Pre-Processing:
regional_esg_grid, global_equiv_resol, make_solo_mosaic, orog, filter_topo, shave

UFS SRW App Training

Jeff Beck

NOAA/GSL/CIRA

21 September 2021

The UFS_UTILS Code Repository

- The UFS_UTILS repository houses numerous utilities necessary for the SRW App and is checked out and built automatically; however, for users interested in the source code:
  - The UFS_UTILS repository can be cloned from this address:
    - https://github.com/NOAA-EMC/UFS_UTILS
  - The SRW App uses the ‘release/public-v2’ and ‘develop’ branches
- The “sorc” directory contains all utility source code used by the SRW App:
  - regional_esg_grid, global_equiv_resol, make_solo_mosaic, orog, filter_topo, shave, sfc_climo_gen, chgres_cube, fvcom_tools, etc.
- Code support exists on all NOAA HPC, Cheyenne, generic linux, and MacOS
  - UFS Users’ Support Forum (Initialization): https://forums.ufscommunity.org/forum/initialization
- Additional information on the UFS_UTILS repository wiki page:
  - https://github.com/NOAA-EMC/UFS_UTILS/wiki

Overview of the “regional_esg_grid” utility

- Generates the Extended Schmidt Gnomonic (ESG) grid files used by the SRW App

- Selection of a predefined domain in the SRW App will automatically set necessary namelist parameters for creation of an ESG grid during the “make_grid” task in the workflow based on variables in the regional_workflow/ush/set_predef_grid_params.sh file

- The ESG grid namelist (RRFS_CONUS_3km domain example):

```
&regional_grid_nml
   delx = 0.0134898241 – Angle increment in the x-direction (deg)
   dely = 0.0134898241– Angle increment in the y-direction (deg)
   lx = -1760 – Half the number of points (plus halos) on the supergrid (x-direction), relative to the center of the domain
   ly = -1050 – Half the number of points (plus halos) on the supergrid (y-direction), relative to the center of the domain
   plat = 38.5 – Center latitude (deg) for the grid
   plon = -97.5 – Center longitude (deg) for the grid
   pazi = 0.0 – Azimuth parameter representing the orientation of the grid in degrees
 /
```

- When generating a new grid, use a pre-existing domain as a template in the “set_predef_grid_params.sh” script and define the grid center lat/lon, grid spacing in kilometers, and number of grid points (no halos); the workflow then calculates the corresponding namelist parameters (see Gerard Ketefian’s “Defining a New Domain” presentation this afternoon)

Output from the “regional_esg_grid” utility

- Produces a single netCDF output file named “regional_grid.nc” with six halo cells (where the LBCs are applied) on each side of the domain

- The “make_grid” workflow task then copies “regional_grid.nc” to a new file name, using a format required by the FV3 executable:
  - Includes C-resolution, number of halo cells, and the tile number (“tile7” indicates a regional grid file, while 1-6 are reserved for global tiles in the MRW App)
    - The C-resolution is the number of points along one tile of a uniform global domain; for the regional grid, it is the equivalent global C-resolution based on the grid spacing at the center of the regional domain (calculated by “global_equiv_resol”; called during the make_grid task)
  - Example: regional_grid.nc -> C3357_grid.tile7.halo6.nc

- In addition to FV3, this file name format is also necessary for the “make_solo_mosaic”, “orog”, “shave”, and “sfc_climo” utilities to create their own output

Overview of the “make_solo_mosaic” utility

- Provides general information about the grid that is required by FV3
- Run as part of the “make_grid” workflow task
- Reads in the C-res-based grid file created by regional_esg_grid
- Creates a C-res-based “mosaic” netCDF file (e.g, C3357_mosaic.halo6.nc); located in the “grid” directory when running the SRW App
- Contains information on the number of tiles (1), the tile name (tile7), the grid file name (e.g., C3357_grid.tile7.halo6.nc), and the path to the grid file

Overview of the “orog” and “filter_topo” utilities

- “orog” utility
  - The “make_orog” task handles the staging of 30” fixed files used to create the raw topography file and the “orog” utility command line arguments
  - Reads in the C-resolution-conforming mosaic and grid files
  - Creates a raw netCDF topography file (e.g., oro.C3357.tile7.nc) with six halo cells

