Creating and Installing Variable Resolution Spectral Element Grids in CESM

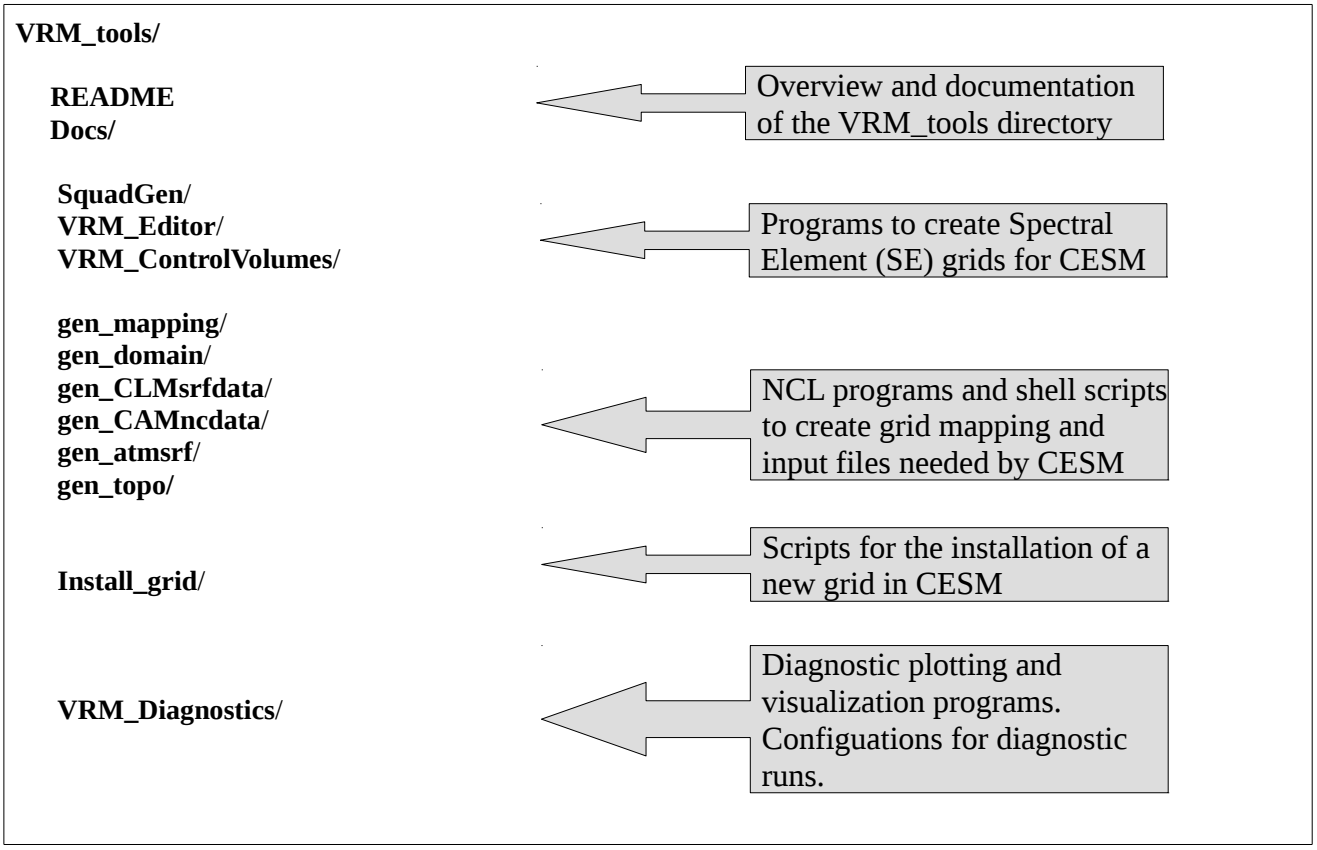
This document provides a description of the process of creating a new spectral element grid, generating the corresponding input files, and installing the new grid option into CESM. Once installed for a stable 5 day test run, the process of adjusting dynamical core parameters for long term stability and evaluating the performance of the new grid begins.

Before a new grid can be created, it is first necessary to obtain and compile some processing programs. These include programs in the the CLM/ATM tools directories contained in each CESM tag as well as programs in the VRM_tools toolkit. Once this is completed, there are three methods available for constructing new grids. *SQuadGen* is a command line program that constructs a grid refinement from a black and white PNG image that delineates the refined regions. The *VRM_Editor* is an interactive interface based on *SQuadGen* in which the user iteratively constructs a refinement map to define the variable resolution regions. The third option is *Create_VRMgrid*, which is a command line interface for the *VRM_Editor* for which the user provides a netcdf file containing the map of refined regions. The output from any of these programs is then fed into a series of processing programs which complete the construction of the final grid, generate the associated mapping and domain files, and interpolate the initial data values needed by CESM onto the new grid. The resulting grid and associated files are then installed into CESM for use.

Once installed, model parameters must be adjusted until satisfactory results are obtained. Initially this requires iteratively adjusting the time stepping/damping values until the model is stable for a 5-day test run. For an experienced user, once a new Spectral Element (SE) grid has been created using one of the three programs available, the process of creating all of the needed CESM input files and building a case that runs stably for the 5-day test can be completed in as short as two working days. Typically this time will vary as there will inevitably be problems that arise due to model development. From there further diagnostic testing for longer model runs should be carried out to fully test the new grid.

The steps from creating a new grid through installation are illustrated for an example case in which a NE30 resolution grid is refined to NE240 over mainland China. This example provides a recipe for new refinement grids. In general, once a new refinement grid is initially created, the steps involved should only differ in the names of the created files. However, since CESM is a continuously evolving developmental model and improved methodologies will likely emerge as users gain more experience using variable resolution grids, the steps documented here will likely need adjustment and updates to account for these changes. In the event that the documentation is out of sync with the processing steps, it should at a minimum provide a guide as to what needs to be done.

The following illustration gives an overview of the contents of the VRM_tools directory.



Preliminaries

Before creating a new grid it is first necessary to compile the requisite programs that will be needed. These are located in the tools directories for each tag and in the VRM_tools directory. Users should establish a local repository in which all of the grid and data files can be maintained and organized. In this directory the files for each new grid created are then contained in a separate sub-directory.

Check out the CESM tag that will be used

The directory in which the CESM tag is installed is referred to as $(CESM) in the remainder of this document. For the example case, a modified version the cesm2.1.0 release tag was used in which the limits restricting Spectral Element usage were removed. Subsequent tags will have the usage of Spectral Element grids fully enabled.

$(CESM) == *My Release Path*/cesm2.1.0/

Compile CESM prerequisite programs in CESM tools

When building and using CESM tools, the default modules that were loaded include:

```
cheyenne% module list

Currently Loaded Modules:
  1) ncarenv/1.3   2) intel/18.0.5  3) ncarcompilers/0.5.0
  4) netcdf/4.7.3  5) mpt/2.19
```

Users that have other modules loaded by default may encounter errors when building and using the tools programs. It is recommend that the current default versions of these modules be loaded when creating and installing VRM grids.

(1) gen_domains:

The gen_domains program reads in a conservative ocean/atmosphere mapping file and creates the land/ocean domain files with complementary masks delineating the fraction of each gridpoint that is associated with each domain. See the README and INSTALL files in the directory for a further description of the program and instructions on how to build it.

To build the executable on Cheyenne:

```
cheyenne% cd $(CESM)/cime/tools/mapping/gen_domain_files/src
cheyenne% ../../../configure --macros-format Makfile --mpilib mpi-serial
cheyenne% (source ./.env_mach_specific.csh; gmake)
```

(2) mksurfdata_map:

Given the mapping files between the desired output grid and the raw input data files, this program creates the “landuse.timeseries_*” and “surfdata_*” datasets needed for model simulations. See the README files in the directory for details about the program and how to build it.

To build the executable on Cheyenne:

```
cheyenne% module load ncl
cheyenne% setenv LIB_NETCDF $NCAR_LDFLAGS_NETCDF
cheyenne% setenv INC_NETCDF $NCAR_INC_NETCDF
cheyenne% cd $(CESM)/components/clm/tools/mksurfdata_map/src
cheyenne% gmake
```

(3) interpic:

Interpic is a program which interpolates horizontally and/or vertically from a given initial conditions dataset and outputs the values onto a user specified grid. See the README in the directory for details about the program and how to build it.

To build the executable on Cheyenne:

```
cheyenne% module load ncl
cheyenne% setenv LIB_NETCDF $NCAR_LDFLAGS_NETCDF
cheyenne% setenv INC_NETCDF $NCAR_INC_NETCDF
cheyenne% cd $(CESM)/components/cam/tools/interpic_new
cheyenne% gmake
```

Build SQuadGen (Optional)

SQuadGen is the original C++ program used to create variable resolution spectral element grids in CESM. It is included in the VRM_tools directory for users that prefer to rely on this program for existing grids.

To build the executable on Cheyenne:

```
cheyenne% module load gnu
cheyenne% cd $(VRM_TOOLS)/SQuadGen
cheyenne% make
```

Build VRM_Editor

VRM_Editor is a C++ program which provides an interactive interface to the *SQuadGen* program. The GUI was written using Qt and requires version 5.2 or greater of that library in order to compile. Since the program is interactive, it is best to run it on a local machine or laptop. The program will also build and run on Casper in the NCAR compute environment. For this case there is a lag in responsiveness to mouse motions when editing the grid, but it is not prohibitive. The process of compiling the editor is dependent on the desired compute environment.

(1) Linux environment:

All mainstream implementations of Linux have repositories available from which users can easily install the needed libraries. From these, install the precompiled Qt and netcdf libraries. The users may need to edit the VRM_Editor.pro file to provide the local directory where the netCDF libraries were installed.

