods, which currently include scp, rsync, and bbcp (Jet and Hera only).
  - Hera: dtn-hera.fairmont.rdhpcs.noaa.gov
  - Niagara: dtn-niagara.fairmont.rdhpcs.noaa.gov
  - Jet: dtn-jet.rdhpcs.fairmont.noaa.gov
- Default authentication uses your RSA token.
- Username is case sensitive in the scp command.
- Only the high-performance filesystems (**scratch**) are available, **NOT** your **/home** filesystem.

C.1.1 On Hera

1. Hera to Jet:

```
cd [path to the parent directory of the file/directory on Hera]
scp [file / -r directory] Chan-hoo.Jeon@dtn-jet.rdhpcs.noaa.gov:/mnt/lfs1/
    projects/jetmgmt/Chan-hoo.Jeon/
```

C.1.2 On Orion

1. Hera to Orion:

```
cd [path to the parent directory on Orion]
scp (-r : only for directory) Chan-hoo.Jeon@dtn-hera.fairmont.rdhpcs.noaa.
    gov:/scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/[file or directory] .
```

2. Orion to Hera:

```
cd [path to the parent directory on Orion]
scp [file / -r directory] Chan-hoo.Jeon@dtn-hera.fairmont.rdhpcs.noaa.gov:/
    scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/
```

C.1.3 On WCOSS

1. WCOSS to Hera:
   SCP transfer to Hera can be accomplished via the MARS/VENUS access nodes:

```
cd [path to the parent directory on WCOSS]
scp [file / -r directory] Chan-hoo.Jeon@dtn-hera.fairmont.rdhpcs.noaa.gov:/
    scratch2/NCEPDEV/stmp1/Chan-hoo.Jeon/
```

2. Hera to WCOSS:

```
cd [path to the parent directory on WCOSS]
scp (-r: only for directory) Chan-hoo.Jeon@dtn-hera.fairmont.rdhpcs.noaa.gov
    :/scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/(file or directory) .
```

C.2 Data Transfer from HPSS

The centralized, long-term data archive system at NESCC is based on the High Performance Storage System (HPSS).
On HPSS:

- MRMS (Multi-Radar/Multi Sensor) radar data (reflectivity):

```
/BMC/fdr/Permanent/{YYYY}/{MM}/{DD}/data/radar/mrms/{YYYYMMDDHH}00.zip
```

– Each zip file contains 180 zip files for every minute for the next three hours and three gauge zip files. For example, the '202006180000.zip' file contains '202006180000.zip' to '202006180259.zip'. The last two digits denote minutes.

```
202006180000.zip
├── 202006180000.zip
│   ├── MRMS_MergedReflectivityQCComposite_00.50_[*].grib2
│   ├── MRMS_MergedReflectivityQC_{AA.BB}_[*].grib2
│   └── MRMS_Reflectivity_{XX}C_00.50_[*].grib2
├── 202006180001.zip
├──     ⋮
├── 202006180258.zip
├── 202006180259.zip
├── 202006180000.Gauge.zip
├── 202006180100.Gauge.zip
└── 202006180200.Gauge.zip
```

Figure C.1: Structure of an MRMS-data zip file

– Data format: 'MRMS_MergedReflectivityQC_{AA.BB}_{YYYYMMDD}-{HHmmSS}.grib2' where {AA.BB} denotes the height of the field in kilometers.

Note) To avoid the 'GRIB_API ERROR:' with *Python*, the original GRIB2 files should be converted to new GRIB2 files using complex packing by *wgrib2*.

- 3 hourly CCPA (Climatology-Calibrated Precipitation Analysis) data (total precipitation):

```
/NCEPPROD/hpssprod/runhistory/rh{YYYY}/{YYYYMM}/{YYYYMMDD}/
   com_ccpa_prod_ccpa.{YYYYMMDD}.tar
```

Note) There is a bug in the hourly CCPA data. The time stamp is wrong for the '00Z' directory. I typically use the 3 hourly data and set my fhzero to 3 hours.

C.2.1 On Hera

1. Load the HPSS module:

```
module load hpss
```

2. Search for files on HPSS:

```
hsi
cd /BMC/fdr/Permanent/2020/06/18/data/radar/
cd /NCEPPROD/hpssprod/runhistory/rh2020/202006/20200618/
# Check if the files you want exist
exit
```

3. Pull data from HPSS:

   - MRMS radar data:

```
cd /scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/TMP
mkdir mrms_2020061806
cd mrms_2020061806
hsi get /BMC/fdr/Permanent/2020/06/18/data/radar/mrms/202006180600.zip
mv 202006180600.zip 2020061806.zip
unzip -l  2020061806.zip. (list of files)
unzip 2020061806.zip 202006180600.zip
unzip 202006180600.zip
```

   - CCPA data:

```
cd /scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/TMP
mkdir ccpa_20200618
cd ccpa_20200618
htar -xvf /NCEPPROD/hpssprod/runhistory/rh2020/202006/20200618/
    com_ccpa_prod_ccpa.20200618.tar
```