- “filter_topo” utility
  - Reads in the raw topography file
  - The “make_orog” task handles the namelist generation on the fly
  - Uses assumptions about the stretch factor and C-resolution to filter the raw topography
  - Not yet adapted for non-uniform regional grids (currently uses the equivalent global C-resolution grid value and stretch_fac=0.999; latter cannot be one, otherwise executable believes grid is global)
  - Interpolates/extrapolates between or beyond a set of seven C-resolution values for which filtering parameters are provided to create the filtered orography
  - Replaces the original raw topography file with the filtered output (e.g., oro.C3357.tile7.nc)
  - The “make_orog” task then copies the output to a C-res-compliant file name with halo number (e.g., C3357_filtered_orog.tile7.halo6.nc)

Overview of the “shave” utility

- The FV3-LAM model executable requires two grid files and two orography files with specific halo values for each:
  - Grid files: one with a halo of three cells, another with a halo of four cells
  - Orography files: one with a halo of four cells, another with no halo cells

- To create these files, the “shave” utility is applied to the original grid and filtered orography files (e.g., C3357_grid.tile7.halo6.nc and C3357_filtered_orog.tile7.halo6.nc)

- The “make_grid” and “make_orog” tasks handle the namelist generation for and execution of the “shave” utility

- Example of final file list in the /grid directory:
  - C3357_grid.tile7.halo3.nc - created by the “shave” utility
  - C3357_grid.tile7.halo4.nc - created by the “shave” utility
  - C3357_grid.tile7.halo6.nc - original grid file
  - C3357_mosaic.halo3.nc - created by the “shave” utility
  - C3357_mosaic.halo4.nc - created by the “shave” utility
  - C3357_mosaic.halo6.nc - original mosaic file

- Example of final file list in the /orog directory:
  - C3357_oro_data.tile7.halo0.nc - created by the “shave” utility
  - C3357_oro_data.tile7.halo4.nc - created by the “shave” utility

Pre-Processing: sfc_climo_gen

UFS SRW App Training

George Gayno

NOAA/EMC/I.M. Systems Group

21 September 2021

Overview

- **sfc_climo_gen** stands for “surface climatological field generation”.
- Runs as the last step when creating a new model grid.
- Creates climatological surface fields - such as vegetation type or plant greenness fraction - from global datasets.
  - Some fields are static (vegetation or soil type), some are time-varying (greenness is a monthly climatology).
  - Fields only change when the grid (land-mask or orography) change.
- Parallel. Built around the ESMF library.
  - Convenient handling of the gnomonic grid.
- For regional grids, will output files with and without the halo region.

Code Structure

- Location of source code within UFS_UTILS - ./sorc/sfc_climo_gen.fd
  - **driver.F90** - The main driver routine.
  - **interp.F90** - The interpolation driver. Reads the input source data and interpolates it to model grid.
  - **model_grid.F90** - Defines the ESMF grid object for the model grid.
  - **output.f90** - Writes the output surface data to a NetCDF file. For regional grids, will output separate files with and without the halo.
  - **program_setup.f90** - Reads the namelist and sets up program execution.
  - **search.f90** - Replace undefined values on the model grid with a valid value (i.e., isolated islands).
  - **source_grid.F90** - Reads model grid specs and land/sea mask for the source data. Set ESMF object for source grid.
  - **utils.f90** - Contains error handling utility routines.

Program inputs - NetCDF

Global datasets - ftp.emc.ncep.noaa.gov/EIB/UFS/SRW/v1p0/fix/fix_sfc_climo/

- 0.05-degree 4-component snow-free albedo - **snowfree_albedo.4comp.0.05.nc**
- 1-degree fractional coverage strong/weak zenith angle albedo - **facsf.1.0.nc**.
- 0.05-degree maximum snow albedo - **maximum_snow_albedo.0.05.nc**.
- 2.6 x 1.5-degree soil substrate temp - **substrate temperature.2.6x1.5.nc**.
- 1.0-degree categorical slope type - **slope_type.1.0.nc**.
- 0.05-degree categorical STATSGO soil type - **soil_type.statsgo.0.05.nc**.
- 0.05-degree categorical IGBP vegetation type - **vegetation_type.igbp.0.05.nc**
- 0.144-degree monthly veg. greenness - **vegetation_greenness.0.144.nc**.