Build the editor:

```
linux% <copy the VRM_Editor directory from $(VRM_TOOLS) to local machine>
linux% cd $(LOCAL_PATH)/VRM_Editor
linux% qmake-qt5 VRM_Editor.pro
linux% make
```

(2) Casper (NCAR) environment:

The VRM_Editor can be build and run on Casper. Though there will be a latency in the responsiveness of the application, depending on the users internet connectivity with the NCAR environment, a patient user can create a new variable resolution grid without much difficulty. It it necessary for the user to load the gnu compiler when building and using the VRM_Editor.

```
casper% <copy the VRM_Editor directory from $(VRM_TOOLS) to local machine>
casper% cd $(LOCAL_PATH)/VRM_Editor
casper% moudule load gnu/9.1.0
casper% qmake-qt5 VRM_Editor.pro
casper% make
```

(3) Mac environment:

The process of building the VRM_Editor for is not as straightforward. If one can manage to correctly install Xcode, Qt, and the netcdf libraries, the build steps are generally the same as for Linux though some environment variables may need to be modified before the build commands will work. Building the Qt and netcdf libraries from source as was done for Casper, worked in the Mac OS as well.

For Mac users whose usage consists of pointing, clicking, and typing, building the interactive editor will be a challenge. To avoid the difficulties of building the interactive program on a Mac, a precompiled executable is provided in the VRM_tools. Thus far, this executable has worked fine on all machines tested.

```
mac% <copy VRM_Editor.app directory from $(VRM_TOOLS)/VRM_Editor/MacOS>
mac% <point at executable... and then click..>
```

The editor was developed using a normal 3-button mouse, so rather than zooming in using the mouse wheel for example, Mac users will have to use equivalent inputs for that operation.

Build Create_VRMgrid (Optional)

Create_VRMgrid is a command-line interface to the *VRM_Editor* analogous to the *SQuadGen* program. The source resides in the same directory as the VRM_Editor and only requires the standard netcdf C libraries. Because Qt uses and overwrites the Makefile, there is a separate Makefile-Create_VRMgrid to compile this program.

```
computer% cd $(LOCAL_PATH)/VRM_Editor
computer% cp Makefile-Create_VRMgrid Makefile
computer% vi Makefile (set netcdf lib paths)
computer% make depend
computer% make
```

Build Gen_ControlVolumes Program

Gen_ControlVolumes is a stand-alone FORTRAN program located in the VRM_tools directory. The program further divides each element into NPxNP gridpoints and calculates the corresponding area weights and connectivity information for the resulting grid. Note that currently the **gnu** compiler must be used to build and run this executable.

```
cheyenne% cd $(VRM_TOOLS)/VRM_ControlVolumes/src
cheyenne% module load gnu
cheyenne% make
```

Build Topography Generation Software

The programs to generate topography a topography file for a given variable resolution grid resides in the GitHub repository, https://github.com/NCAR/topo. Users should download the download the source from this location and follow the relevant instructions on how to build and use the programs.

```
cheyenne% cd $(VRM_TOOLS)/gen_topo
cheyenne% git clone https://gihub.com/NCAR/topo topo
cheyenne% cd topo

<Follow instruction in topo to build necessary executables and load the
relevant python modules>
```

Build ESMF Tools (if necessary)

The scripts to interpolate data values onto the variable resolution mesh use tools from the Earth System Modeling Framework(ESMF), which are currently accessed via NCL. CESM is in the process of transitioning to NUOPC, The National Unified Operational Prediction Capability, as a standard way of building and connecting model components. Since ESMF uses a custom unstructured grid format rather than the SCRIP format, users must use the *ESMF_Scrip2Unstruct* program to create a MESH file for each of their SCRIP files. Users who do not have ESMF available in their compute environment must download the source from the ESMF website and follow the instructions there to build this tool. Users with access to the NCAR compute environment can access these tools after loading appropriate ESMF modules.

Create a repository location for the VR grid data and input files

Each new grid requires a set of mapping, domain, and associated data files. The processing scripts store all of these files in a common directory that must be specified. Let $(REPO) refer to this directory. Users need to create this repository directory to store and maintain all of their VRM grids. Each new grid will reside in a separate directory which contains all of the associated files that are generated.

The convention for naming new variable resolution grids is (currently NP is always 4):

ne0np4.GRIDNAME**.ne**ZZ**x**RR

where GRIDNAME is a descriptive name for the grid, ZZ is the base resolution of the grid, and RR indicates the level of refinement (the refined resolution is ZZ*RR). Make a directory for each new grid:

$(REPO) == *My_Path*/VRM_files/

```
cheyenne% mkdir $(REPO)/ne0np4.GRIDNAME.neZZxRR
```

For the China refinement example grid illustrated in the remainder of this document, create the directory:

```
cheyenne% mkdir $(REPO)/ne0np4.China01.ne30x8
```

Incorporating an index into the GRIDNAME is a good practice since revised versions of a refined grid often need to be created.

Creating a Variable Resolution Mesh

This section covers the process of creating a new variable resolution spectral element grid. The first step is to use one of the three possible programs to create finite element mesh that spans the surface of the globe. Without refinement these programs divide each face of a cube into an (NExNE) grid and map the resulting gridpoints onto the unit sphere. The netCDF file they produce contains the grid description of this cubed sphere grid and is stored in what is referred to as an EXODUS file. In terms of the spectral element grid nomenclature (i.e. ne30np4) this file contains the 'NE' part of the grid. The second step in creating a spectral element grid is to run the *Gen_ControlVolumes* program which further subdivides each of the (6xNExNE) quadrilateral elements with an (NPxNP) grid and creates two separate output files describing the resulting grid, a SCRIP file and a LATLON file. The SCRIP file format was originally established for the Spherical Coordinate Remapping and Interpolation Package. It contains grid coordinates in terms of cell centers and an associated set of cell corner points that trace out the edges of the cell. The LATLON file contains the (Lat,Lon) coordinates and associated area weights for the set of unique gridpoints, the information typically found in a CESM history file. For grids with refinement, the refinement is carried out by varying the number of 'NE' elements in the EXODUS file before the NP gridpoints are added by *Gen_ControlVolumes*. At the end of this section, some examples of generated variable resolution grids are provided for clarity.

The SQuadGen Program

The original command line program to create a variable resolution grid, *SQuadGen* uses a rectangular black and white PNG image as input to define a region of refinement. The horizontal pixels are associated with Longitude values and the vertical with Latitude. Using this as a guide, the user delineates the desired geographic regions for refinement in an image editor. Options on the command line control the methods for generating a resulting grid from the given image file. Running the program with no arguments will provide a listing of the available options.

```
computer>./SQuadGen
Parameters:
  --grid_type <string> ["CS"] (Options: ICO | CS)
  --refine_type <string> ["LOWCONN"] (Options: LOWCONN | CUBIT | LOWCONNOLD)
  --refine_level <integer> [2]
  --resolution <integer> [10]
  --refine_file <string> [""]
  --output <string> [""]
  --loadcsrefinementmap <bool> [false]
  --smooth_type <string> ["NONE"] (Options: NONE | SPRING | PRESSURE)
  --smooth_dist <integer> [1] (Smooth distance, -1 = smooth entire mesh)
  --smooth_iter <integer> [10]
  --lon_base <double> [-180.000000]
  --lat_base <double> [0.000000]
  --x_rotate <double> [0.000000]
  --y_rotate <double> [0.000000]
  --tessellate <integer> [0]
  --subcellres <integer> [0]
  --invert <bool> [false]
  --block_refine <bool> [false]
----------------------------------------
./SQuadGen: No output file specified
```

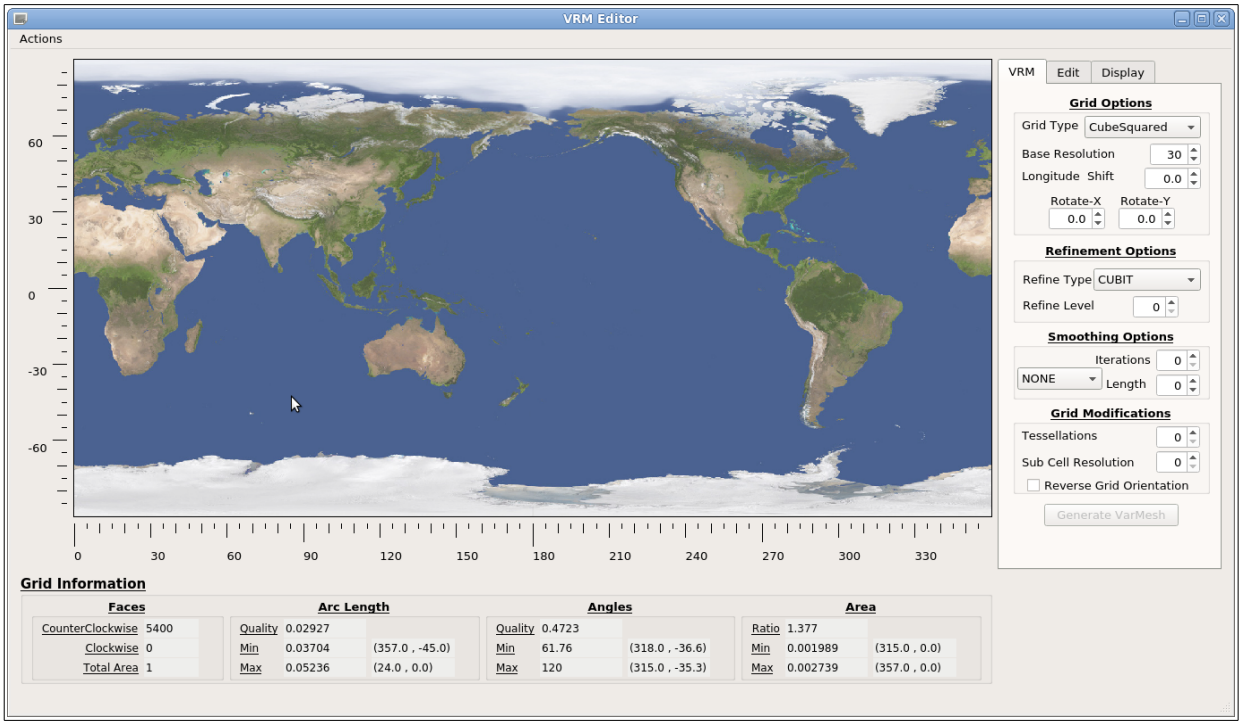
The README file in the program directory provides a more detailed description of the program inputs.