4. Convert to new GRIB2 file using complex packing:

```
module load wgrib2
wgrib2 input.grib2 -set_grib_type c3(or complex3) -grib_out output.grib2
```

C.3 External Data Transfer from Hera to Local Desktop

1. Transfer data to the 'eDTN' (on Window #1, Hera)

```
cd [path to the files]
tar -cvzf [zip file name: fv3.tar.gz] [file names: fv3_isfc_*]
scp [zip file name] Chan-hoo.Jeon@edtn.fairmont.rdhpcs.noaa.gov:
```

2. Transfer data from 'eDTN' to a local desktop (on Window #2, local)

```
cd [path to the directory where the file will be downloaded on the local
    desktop]
scp Chan-hoo.Jeon@edtn.fairmont.rdhpcs.noaa.gov:[file name] .
tar -xvzf [zip file name: fv3.tar.gz]
```

C.4 SSH Port Tunnel from Linux-like Systems: Between Desktop and HPC

Note) This is more useful for transferring a number of files.

1. Find local port number
   - Close all current sessions opens on the HPC system.
   - Open a window terminal on your Desktop.
   - Find your unique local port number by logging onto your specified HPC system (Hera/-Jet). It can be found on the terminal windows as

     ```
     Local port 2027 forwarded to remoted host.
     Remote port YYYYY forwarded to local host.
     ```

   - Close all sessions once it's been recorded.
2. Open two terminal windows.
3. On Window #1, enter the following (e.g. local port number=2027):

   ```
   ssh -X -L2027:localhost:2027 Chan-hoo.Jeon@hera-rsa.rdhpcs.noaa.gov
   ```

4. On Window #2, After the first session has been opened with the port forwarding, then any further connections (login via ssh, copy via scp) will work as expected (**Password: PIN+Token code**).

- To transfer a file (files) to HPC

```
scp -P 2027 /export/emc-lw-chjeon/cjeon/NOAA/*.png Chan-hoo.
    Jeon@localhost:/scratch2/NCEPDEV/fv3-cam/Chan-hoo.Jeon/tools/
    fv3sar_pre_plot/Fig/
```

or

```
rsync <put rsync options here> -e 'ssh -l Chan-hoo.Jeon -p 2027' /local/
    path/to/files Chan-hoo.Jeon@localhost:/path/to/files/on/HPC
```

- To transfer a file from HPC

```
scp -P 2027 Chan-hoo.Jeon@localhost:/scratch2/NCEPDEV/fv3-cam/Chan-hoo.
    Jeon/tools/fv3sar_pre_plot/Fig/*.png /export/emc-lw-chjeon/cjeon/
    NOAA/Fig/
```

or

```
rsync <put rsync options here> -e 'ssh -l Chan-hoo.Jeon -p 2027' Chan-
    hoo.Jeon@localhost:/path/to/files/on/HPC /local/path/to/files
```

Appendix D

HPC Job Commands

D.1 Hera

D.1.1 Slurm

No.	Command	Description
1	sbatch <runscript>	Submit a job
2	scancel <jobID>	Cancel the job from the queue
3	squeue -u <userID>	Display a listing of the current jobs in the queue
4	scontrol show job <jobID>	Display detailed information about jobs
5	squeue -- start	Display a point-in-time estimate of when jobs may start

Table D.1: Slurm commands on Hera

D.1.2 Other Useful Commands

No.	Command	Description
1	saccount_params	Display information regarding project for a user
2	shpcrpt -c hera -u <userID>	Display current MTD compute information for all projects for a user
3	module list	Show loaded modules
4	module avail	Show all available modules
5	module load <package>	load the package

6	module unload <package>	unload the package
7	module swap <A> <B>	Swap package A for package B
8	module purge	Unload all loaded modules
9	module spider	Show all possible modules

Table D.2: Other commands on Hera

D.2 Orion

D.2.1 Slurm

No.	Command	Description
1	sbatch <runscript>	Submit a job
2	scancel <jobID>	Cancel the job from the queue
3	squeue -u <userID>	Display a variety of information
4	squeue -p batch –start	Report the start time of pending jobs
5	showuserjobs	Display the current node and batch job status broken down into userids
6	sacct	Display accounting information from the Slurm database
7	sjstat	Display short summary of running jobs and scheduling pool data
8	sview	Graphical user interface to get and update state information

Table D.3: Slurm commands on Orion

D.2.2 Other Useful Commands

No.	Command	Description
1	quota -s	Display user's disk usage
2	saccount_params	Display information regarding project for a user (‘module load contrib noaatools’ must be run in advance.)
3	module list	Show loaded modules
4	module avail	Show all available modules
5	module load <package>	load the package

6	module unload <package>	unload the package
7	module swap <A> <B>	Swap package A for package B
8	module purge	Unload all loaded modules
9	module spider	Show all possible modules

Table D.4: Other commands on Orion