Model grid data -

- Mosaic and grid files - **CRES_mosaic.halo#.nc** and **CRES_grid.tile#.halo#.nc**. From ‘regional_esg_grid’ utility.
- Orography files including halo - **CRES_oro_data.tile#.halo#.nc**. From ‘orog’ utility.

Program inputs - file format

All data is NetCDF on a regular latitude and longitude grid. Ex: veg type.

```
netcdf vegetation_type.modis.igbp.0.05 {
dimensions:
        idim = 7200 ;
        jdim = 3600 ;
        jdim_p1 = 3601 ;
        time = 1 ;
variables:
        float time(time) ;
                time:units = "days since 2015-1-1" ;
        double lat(jdim) ;
                lat:long_name = "grid cell center latitude" ;
        double lon(idim) ;
                lon:long_name = "grid cell center longitude" ;
        double lat_corner(jdim_p1) ;
                lat_corner:long_name = "grid cell corner latitude" ;
        double lon_corner(idim) ;
                lon_corner:long_name = "grid cell corner longitude" ;
        float vegetation_type(time, jdim, idim) ;
                vegetation_type:landice_category = 15 ;
                vegetation_type:missing_value = -999.9f ;

// global attributes:
                :source = "IGBP VEG TYPE" ;
                :projection = "regular lat/lon" ;
```

i/j dimensions of dataset.

Number of time periods.

Valid time of each record.

Latitude/longitude of each grid point.

Veg. type determines land ice.

Non-land points set to missing.

Try your own data. **Only tested with regular lat/lon grids.**

Namelist options

- **input_facsf_file** - path/name of input fract. strong/weak zenith albedo data.
- **input_substrate_file** - path/name of input soil substrate data.
- **input_maximum_snow_albedo_file** - path/name input max snow alb. data.
- **input_snowfree_albedo_file** - path/name of input snow-free albedo data.
- **input_slope_type_file** - path/name of input slope type data.
- **input_soil_type_file** - path/name of input soil type data.
- **input_vegetation_type_file** - path/name of input vegetation type data.
- **input_vegetation_greenness_file** - path/name of input greenness data.

- **mosaic_file_mdl** - path/name of the model mosaic file.
- **orog_dir_mdl** - directory containing the model orography file.
- **orog_files_mdl** - model orography file.

Namelist options (continued)