Upon completion, in addition to creating an EXODUS file containing the resulting grid, the program produces a refine_map.dat file. This is an ASCII file which contains discrete refinement levels for the cube faces. These values can be subsequently edited and iteratively passed to SQuadGen to correct problematic elements. As illustrated, after the first execution, a file refine_map.dat is created. This ASCII file is then edited to adjust the refinement as desired. Then a subsequent run with the -loadcsrefinementmap flag will cause the program to read the refine_map.dat file and incorporate the modification into the output grid. Subsequent edits can then be made until the user is satisfied with the resulting grid.

```
computer> ./SQuadGen --refine_file refinement.png --refine_level 2 --refine_type
CUBIT --resolution 30 --smooth_type NONE --output newmesh.g

computer> vi refine_map.dat  (modify refinement levels as desired)

computer> ./SQuadGen --refine_file refinement.png --refine_level 2 --refine_type
CUBIT --resolution 30 --smooth_type NONE --output newmesh.g -loadcsrefinementmap
```

The VRM_Editor

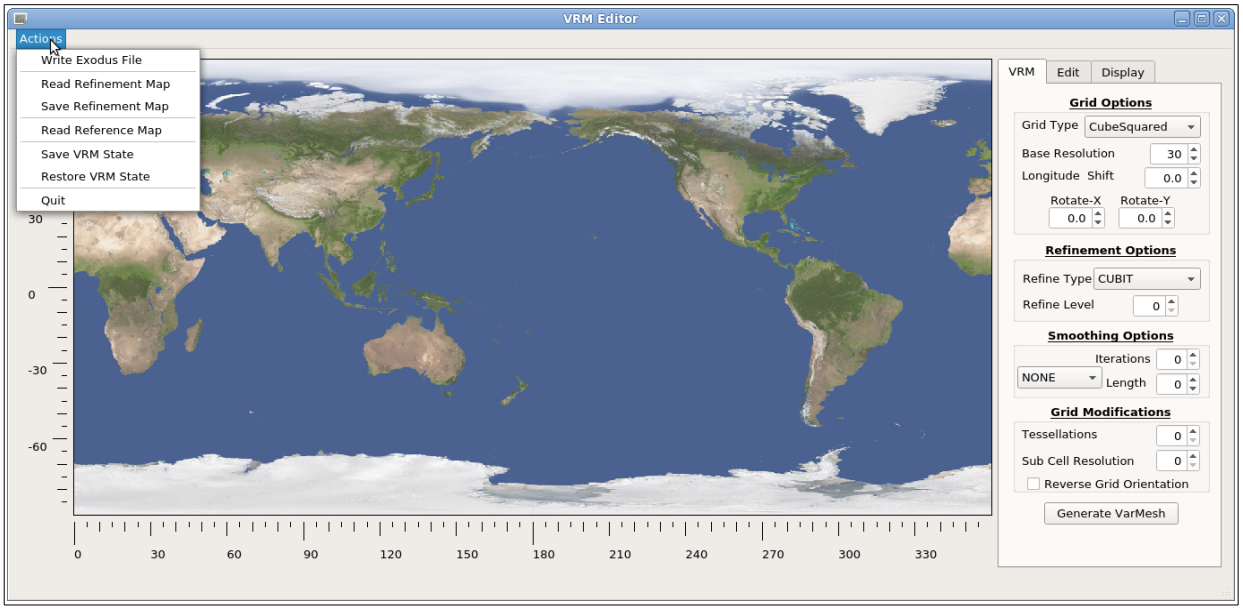
VRM_Editor is a graphical user interface for *SQuadGen* which is used to interactively edit and create unstructured grids with varying resolution over specified regions. Rather than defining a region via a PNG image, the editor is used to create a reference map with continuous values between 0 and 1, where 0 corresponds to the base resolution region, and 1 corresponds to the region of maximum refinement. At start-up the VRM_Editor displays a a map of the Earth with an ***Actions*** menu at the upper left and 3 tab menus on the right hand side.



At the bottom of the screen, there is a table containing information about the grid created. When a parameter is changed, the displayed grid is no longer valid. To alert the user to this case, the table of grid values is faded (deactivated) to indicate that the values may no longer be valid. The left column contains a count of the faces with a clockwise/counterclockwise orientation, and a normalized sum of the face areas. For each of the quantities: area, arc length, and element angles, the min/max values as well as their geographic position are displayed. There can be multiple occurrences of these extreme values,the displayed location is just the first occurrence. The area ratio value represents is just the fraction of the global min/max area values. The quality values for arc length and area are defined as

$$Q_{ang} \equiv \frac{1}{4\,N_{Faces}} \sum_{i=1}^{N_{Faces}} \left(1 - \frac{\alpha_{min}(i)}{\alpha_{max}(i)}\right) \qquad Q_{arc} \equiv \frac{1}{4\,N_{Faces}} \sum_{i=1}^{N_{Faces}} \left(1 - \frac{L_{min}(i)}{L_{max}(i)}\right)$$

where the max/min are the largest/smallest values of each quantity for a given face.



The ***Actions*** menu contains options to read and write refinement information as well as the EXODUS file containing the grid information for the newly constructed grid. The ***VRM*** tab contains the user controls for the input values to *SQuadGen*. The ***Edit*** tab contains controls for constructing a refinement map with values in the range [0,1], in which 0 values correspond to regions at the base resolution and values of 1 correspond to the high resolution region. The ***Display*** tab contains user controls which determine what values are optionally displayed.

The view can be zoomed in for a closer look at grid details using the mouse wheel or equivalent control. While zoomed in the map can be scrolled up/down/left/right using the arrow keys. Note that the focus needs to be in the map area, so a mouse click may be needed before the scroll arrows become responsive. Finally, the view can be reset to the default using the ESC key.

The Actions Menu:

- Write Exodus File

Once the user is satisfied with a configured grid, the results are saved in an EXODUS file. This file contains the grid information of the NE elements only. The naming convention for this file has the form:

"**GRIDNAME**_ne**ZZ**x**RR**_EXODUS.nc"

where

GRIDNAME = a unique descriptive name for the grid
ZZ = The Base resolution of the grid
RR = Indicated the level of refinement
(For an ne30 base grid refined to ne120, RR = (120/30) = 4.)

The EXODUS file created by *SQuadGen* contains the required information about the finite element mesh. EXODUS files created by *VRM_Editor* or *Create_VRMgrid* contain additional information (shown in red) regarding how the grid was generated as well as the relevant information about the grid.

```
netcdf EXODUS_file {
dimensions:
            len_string = 33 ;
            len_line = 81 ;
            four = 4 ;
            time_step = UNLIMITED ; // (0 currently)
            num_dim = 3 ;
            num_nodes = 5402 ;
            num_elem = 5400 ;
            num_el_blk = 1 ;
            num_qa_rec = 1 ;
            num_el_in_blk1 = 5400 ;
            num_nod_per_el1 = 4 ;
            num_att_in_blk1 = 1 ;
            ref_lat = 360 ;
            ref_lon = 720 ;
            StrLen = 12 ;
variables:
            double time_whole(time_step) ;
            char qa_records(num_qa_rec, four, len_string) ;
            char coor_names(num_dim, len_string) ;
            char eb_names(num_el_blk, len_string) ;
            int elem_map(num_elem) ;
            int eb_status(num_el_blk) ;
            int eb_prop1(num_el_blk) ;
            eb_prop1:name = "ID" ;
            double attrib1(num_el_in_blk1, num_att_in_blk1) ;
            int connect1(num_el_in_blk1, num_nod_per_el1) ;
            connect1:elem_type = "SHELL4" ;
            double coord(num_dim, num_nodes) ;
            double ref_lat(ref_lat) ;
            double ref_lon(ref_lon) ;
            double refineMap(ref_lat, ref_lon) ;
            refineMap:long_name = "refinement map" ;
            refineMap:description = "Lat,Lon map of [0.=low res,1.=hi res] refinement values" ;
            int VRM_Resolution ;
            int VRM_Refinement ;
            int VRM_Tessellations ;
            int VRM_SubCellResolution ;
            int VRM_TransSmoothDist ;
            int VRM_SmoothIterations ;
            double VRM_GridXRotate ;
            double VRM_GridYRotate ;
            double VRM_GridShiftLon ;
            char VRM_ReverseOrientation ;
            char VRM_RefineType(StrLen) ;
            char VRM_GridType(len_string) ;
            char VRM_SmoothType(len_string) ;
            int GRID_Num_Clockwise ;
            int GRID_Num_Counter ;
            double GRID_Total_Area ;
```

```
            double GRID_Area_Ratio ;
            double GRID_MinArea ;
            double GRID_MinArea_Lon ;
            double GRID_MinArea_Lat ;
            double GRID_MaxArea ;
            double GRID_MaxArea_Lon ;
            double GRID_MaxArea_Lat ;
            double GRID_Qarc ;
            double GRID_MinArc ;
            double GRID_MinArc_Lon ;
            double GRID_MinArc_Lat ;
            double GRID_MaxArc ;
            double GRID_MaxArc_Lon ;
            double GRID_MaxArc_Lat ;
            double GRID_Qang ;
            double GRID_MinAng ;
            double GRID_MinAng_Lon ;
            double GRID_MinAng_Lat ;
            double GRID_MaxAng ;
            double GRID_MaxAng_Lon ;
            double GRID_MaxAng_Lat ;

// global attributes:
            :api_version = 4.98f ;
            :version = 4.98f ;
            :floating_point_word_size = 8 ;
            :file_size = 0 ;
            :title = "VRM_Editor(/home/VRM_Editor/EXODUS_file.g) " ;
}
```

- Read Refinement Map

  The user can read in an existing refinement map which was either previously saved from the *VRM_Editor* or alternatively created by an external program. The map is stored on a uniform lat/lon grid with values standardized between [0,1] in a netCDF file. Any netCDF file with longitude, latitude coordinates and a variable refineMap containing values in the range [0.,1.] can be read in and used as a refinement map. The longitude values must be in the range [0.,360.] and the nominal dimensions are [720,360]. These dimensions can be increased, but the *VRM_Editor* tends to become less responsive for interactive editing as this resolution is increased.