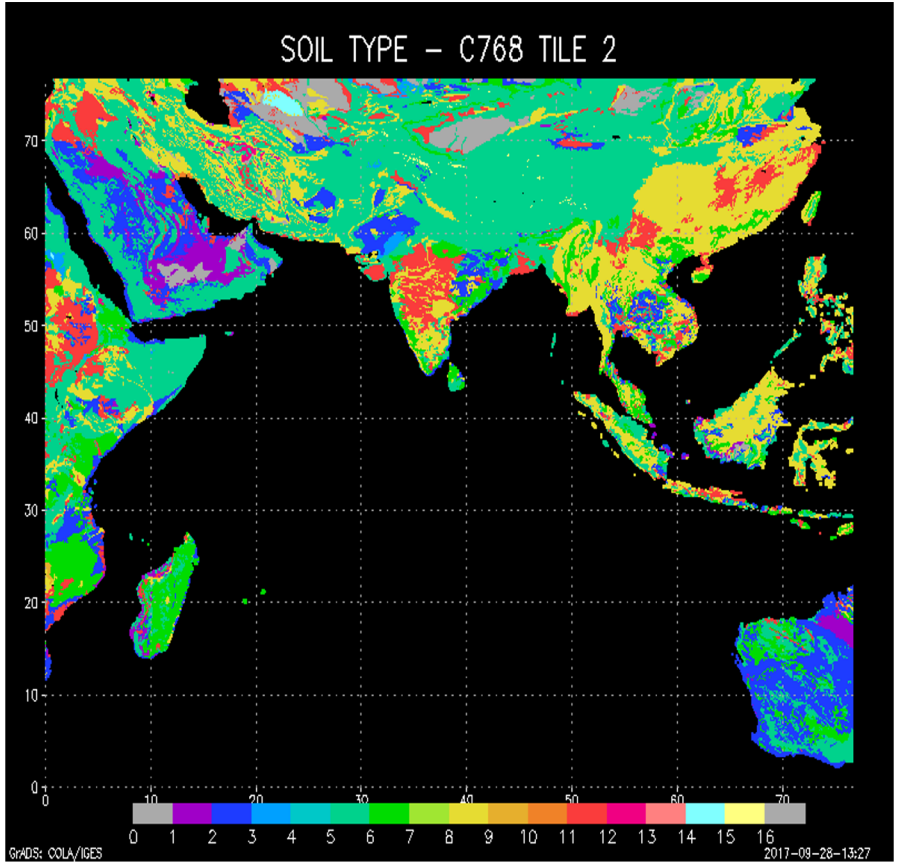
- **halo** - number of rows/columns of the lateral halo.
- **maximum_snow_albedo_method** - interpolation method for this field. Bilinear or conservative. Default is bilinear.
- **snowfree_albedo_method** - interpolation method for this field. Bilinear or conservative. Default is bilinear.
- **vegetation_greenness_method** - interpolation method for this field. Bilinear or conservative. Default is bilinear.

Output files

- All files with and without halo (all NetCDF).

- Fractional coverage strong/weak zenith angle albedo - **CRES_facsf.tile#.halo#.nc**
- Maximum snow albedo - **CRES_maximum_snow_albedo.tile#.halo#.nc**
- Soil substrate temperature - **CRES_substrate_temperature.tile#.halo#.nc**
- Snow-free albedo - **CRES_snowfree_albedo.tile#.halo#.nc**
- Slope type - **CRES_slope_type.tile#.halo#.nc**
- Soil type - **CRES_soil_type.tile#.halo#.nc**
- Vegetation type - **CRES_vegetation_type.tile#.halo#.nc**
- Vegetation greenness - **CRES_vegetation_greenness.tile#.halo#.nc**

Example: Soil type



- Output files are input to chgres_cube.
- Time-varying fields are interpolated to the cycle time.
- There is a chgres_cube option that can override this option for some fields.
- In this case, certain HRRR-based fields are used.
- More on this option later.

Pre-Processing: chgres_cube

UFS SRW App Training

Larissa Reames

NOAA/NSSL/CIMMS

21 September 2021

SRW App Pre-Processing Summary

- The SRW App provides options to run with the following predefined regional grids:
  - 3-km (C3357), 13-km (C775), and 25-km (C403) CONUS domains
- The grid, orography, and sfc_climo utilities can be run for each of the predefined regional grids as part of the workflow, or users can point to pre-existing output from each of these utilities on disk
- Each of the pre-processing utilities used in the SRW App rely on grid-based information to generate their output
- Chgres_cube constitutes the only pre-processing component that uses additional information defined in the config.sh script (e.g., date, external model data)
- Brief note: will mention **develop** vs **release** branch a few times -- may have to run chgres_cube outside the app if you want to use certain features available only in **develop**

Overview of chgres_cube

- For those familiar with WRF, chgres_cube can be thought of as a combination of the “ungrib”, “metgrid”, and “real” executables
- Built around Earth System Modeling Framework (ESMF) and based on an older NCEP spectral GFS pre-processor (“global_chgres”)
- Reads in global or regional external model (atmospheric and surface) data from any one of a multitude of model sources (e.g., GFS, RAP, HRRR, etc.) in NetCDF, grib2, or nemsio format
- Interpolates external model data variables to the target grid for initialization of the FV3
- Writes out two NetCDF IC files for regional domains: surface and atmosphere
- If chosen, writes out one file w/atmospheric BCs

Static Inputs to chgres_cube

- Vertical coordinate definition file - *global_hyblev.l[levs].txt*
- Model grid mosaic file - [*CRES]_mosaic.nc*
- Model grid files - *[CRES]_grid.tile7.nc*
  - Grid point lat/lon; edge distance, area.
- Model orog files - *[CRES]_oro_data.tile7.nc*
  - Land-mask, orography, gravity wave drag fields.
- Surface climatological fields from make_sfc_climo
  - *[CRES]_facsf.tile7.nc* (fractional coverage strong/weak zenith angle albedo)
  - *[CRES]_maximum_snow_albedo.tile7.nc* (maximum snow albedo)
  - *[CRES]_slope_type.tile7.nc* (slope type - category)
  - *[CRES]_snowfree_albedo.tile7.nc* (snow-free albedo)
  - *[CRES]_soil_type.tile7.nc* (soil type - category)
  - *[CRES]_substrate_temperature.tile7.nc* (soil substrate T)
  - *[CRES]_vegetation_greenness.tile7.nc* (plant greenness)
  - *[CRES]_vegetation_type.tile7.nc* (vegetation type - category)

External Model Input Data

- Nemsio, Sigio, FV3GFS NetCDF, & grib2 supported, but grib2 is generally most readily available and most appropriate for regional applications. Some caveats:
  - nsst data & ice thickness/column temp not included, default values may be used
  - Some grib2 files have non-critical levels of some tracers (e.g., ozone near sfc, spfh near TOA) removed, treatment according to varmap table entry (discussed soon)
- Source models supported
  - Pgrb2 & pgrb2+bgrb2^ FV3GFS (^develop branch only)
  - GFS (FV3 and Spectral), NAM, RAP, and HRRR grib2 files
  - Other types (Spectral GFS sigio/sfcio (v12, v13)*; Spectral GFS gaussian nemsio (v14) *, FV3GFS gaussian nemsio (v15 - current OPS), FV3GFS gaussian NetCDF (v16+), FV3GFS tiled warm restart files (NetCDF) *, FV3GFS tiled history files (NetCDF)* (* HPSS only)

Online External Model Input Data

- Supported files can be acquired from a large variety of online sources
  - HPSS (extensive selection)
  - NOMADS (~ prior 10 days; gfs, hrrr, nam, rap folders at NOMADS)
  - NCEI archive (at least last 6 months online, sometimes more; NCEI)
  - AIRS archive (much older files must be requested; AIRS)
  - Cloud data storage (best source of HRRR data; AWS, Google)
- For exact file names of supported file types and external online links, see the SRW App release documentation:
  - Regional model data sources
  - Global model data sources
- Namelist setup for each input data, see this link:
  - https://noaa-emcufs-utils.readthedocs.io/en/ufs-v2.0.0/ufs_utils.html#regional-chgres-cube-namelist-options

Namelist options -- required

- *fix_dir_target_grid* - Path to the FV3-LAM surface climatological files (such as albedo).
- *fix_dir_input_grid* - Directory containing RAP lat/lon file. On NOAA HPC machines, typically the “fix/fix_am” directory of the UFS_UTILS directory.
- *mosaic_file_target_grid* - Path and name of the FV3-LAM mosaic file.
- *orog_dir_target_grid* - Directory containing the FV3-LAM orography and grid files (NetCDF).
- *orog_files_target_grid* - Names of the FV3-LAM orography file.
- *vcoord_file_target_grid* - Path and name of the model vertical coordinate definition file (“global_hyblev.l$LEVS.txt).
- *data_dir_input_grid* - Directory containing the GRIB2 initial conditions data
- *grib2_file_input_grid* - Name of the GRIB2 input data file (req. only for grib2)
- *varmap_file* - Path and name of the variable mapping (VARMAP) table. (req. only for grib2)
- *input_type* - Input data type. Valid options restart, history, gaussian_nemsio, gaussian_netcdf, grib2, gfs_gaussian_nemsio, gfs_sigio
- *cycle_mon/day/hour* - Month/day/hour of your model initialization
- *convert_atm* - Set to ‘true’ to process atmospheric fields
- *convert_sfc* - Set to ‘true’ to process surface fields

Namelist options -- required (cont.)