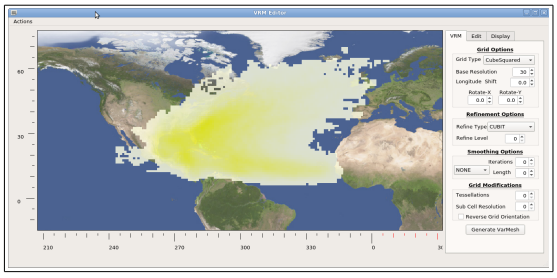
```
netcdf refmap_file {
dimensions:
        latitude = 360 ;
        longitude = 720 ;
variables:
        double latitude(latitude) ;
        double longitude(longitude) ;
        double refineMap(latitude, longitude) ;
}
```

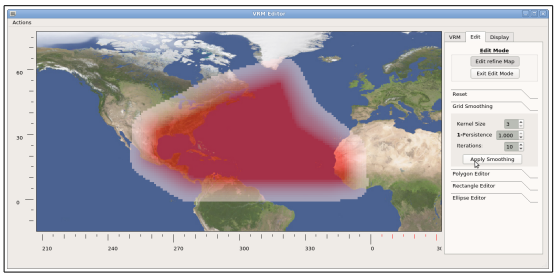
- Save Refinement Map

  The user is prompted to save the current refinement map in a netCDF file for later use.

- Read Reference Map

  The user is prompted to read in a reference map from a netCDF file. A reference map is similar to a refinement map except that values are not required to span the range [0,1], and the editor cannot modify these values. This map, displayed in yellow by default, can be used as a guide when constructing a refinement region associated with some physical phenomena. An example is the PDF of tropical storm tracks in the Atlantic basin shown in yellow. Using this as a guide in the editor, a refinement map is constructed to encompass this region using the polygon editor. Then after applying the smoothing operator the result, shown in red, is a refinement domain tailored to hurricane studies in the Atlantic basin.





The netCDF file format for the reference map is exactly the same as for the refinement map. The only difference is that the values contained in the refineMap variable are not constrained to the [0.,1.] interval. This provides the added flexibility to be able to use a previous refinement map as reference when constructing a revised version of a given grid.

```
netcdf TC_reference {
dimensions:
        latitude = 360 ;
        longitude = 720 ;
variables:
        double latitude(latitude) ;
        double longitude(longitude) ;
        double refineMap(latitude, longitude) ;
}
```

- Save VRM State *(** not implemented yet **)*

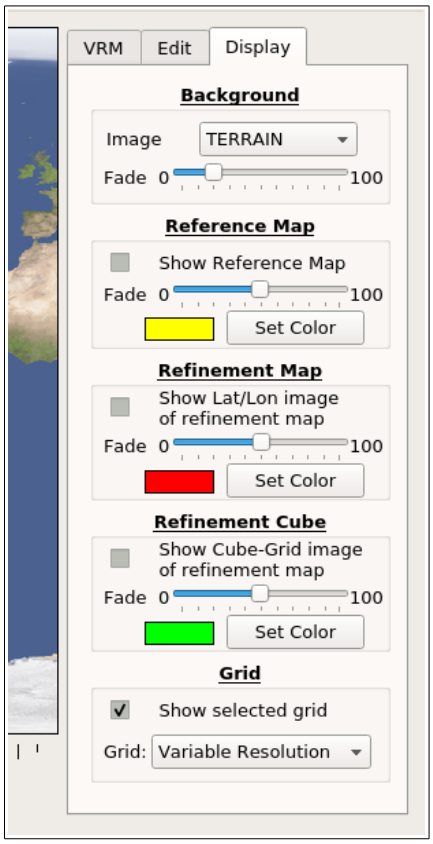
  Save the current state of the VRM_Editor completely.

- Restore VRM State *(** not implemented yet **)*

  Read in previously saved state so that the user can pick up where they left off.

- Quit

The Display Tab:



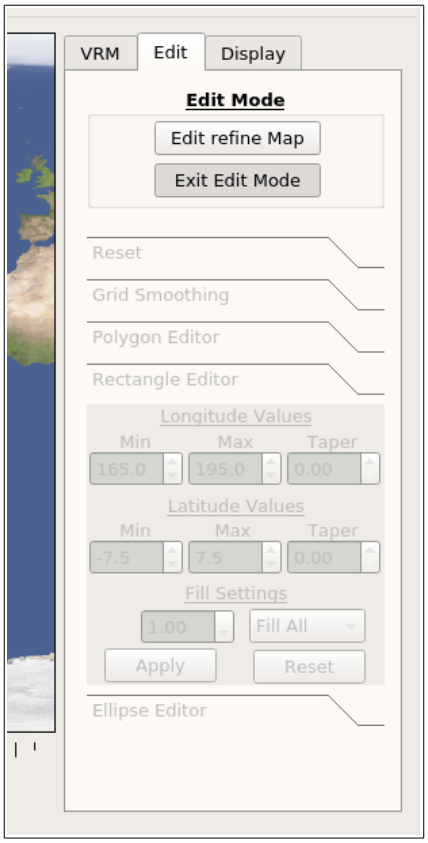
This tab contains options which control the displayed fields. The Background image option can be either the default terrain map, a land/water map, or neither.

The Reference Map, Refinement Map, and Refinement Cube values can each be toggled on/off, and each has a slider which can be used to vary their transparency. To accommodate the fact that colors can be perceived differently, each has an option to change the default color used. *(** not implemented yet **)*.

In *SQuadGen*, the refinement values are converted to discrete integer values for elements on a grid with resolution one refinement level lower then the highest resolution region. These integer values are displayed in the Refinement Cube map, and are the actual values used to subdivide the elements to higher resolutions. It is sometimes useful to view these values to understand the appearance to the resulting refined grid.

The Grid option allows the user to toggle the display of the grid on/off and to optionally display the variable resolution grid or the corresponding base grid without refinement.

Note that the display options are dependent on the current Edit mode, so the displayed fields will change as the editor enters/leaves edit mode.



The Edit Tab:

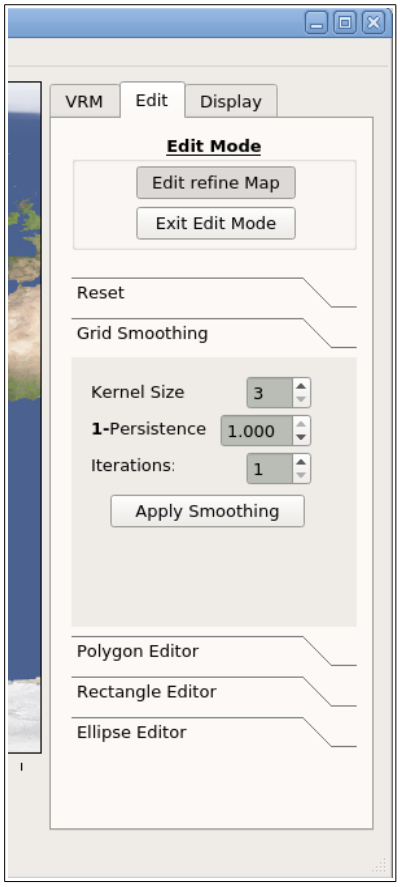
When the Edit tab is initially selected, the toolbox of editing options is disabled. The Edit refine Map button is used to enter edit mode in which the user can modify the current refinement map values. The toolbox contains a set of editing options:

- Reset
- Grid Smoothing
- Polygon Editor
- Rectangular Editor
- Ellipse Editor

Once the user is satisfied with the constructed refinement map, selecting the Exit Edit Mode button will exit edit mode and the user will be prompted to either save or discard the changes to the refinement map values.

The **Reset** tool (not shown) offers a single button which will clear the current refinement map by setting all values equal to 0.

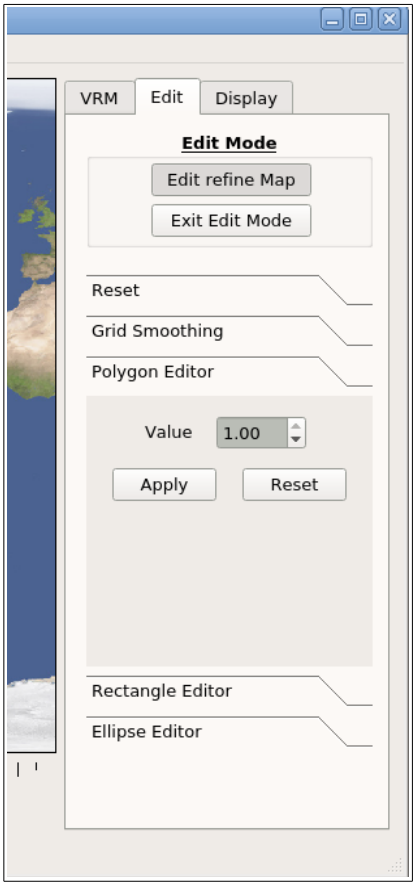
The **Grid Smoothing** tool is used to iteratively apply a smoothing kernel to the refinement map values in order to reduce discontinuities in the map values. These most typically occur when creating regions using the polygon editor tool, where values inside the polygonal region are set to a specified value and those outside are unchanged.



The 1-Persistence value is used to preserve regions where the refinement map values are equal to 1. Without this option, the repeated smoothing operations would act to erode the refinement region around the boundaries. The resulting refinement grid typically does not encompass the users desired high resolution region.

With polygonal defined regions, there are instances where vertices produce relatively sharp angles in the high resolution refinement defined by values of 1. Setting the 1-Persistence value to a value slightly less than 1.0000 will allow the smoothing operator to taper the values along these edges without adversely affecting the refinement region.

When the **Polygon Editor** tool is selected a rectangle appears with 8 vertices that can be dragged and place with the mouse. If the display is zoomed in, this may not be visible. Clicking of the reset button will center the polygon in the current display.