- *regional* (valid when *convert_atm* = .true.)
  - Set to 0 to create initial condition atmospheric file
  - Set to 1 to create initial condition atmospheric file and zero hour boundary condition file
  - Set to 2 to create a boundary condition file. Use this option for all but the initialization time.
- *halo_blend* - Integer number of row/columns to apply halo blending into the domain, where model and lateral boundary tendencies are applied (note that no actual blending is done in chgres_cube).
- *halo_bndy* - Integer number of rows/columns that exist within the halo, where pure lateral boundary conditions are applied.
- *external_model* - Name of source model for input data. Valid options: ‘GFS’, ‘NAM’, ‘RAP’, ‘HRRR’. (Default: ‘GFS’; req. only for grib2))

Namelist options -- optional

- *geogrid_file_input_grid* - Full path to the RAP or HRRR geogrid file corresponding to the external model input data. Only used with *external_model* = ‘HRRR’ or ‘RAP’.
- *nsoill_out* - Number of soil levels to produce in the sfc_data.nc file (Default: 4).
- *sotyp_from_climo* - Use soil type from climatology. Valid options: .true. or .false. (Default: .true.)
- *vgtyp_from_climo* - Use vegetation type from climatology. Valid Options: .true. or .false. (Default: .true.)
- *vgfrc_from_climo* - Use vegetation fraction from climatology. Valid options: .true. or .false. (Default: .true.)
- *lai_from_climo* - Use leaf area index from climatology. Valid options: .true. or .false. (Default: .true.)
- *minmax_vgfrc_from_climo* - Use min/max vegetation fraction from climatology. Valid options: .true. or .false. (Default: .true.)
- *tg3_from_soil* - Use tg3 from input soil. Valid options: .true. or .false. . Default: .false.
- *thomp_mp_climo_file* - Location of Thompson aerosol climatology file. Provide only if you wish to use these aerosol variables.

Variable Mapping (varmap) Table for GRIB2 Data

- The chgres_cube namelist contains an entry (“varmap_file”) to define the varmap table
- Controls how chgres_cube handles variables that may be missing from external model GRIB2 files
- Each varmap file contains three columns
  - Column 1: Variable names the code searches for in the GRIB2 files
  - Column 2: Variable names written to the chgres_cube output NetCDF files
  - Column 3: Behavior to follow (“skip”, “set_to_fill”, “intrp”^, or “stop”) when the code can’t find the variables in column 1 (^develop branch only)
  - Column 4: If column 3 = “set_to_fill”, then this value is used to fill in all points in the input field (note that certain variables *may* be overwritten by climatology, particularly the vegetation type, soil type, vegetation fraction, max/min vegetation fraction, and leaf area index surface fields)
  - Column 5: Variable type descriptor (“T” - 3D tracer, “D” - 3D non-tracer, “S” - 2D surface array)

Variable Mapping Table example

dzdt	dzdt	set_to_fill	0	D
delta_p	delp	skip	0	D
sphum	sphum	intrp	1E-7	T
liq_wat	liq_wat	intrp	0	T
o3mr	o3mr	intrp	1E-7	T
rainwat	rainwat	intrp	0	T
ice_wat	ice_wat	intrp	0	T
snowwat	snowwat	intrp	0	T
graupel	graupel	intrp	0	T
ice_nc	ice_nc	intrp	-1.0	T
rain_nc	rain_nc	intrp	-1.0	T
water_nc	water_nc	intrp	-1.0	T
liq_aero	liq_aero	intrp	0	T
ice_aero	ice_aero	intrp	0	T
sgs_tke	sgs_tke	intrp	0	T
vtype	vtype	skip	0	S
sotype	stype	skip	0	S
vfrac	vfrac	skip	0	S
fricv	uustar	skip	0	S