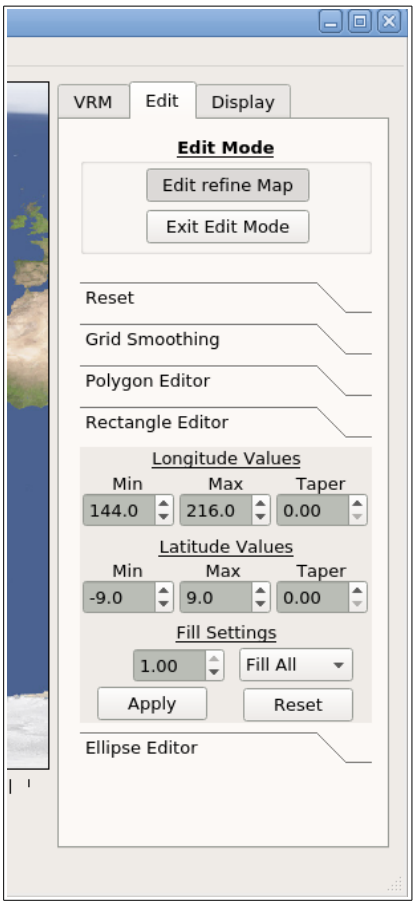


Using the mouse, the vertices can be placed to define a polygonal region for refinement. The entire polygon can also be selected and moved with the mouse. When the apply button is selected, values inside the polygonal region are set to the specified value. By default, the values inside the polygon are only updated if the current value in of the reference map is less than the specified value. This enables the user to define their desired high resolution region and set the values there equal to 1. Then the user can move the vertices to define a larger region around this refinement and set refinement values less than 1. Continuing with successively smaller values, the user can define the transition regions from the base resolution to their high resolution refinement.

e.g.

For a refinement from NE30 to NE480, the user can define an NE480 region with values of 1.0 embedded in regions of NE240, NE120, and NE60 with refinement values of 0.75, 0.50, and 0.25 respectively.

When the **Rectangle Edit** tool is selected a black rectangle appears on the map. As with the polygon editor, if the display is zoomed in, the rectangle may not be visible. Clicking on the reset button will center the rectangle in the current display. The rectangle cannot be selected using the mouse, the Latitude and Longitude ranges for the rectangle are set in the appropriate boxes. Note that there are no corrections if the user decides to set a minimum value greater than the maximum value for either Latitude or Longitude. Do this and something bad will probably happen.



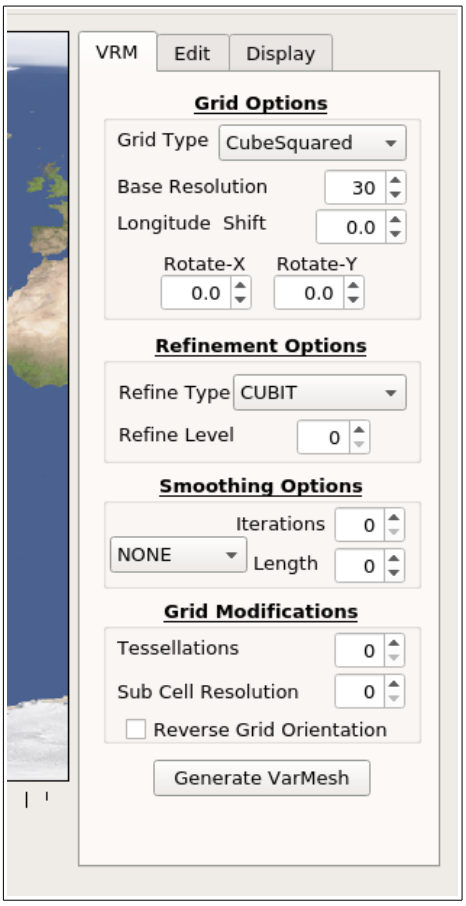
To accommodate rectangles that cross the Greenwich meridian, the range of Longitude values span [-180, 540], and to accommodate the creation grids with polar refinements, the range of Latitudes span the range [-135,135]. The refinement map values outside the range [-90.,90.] are not used. (See the polar refinement example below).

The taper values provide a means of adding a smooth transition from the fill value to 0 around the rectangular region. When the taper value for Longitude is increase from 0, edges of green rectangles appear on the left/right to indicate the transition region. Similarly, when the latitude taper is increased from 0, the top/bottom edges appear to indicate the transition region.

When the Apply button is selected, the values inside the rectangle are updated using the specified value. Here there are two options on how the values are updated. The **Fill All** option will update all refinement map values based on the selected rectangle, the **Fill Max** option will only replace smaller refinement map values using the rectangle settings. With this option, the user can define multiple rectangular regions as they desire.

The **Ellipse Edit** tool, to create circular or elliptical refinement regions has not yet been implemented. It may not be needed as it is possible to create similar regions using the rectangle edit tool with relatively large taper values.

The VRM Tab:



The VRM tab contains all of the optional input values for the underlying *SQuadGen* program. When the Generate VarMesh button is invoked, the current settings are passed to *SQuadGen*, the corresponding grid is calculated based on the current refinement map values, and the result is displayed. Once the grid is calculated, the Generate VarMesh button is disabled until a setting is changed that requires that the grid be recalculated.

One significant difference between the VRM_Editor interface and the command-line SQuadGen program is that grid rotations are carried out differently. Rather than applying the grid rotation values after the refinement has been calculated, for the VRM_Editor, the rotations are applied before the grid refinement is done. This has the effect of rotating the base grid relative to the refinement region, rather than rotating a calculated grid refinement into some desired position. It is often the case that along edges between cube faces and near vertices of the cube, the refinement process can lead to skewed elements with relatively short distances between gridpoints. Since the longest stable time step that is possible is limited by this distance, if possible, it is desirable to be able to position the grid so that a given refined region is in the center of a cube face.

The Grid Options include a selection of **Grid type**, the **Base Resolution** of the grid, and angles for 3D rotations of the base grid. There a two optional grid types, CubeSquared and Icosahedral. The Icosahedral option works to generate a base icosahedral grid, but applying any of the refinement options will cause the program to fail. CubeSquared is the appropriate option for variable resolution spectral element grids. The default base resolution is 30 for an NE30 grid.

The Refinement Options control the resolution of the refined region and the method used to connect the regions with differing resolutions. The **Refine Level** option specifies the high resolution region as a power of 2 multiple of the base resolution.

$$NE_{high} = NE_{base} * 2^{refine\,level}$$

For the **Refine Type**, there are three options, CUBIT, LOWCONN, and LOWCONNold. The CUBIT option connects the two regions using templates that span only one of the lower resolution grid elements, while the LOWCONN and LOWCONNold are variations of a method that connects the regions using templates that span two of the lower resolution elements. The LOWCONN options are generally preferred since they increase the length of the transition region, resulting in quadrilaterals with larger acute angles and that are less distorted. However, there are instances where these options lead to spurious behavior in the refinement. For these cases, the CUBIT option combined with smoothing options can produce a more appropriate grid refinement for the users needs.

There are currently three Smoothing Options available, each with an optional number of **Smoothing Iterations** to apply, and a parameter which sets a **Length** scale for the smoothing. A fourth option is under development and is not yet available for general use.

NONE	No smoothing
SPRING	Gridpoints near the transitions are adjusted using spring-dynamics mesh smoothing on the refined mesh.
PRESSURE	Some other variation of the SPRING smoothing? Experimental?
REFMAP	For each iteration, the current refinement map values are interpolated and used to determine the reference area values surrounding each gridpoint. The position of each gridpoint is then moved so as to adjust the actual area values toward the reference area values. *(** not implemented yet **)*

The Grid Modification options are not applicable to creating variable resolution spectral element grids so it is best to just leave these values alone.

The Create_VRMgrid Program

Another alternative to running the interactive editor is the *Create_VRMgrid* program. It is a command-line program contained in the VRM_Editor directory which does not require the Qt libraries. It is similar to the *SQuadGen* program except that the rotation options correspond to those in the *VRM_Editor* and it requires a netCDF file containing refinement map values rather than a PNG image. Running the program with no argument will provide a listing of the available options. The refine_file can either be saved from the *VRM_Editor* or constructed by some other means. The only requirement is that the netCDF file have the format outlined above. The EXODUS file to be created is specified by the -output value.

```
computer>./Create_VRMgrid

Input Parameters:
  --refine_type <string> ["CUBIT"] (Options: CUBIT | LOWCONN | LOWCONNold)
  --grid_type <string> ["CubeSquared"] (Options: Icosahedral | CubeSquared)
  --smooth_type <string> ["NONE"] (Options: NONE | SPRING | PRESSURE)
  --resolution <integer> [30]
  --refine_level <integer> [0]
  --tessellate <integer> [0]
  --subcells <integer> [0]
  --smooth_dist <integer> [0]
  --smooth_iter <integer> [0]
  --reverse_orient <bool> [false]
  --x_rotate <double> [0.000000]
  --x_rotate <double> [0.000000]
  --lon_shift <double> [0.000000]
  --refine_file <string> ["NULL"]
  --output <string> ["NULL"]

./Create_VRMgrid: No output file specified
```

Generate Control Volumes

Once an EXODUS file has been created using one of the three methods above, it should be moved into the local repository located in the directory $(REPO).

```
cheyenne% mkdir $(REPO)/ne0np4.GRIDNAME.neZZxRR/grids
cheyenne% mv GRIDNAME.neZZxRR_EXODUS.nc $(REPO)/ne0np4.GRIDNAME.neZZxRR/grids/
```

Given an EXODUS file, the *Gen_ControlVolumes* program subdivides each element with (NPxNP) points and calculates the control volumes and related information. The program generates the final grid information in two formats, SCRIP and LATLON. The SCRIP file contains the position and associated area for a set of grid cells as well as a corresponding set of corner points that trace out the edges of each cell.

```
netcdf GRIDNAME_ne30x8_np4_SCRIP {
dimensions:
        grid_size = 251534 ;
        grid_corners = 12 ;
        grid_rank = 1 ;
variables:
        int grid_dims(grid_rank) ;
        int grid_imask(grid_size) ;
        double grid_area(grid_size) ;
                grid_area:long_name = "area weights" ;
                grid_area:units = "radians^2" ;
        double grid_center_lat(grid_size) ;
                grid_center_lat:units = "degrees" ;
        double grid_center_lon(grid_size) ;
                grid_center_lon:units = "degrees" ;
        double grid_corner_lat(grid_size, grid_corners) ;
                grid_corner_lat:units = "degrees" ;
        double grid_corner_lon(grid_size, grid_corners) ;
                grid_corner_lon:units = "degrees" ;

// global attributes:
                :Grid = "Variable Resolution: GRIDNAME_ne30x8_EXODUS.nc" ;
                :Created\ by = "Gen_ControlVolumes.exe" ;
}
```