Output Files

- Atmospheric file (for initial time)
  - Surface pressure, temperature, winds, tracers, etc.
- Surface file (for initial time)
  - Fifty surface and Near Sea Surface Temperature (NSST) fields
- Boundary Conditions (for initial and subsequent times)
  - Atmospheric variables on the halo+blending zones on the edges of domain
  - 4 entries (n, s, e, w edges) for each variable type

External Model Data Processing in chgres_cube

- Processing of the atmospheric & surface (and NSST if not using grib2) fields are independent of each other
- Namelist options allow the user to determine whether to process all fields together or one at a time
- *convert_atm*, *convert_sfc*, and *convert_nsst* namelist settings can be defined as “true” or “false”
- Boundary conditions created if namelist entry *regional*=1,2 & *convert_atm*=true
- To create all LBCs for a given simulation, the SRW App workflow handles creation of atmospheric chgres_cube netCDF output automatically by running chgres_cube in a loop and modifying the namelist with the necessary settings

External Model Atmospheric Data Processing in chgres_cube

- Similar to the old spectral GFS program (‘global_chgres’)
- Horizontally interpolates from external model data input to FV3 model grids
  - For winds, ‘x’, ‘y’, and ‘z’ components are processed
- Adjusts surface pressure for terrain differences
- Reads the vertical coordinate definition file
- Computes 3D pressure
- Vertically interpolates to model hybrid levels.
- MP schemes supported by the **App** (via CCPP suite file): Thompson, GFDL, and Zhao-Carr
  - chgres_cube on its own can process whatever tracers you tell it to
  - Namelist option available to provide aerosols in chgres output for Thompson MP if unavailable in input data; handled automatically in the SRW App workflow based on external model data and CCPP suite definition file

External Model Surface Data Processing in chgres_cube

- Assumes Noah LSM
- Reads static surface climatology fields
  - User can set various *[var]_from_climo* namelist options to *false*, causing some variables (such as soil & vegetation type) from input data to override climatology (climo files still required). Make sure you know that your input soil (STATSGO) and vegetation (IGBP) classifications are correct
  - Be careful with *fhcyc* weather model namelist entry if any *[var]_from_climo* fields are *false*
- Interpolates surface fields from external model data input to the FV3 model grid
  - Account for masks - land to land; non-land to non-land; land ice to land ice.
- Adjusts soil temperature for terrain differences
- Computes frozen portion of total soil moisture
- Sets roughness for land and ice points (from vegetation type table)
- Rescales soil moisture for soil type differences
  - Attempts to preserve latent/sensible heat fluxes

Code Structure

- chgres_cube is built around Fortran modules
  - chgres.F90 - main driver routine
  - program_setup.F90 - reads namelist, computes soil parameters, sets up program execution
  - model_grid.F90 - sets ESMF grid objects for external model data input and FV3 model grids
  - static_data.F90 - reads FV3 model grid surface climatology data, interpolates in time
  - write_data.F90 - writes FV3 model grid NetCDF files
  - input_data.F90 - reads external model atmospheric and surface fields
  - utils.f90 - general utility routines
  - grib2_util.F90 - utility routines for GRIB2 data
  - surface.F90 - processes surface/NSST fields

UFS_UTILS Code Contributions

- Developers are welcome to tinker with any pre-processing utility code, but *only under their own fork* of the repository.
- Contributions back into the authoritative UFS_UTILS repository are welcomed, but must follow GitFlow protocols:
  - Create a ‘feature’ branch off ‘develop’ in the contributor’s fork of UFS_UTILS.
  - Ensure ‘develop’ under their fork stays up-to-date with the authoritative repository.
- Bug fixes and changes that support current or future NCEP operations will be given priority for inclusion into the authoritative repository
- Prior to code merges, unit tests (run automatically by Github) must pass, and a series of regression tests must be run on all NOAA officially supported machines - WCOSS, Orion, Jet ,and Hera.
- For more details, see the UFS_UTILS wiki page:
  - https://github.com/NOAA-EMC/UFS_UTILS/wiki

Documentation

- Additional information on chgres_cube exists online in relation to the SRW App release:
  - https://noaa-emcufs-utils.readthedocs.io/en/ufs-v2.0.0/ufs_utils.html