The LATLON file provides the Longitude, Latitude positions and associated area weights for the set of unique gridpoints. These are the grid values typically found in CESM history files, however, unlike the history files, this file also contains an array of *element_corner* indices for each quadrilateral of the grid. This connectivity information is needed to properly render the data values on the unstructured grid. Unfortunately the CESM history files do not currently adhere to the concept of a self describing dataset and this vital information is omitted.

```
netcdf GRIDNAME_ne30x8_np4_LATLON {
dimensions:
        ncol = 251534 ;
        ncorners = 4 ;
        ncenters = 251532 ;
variables:
        double lat(ncol) ;
                lat:long_name = "column latitude" ;
                lat:units = "degrees_north" ;
        double lon(ncol) ;
                lon:long_name = "column longitude" ;
                lon:units = "degrees_east" ;
        double area(ncol) ;
                area:long_name = "area weights" ;
                area:units = "radians^2" ;
        int element_corners(ncorners, ncenters) ;

// global attributes:
                :Grid = "Variable Resolution: GRIDNAME_ne30x8_EXODUS.nc" ;
                :Created\ by = "Gen_ControlVolumes.exe" ;
}
```

For the given EXODUS file, the Gen_ControlVolumes program will create files following the naming convention:

"**GRIDNAME**_ne**ZZ**x**RR**_np**P**_SCRIP.nc"
"**GRIDNAME**_ne**ZZ**x**RR**_np**P**_LATLON.nc"

where GRIDNAME, ZZ, and RR are as for the corresponding EXODUS file, and **P** = NP. Currently the models are only configured for a value of NP=4.

To generate the SCRIP and LATLON files, edit the 'input_nl' file in the Gen_ControlVolumes directory to set the path to the grid files in the local $(REPO) directory and the name of the grid. Note that the value of $(REPO) indicated here should be the full pathname and that the *GridName* value is just the root grid name without the '_EXODUS.nc' at the end. When completed the program should have created the SCRIP and LATLON files for the grid in the $(REPO)/grids directory.

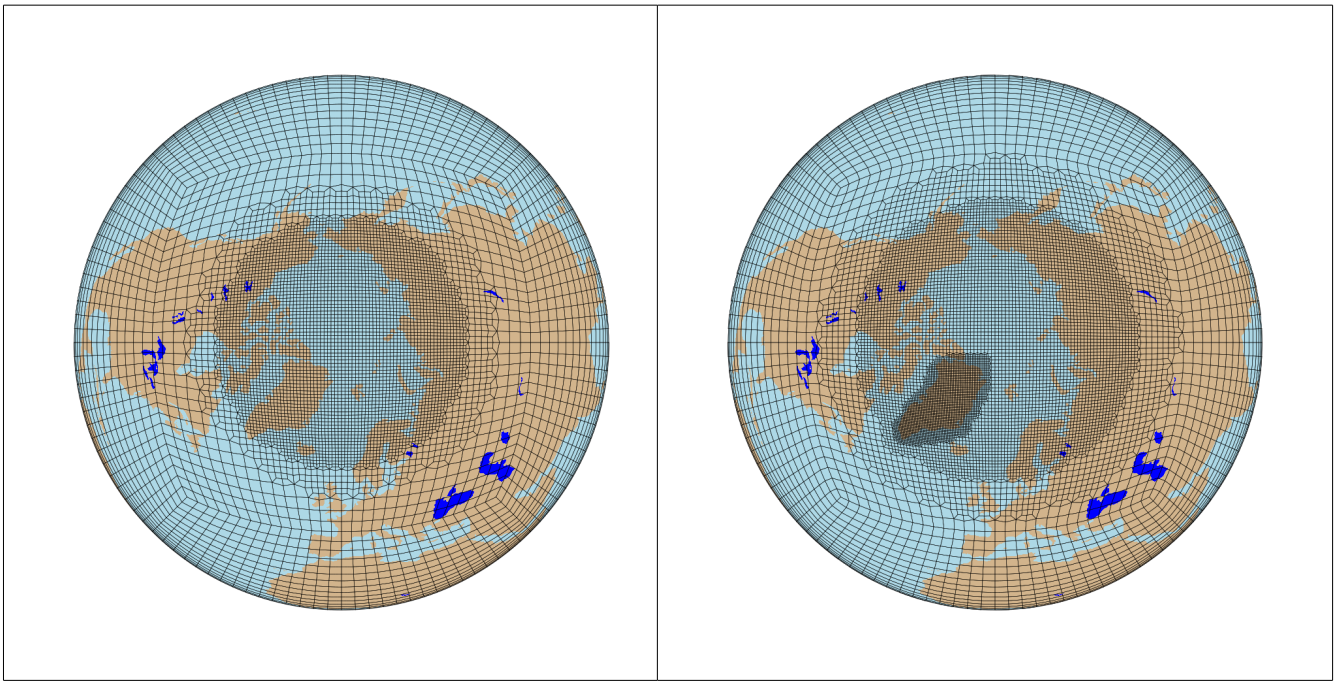
```
cheyenne% cd $(VRM_TOOLS)/VRM_ControlVolumes/src
cheyenne% vi input_nl

      set the input_nl variables:
         GridPath = '$(REPO)/ne0np4.GRIDNAME.neZZxRR/grids/'
         GridName = 'GRIDNAME_neZZxRR'

cheyenne% ./Gen_ControlVolumes.exe input_nl
```

Some Examples

A Polar Refinement grid:

As an example, consider a refinement in the North polar region in which the transition from low to high resolution is located at 60N. To create this grid, the best tool to use is the rectangular editor. After starting the *VRM_Editor*, select the Edit tab and click on the 'Edit refine Map' button. Select the 'Rectangular Editor' option and set the Longitude Min/Max values to span the entire range [-1.,361.]. Now set the Latitude range to [60.,135.] and set the taper value for Latitude to 5 degrees. Click on the 'Apply' button to fill in the refine map values with values of 1 at the pole which taper down to 0 at about 65N. Click on 'Exit Edit Mode' and keep the changes when asked.

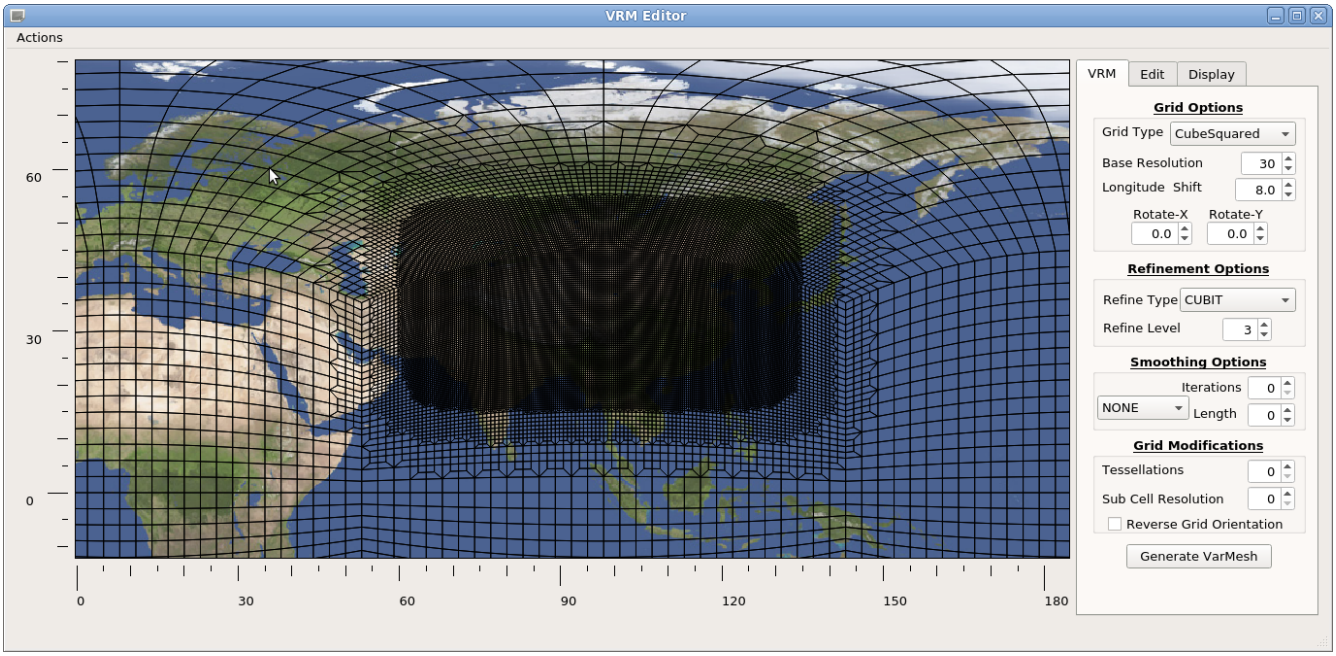
Now change to the VRM tab and set the refinement options to a refinement level of 2 using the CUBIT refinement type. Change the Smoothing options to use 10 iterations of SPRING smoothing with a smoothing length of 3. Click on 'Generate VarMesh to create the resulting grid with an NE120 resolution over the North Pole which tapers down to a NE30 resolution below 60N. From the Actions menu, now save the results in an EXODUS file. The image shows the resulting refined grid from this file.



Combining the polygon editor with the rectangle editor and applied smoothing, a refinement to NE240 can readily be added over Greenland. Without providing the details, the second image above shows such a polar grid.

Refinement over China Example Case

To create the refined grid to be used as an example for the remainder of this document, run the *VRM_Editor* program and enter edit mode. For the rectangular editor, set the Longitude range to [55,140] and the Latitude range to [10,60]. Then, for each, set the taper value to 7 degrees. Once the refinement map is created, exit edit mode and switch to the VRM tab. The default base resolution is 30, change the default Longitude shift to +8 degrees. Then for the CUBIT refine type, set the refine level to 3. Leave the default smoothing set to NONE and invoke the Generate VarMesh button. The resulting grid is an NE30 grid with a NE240 refinement region over China.



In the Actions menu select write EXODUS file and save the resulting grid in the $(REPO) directory. Name the resulting file: China01_ne30x8_EXODUS.nc

```
cheyenne% mkdir $(REPO)/ne0np4.China01.ne30x8
cheyenne% mkdir $(REPO)/ne0np4.China01.ne30x8/grids
cheyenne% mv China01.ne30x8_EXODUS.nc $(REPO)/ne0np4.China01.ne30x8/grids/
```

Now create the SCRIP and LATLON file for this grid using Gen_ControlVolumes with the input_nl file.

```
&input_nl
  GridPath = '/path/to/my/repo/ne0np4.China01.ne30x8/grids/'
  GridName = 'China01_ne30x8'
/
```

At this point the grid files needed for the refinement over China are complete. To use this grid for CESM experiments, the associated data files must first be constructed and then the model must be configured to make use of resulting set of files in **$(REPO)/ne0np4.China01.ne30x8**.

Generate Input Datasets for CESM

Once the grid is created, it is necessary to create the mapping and domain information need for CESM. The initial data and surface forcing values must be interpolated onto the new grid as well. These steps require programs found in the *tools* directories in CESM as well as the scripts and NCL programs found in VRM_tools.

For variable resolution grids, the spectral element dycore requires a different initialization process at start up. Typically, for uniform resolutions the grid names in CESM have the form neZZnp4, where ZZ = {30, 120, 240, etc}. To identify variable resolution grids for alternate processing, they all must begin with the prefix 'ne0np4'. The typical variable resolution grid has the form:

ne0np**4**.**GRIDNAME**.ne**ZZ**x**RR**

For the China refinement example, the model grid name will be:

ne0np4.China01.ne30x8

In the VRM_tools directory, there are a number sub-directories each of which contain scripts or NCL programs that create a subset of the needed files. To create all of the files required to run the model, it is just a matter of going through each of these directories, editing values in the associated script/program to indicate the name/path information for the new grid, and then running it. Once all of the directories are processed, the needed grid files are ready for use.

In each directory there is a MAKE script. Is uses a set of user specified environment variables to build a processing script that generates the needed data values. The environment variables that the user must set are:

```
cheyenne% setenv VRM_CESM_TAG  "/path/to/cesm/tag/"
cheyenne% setenv VRM_REPO_PATH "/path/to/my/repo/"
cheyenne% setenv VRM_GRID_TAG  "MYNEWGRIDNAME_neZZxRR"
cheyenne% setenv VRM_GRID_NAME "ne0np4.MYNEWGRIDNAME.neZZxRR"
cheyenne% setenv VRM_GRID_NCOL "NNNNNNN"
cheyenne% setenv VRM_DATE      "YYMMDD"
```

The VRM_CESM_TAG is the path to the tag in which the CAM/CLM tools have been built, VRM_REPO_PATH is the root path to the users repository, VRM_GRID_TAG is the name assigned to the grid when it is created, VRM_GRID_NAME is the CESM resolution name assigned to the grid, VRM_GRID_NCOL are the number of unique gridpoints for the new grid, and VRM_DATE is the creation date that will be assigned to all created files. The number of unique columns for a grid can be found in the LATLON grid file. The MAKE scripts are set up to use the current CESM default datasets for creating the files needed for the new grid. The defaults may need to be updated from time to time as CESM development continues. When verifying the setting for each processing script, users are encouraged to make a note of the files used and consider it they are out of date for the current CESM tag.

For the China01_ne30x8 example case, the environment settings are:

```
cheyenne% setenv VRM_CESM_TAG  "/path/to/cesm/tag/"
cheyenne% setenv VRM_REPO_PATH "/path/to/my/repo/"
cheyenne% setenv VRM_GRID_TAG  "China01_ne30x8"
cheyenne% setenv VRM_GRID_NAME "ne0np4.China01.ne30x8"
cheyenne% setenv VRM_GRID_NCOL "251534"
cheyenne% setenv VRM_DATE      "190813"
```

Run gen_mapping

In the gen_mapping directory, there are scripts which create the data files needed to map values from/to the new variable resolution grid and the grids used by various CESM components. After running these scripts, a directory

$(REPO)/ne0np4.GRIDNAME.neZZxRR/maps

will have been created. It contains all of the mapping combinations a user should need. Note that the source grid files for ROF, GLC, and WAV may need to be modified over time as the model defaults for these grids are updated. To generate the need scripts:

```
cheyenne% cd $(VRM_tools)/gen_mapping/
cheyenne% ./MAKE_genMapping_scripts.csh
```

This script will create a directory with the name of the new grid which contains three processing scripts to create maps for the new grid. Prior to running each of these, the scripts should be checked to ensure that the grid-specific variables have been set correctly. For the China01_ne30x8 example the values should be:

```
VRrepoPath  = "/path/to/my/repo/"
VRgridName  = "ne0np4.China01.ne30x8"
VRscripName = "China01_ne30x8_np4_SCRIP.nc"
cdate       = "190813"
```

These scripts require NCL and can be run on Capser or Cheyenne. In the NCAR environment, this means that the ncl module must be loaded, and if the scripts are run on Cheyenne, **qcmd** must be used.

```
cheyenne% cd ne0np4.China01.ne30x8
cheyenne% module load ncl
cheyenne% qcmd -- 'sh merged-mapping_China01_ne30x8_01.sh >& LOG_China01_ne30x8_01'
cheyenne% qcmd -- 'sh merged-mapping_China01_ne30x8_02.sh >& LOG_China01_ne30x8_02'
cheyenne% qcmd -- 'sh merged-mapping_China01_ne30x8_03.sh >& LOG_China01_ne30x8_03'
```

Users are encouraged to check the output in the LOG files to ensure that the processing scripts ran correctly without errors.

Run gen_domain

The MAKE script in the gen_domain directory creates two scripts in a VRM_GRID_NAME sub-directory, one for gx1v7 and the other for tx0.1v2. These generate the lnd/ocn domain files for each of these masks in the directory

$(REPO)/ne0np4.GRIDNAME.neZZxRR/domains

To create these scripts:

```
cheyenne% cd $(VRM_tools)/gen_domain/
cheyenne% ./MAKE_genDomains_scripts.csh
```

Check these scripts to see that the grid dependent variables have been set correctly. For the China01_ne30x8 example, the values are:

```
CESMroot    = "/path/to/cesm/tag/"
VRrepoPath  = "/path/to/my/repo/"
VRgridName  = "ne0np4.China01.ne30x8"
VRscripFile = "${VRrepoPath}/${VRgridName}/grids/China01_ne30x8_np4_SCRIP.nc"
cdate       = "190813"
```

Run these scripts to generate the needed lnd/ocn domain files.

```
cheyenne% cd ne0np4.China01.ne30x8
cheyenne% qcmd -- 'sh genUnigridDomains_China01_ne30x8_gx1v7.sh >&
                                                    LOG_China01_ne30x8_gx1v7'
cheyenne% qcmd -- 'sh genUnigridDomains_China01_ne30x8_tx0.1v2.sh >&
                                                     LOG_China01_ne30x8_tx0.1v2'
```

Users are encouraged to check the output in the LOG files to verify that the programs ran without errors.

Run gen_CLMsrfdata

The script in the gen_CLMsrfdata directory generates the CLM fsurdat and transient land use files for 1850/2000 runs. This is the most time consuming step for creating grid datasets. The files generated will be located in the directory

$(REPO)/ne0np4.GRIDNAME.neZZxRR/clm_surfdata_5_0/

Begin by creating the processing script.

```
cheyenne% cd $(VRM_tools)/gen_domain/
cheyenne% ./MAKE_genCLMsurfdata_script.csh
```

This script is a little different from the others in that it is a longer job and must be submitted to Cheyenne via a **qsub** command. So in addition to checking the grid dependent variable settings, users must also set a project number to use. For the China01 example, the values in the script are:

```
#PBS  -A P12345678

CESMROOT    = "/path/to/cesm/tag/"
OUTBASE     = "/path/to/my/repo/"
VRgridName  = "ne0np4.China01.ne30x8"
VRscripFile = "${VRrepoPath}/${VRgridName}/grids/China01_ne30x8_np4_SCRIP.nc"
cdate       = "190813"
```

This job typically takes a couple of hours to run, but for very refined grids it can take much longer to complete.

```
cheyenne% cd ne0np4.China01.ne30x8
cheyenne% qsub < genCLMsurfdata_China01_ne30x8.sh
```

Note that in addition to creating the needed landuse and fsurdat files in the clm_surfdata_5_0 directory of the $(REPO), there is a directory **maps_clm** that is also created. These are the mapping files that were used to create the CLM files. They are somewhat large files, so once it is verified that the program ran correctly, the contents of this directory can be erased.

Run gen_CAMncdata

The script in the gen_CAMncdata interpolates data horizontally and vertically from an initialization file onto a given atmospheric model grid. When the script is complete, the resulting initialization file is stored in the directory

$(REPO)/ne0np4.GRIDNAME.neZZxRR/inic

To create this script:

```
cheyenne% cd $(VRM_tools)/gen_CAMncdata/
cheyenne% ./MAKE_interpic_script.csh
```

As for the other data generation scripts, first verify that the grid-dependent variable have been set correctly. The appropriate values for the China01 example are:

```
CESMROOT     = "/path/to/cesm/tag/"
VRdate       = "190813"
VRgridName   = "ne0np4.China01.ne30x8"
VRrepoPath   = "/path/to/my/repo/"
VRlatlonFile = "${VRgridPath}/grids/China01_ne30x8_np4_LATLON.nc"
```

This script requires NCL and the NCO operators, so load them on Cheyenne first before running

```
cheyenne% cd ne0np4.China01.ne30x8
cheyenne% module load ncl
cheyenne% module load nco
cheyenne% qcmd -- 'sh interpic_script_China01_ne30x8.sh >& LOG_China01_ne30x8'
```

As with the other scripts in VRM_tools, user should check the LOG file fro errors.

Run gen_atmsrf

The script in gen_atmsrf interpolates the default input 'atmsrf' file to the given variable resolution grid. The default is currently 'atmsrf_ne120np4_181018.nc'. If a more appropriate (i.e. higher uniform resolution) file is available, the source input file should be updated. Otherwise, the user just needs to run the MAKE script, verify the grid-dependent variable values in the resulting script and then run it. The resulting file will be created in the directory

$(REPO)/ne0np4.GRIDNAME.neZZxRR/atmsrf

To create the script:

```
cheyenne% cd $(VRM_tools)/gen_atmsrf/
cheyenne% ./MAKE_genAtmsrf_script.csh
```

Verify the values at the top of the script. For the China01 example:

```
cdate        = "190813"
VRgridName   = "ne0np4.China01.ne30x8"
VRgridPath   = "/path/to/my/repo/"+VRgridName
VRscripFile  = VRgridPath+"/grids/+"China01_ne30x8_np4_SCRIP.nc"
VRlatlonFile = VrgridPath+"/grids/+"China01_ne30x8_np4_LATLON.nc"
```

Load the NCL module if on Cheyenne, and the run the program.

```
cheyenne% cd ne0np4.China01.ne30x8
cheyenne% module load ncl
cheyenne% qcmd -- 'ncl gen_atmsrf_China01_ne30x8.ncl >& LOG_China01_ne30x8'
```

Run gen_topo

There are currently two options for creating topography data for variable resolution grids. The first, Regrid_topo, carries out an interpolation of the current default topography for ne120np4 grid. This includes the interpolation of the ridge date needed for orographic gravity wave parameterizations. This program will produce workable results and is best used for quickly installing and testing a new grid. For scientifically meaningful experiments it is preferable that the users use the topo program available from https://github.com/NCAR/topo to generate topography date consistent with the variable resolution grid directly from the high resolution source datasets.

Regrid_topo

The NCL program in Regrid_topo interpolates topography information, including ridge data used in the gravity wave parameterizations, from the current ne120np4 default file. The interpolated results will be created in the directory

$(REPO)/ne0np4.GRIDNAME.neZZxRR/topo

First create the script:

```
cheyenne% cd $(VRM_tools)/gen_topo/Regrid_topo/
cheyenne% ./MAKE_RegridTopo_script.csh
```

Verify that the grid-dependent variable are set correctly, for the China01 example the values are:

```
VRgridName   = "ne0np4.China01.ne30x8"
VRgridPath   = "/path/to/my/repo/"+VRgridName
VRscripFile  = VRgridPath+"/grids/"+"China01_ne30x8_np4_SCRIP.nc"
VRlatlonFile = VRgridPath+"/grids/"+"China01_ne30x8_np4_LATLON.nc"
cdate        = "190813"
```

Then run it:

```
cheyenne% module load ncl
cheyenne% qcmd -- 'ncl Regrid_topo_China01_ne30x8.ncl >& LOG_China01_ne30x8'
```

topo

In the topo directory, follow the instructions provided in the repository to configure and run the programs to generate a true topography file for the VRM grid directly from source data.

Run gen_ESMFmesh

The script in gen_ESMFmesh will create the ESMF mesh file needed by NUPOC from the current SCRIP file for the grid.

To create the script:

```
cheyenne% cd $(VRM_tools)/gen_ESMFmesh/
cheyenne% ./MAKE_ESMFmesh_script.csh
```

Verify that the grid dependent values are set correctly and that the current/correct version of ESMF modules are set.

```
module load intel/17.0.1
module load esmflibs/7.1.0r
module load esmf-7.1.0r-ncdfio-uni-O

VRrepoPath="/path/to/my/repo/"
VRgridName="ne0np4.China01.ne30x8"
VRscripFile="China01_ne30x8_np4_SCRIP.nc"
VRmeshFile="China01_ne30x8_np4_MESH.nc"
```

The log into Casper and run the script to generate the MESH file.

```
cheyenne% cexecdav
casper% cd ne0np4.China01.ne30x8
casper% chmod 744 gen_ESMFmesh_China01_ne30x8.sh
casper% ./gen_ESMFmesh_China01_ne30x8.sh >& LOG_China01_ne30x8'
```

A Note About the CLM finidat File

Currently, values from the specified CLM initialization file are interpolated to the model grid resolution automatically at startup. This has rendered the previously developed program to carry out this interpolation obsolete. However, if a variable resolution grid with a high resolution over land is being used, the user should be mindful that it may be necessary to change the default startup file to the resolution closest to the VRM highest resolution, if such a file is available.

Known Issues

VRM_Editor crashing

The VRM_Editor fails with a segmentation fault from time to time. This usually occurs after a large number of iterations of re-editing a refinement map and generating new grids in an interactive session, in particular when using the polygon editor with smoothing. Failure is also more common when there are a *very* large number of refinements. It is suspected that there is a memory leak somewhere in that SQuadGen was not originally designed for iterative use, but the actual origin of the failures is not currently known. Prior to failure, there have been no adverse effects on any of the grids generated thus far. If a failure occurs, it is recommended that the user just try to generate the desired grid again. It the failure repeats, then add a little smoothing, or perhaps smooth a bit less and the failure will likely not occur.

For this reason, when constructing a more involved refinement map, it is recommended that the user save intermediate copies of the map. Then if a failure does occur, the generation can begin from the most recently saved map rather than starting from scratch.

CLM tools

Problem: The CLM tools are hard-coded to use a particular version of ESMF. So whenever the systems are updated, these scripts no longer work and must be fixed. After the recent upgrade to Cheyenne, the value of ESMFBIN_PATH in genCLMsurfdata_TEMPLATE.sh must be changed to:

ESMFBIN_PATH = "/glade/u/apps/ch/opt/esmf/7.1.0r-ncdfio/intel/17.0.1/bin/bin0/Linux.intel.64.mpi.default"

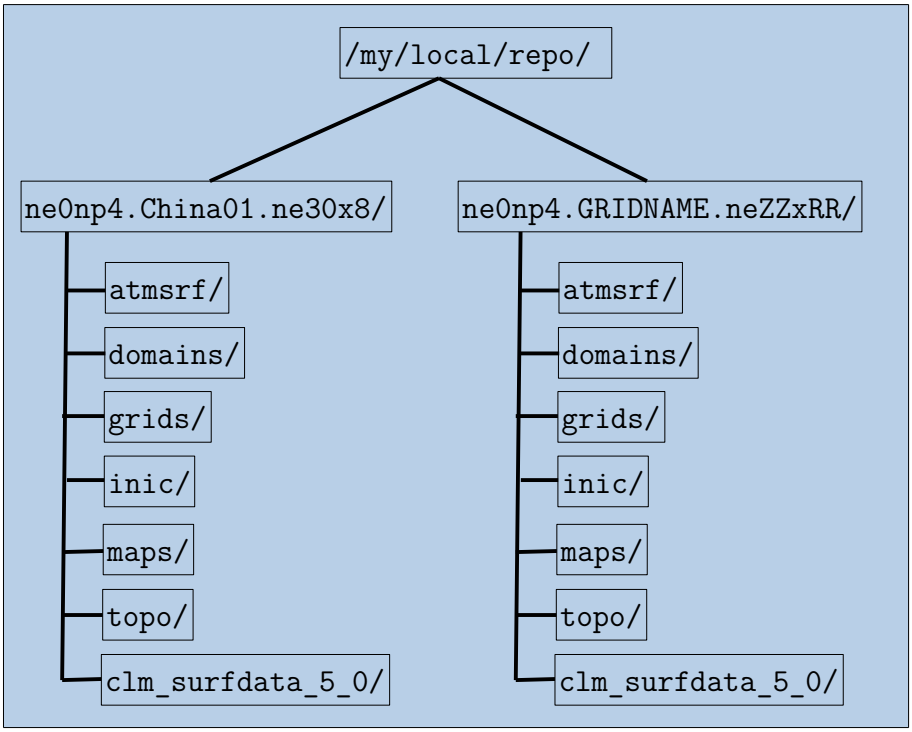
Also, in the directory $(CESM)/components/clm/tools/mkmakdata/ , the mkmapdata.sh script must be fixed to set the specific ESMF version and the ncl module load command does not work unless the explicit version is given:

cheyenne% cd $(CESM)/components/clm/tools/mkmapdata/
cheyenne% cp mkmapdata.sh mkmapdata.sh-ORIG
cheyenne% vi mkmapdata.sh

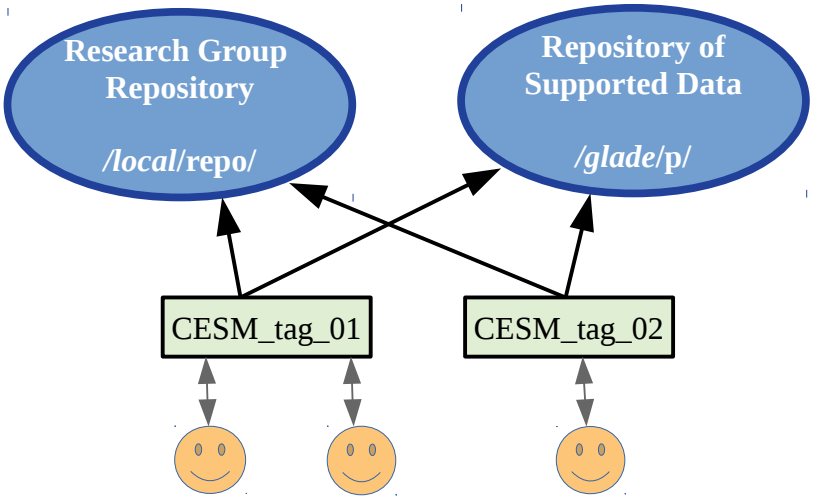
Replace instances of "**esmfvers=7.0.0**" with "**esmfvers=7.1.0r**"
Replace "**module load ncl**" with "**module load ncl/6.6.2**"

Installing a new Grid in CESM

Once the variable resolution grid files have been created and all of the data processing scripts have been run, the user has generated a repository of required values stored with the following directory structure.



The next step is to install these files into CESM so that the newly created gird can be readily used. The usage scenario envisioned for the CESM 2.2 release is for a research group to be able to create and maintain a common repository of grid information, as illustrated in the following diagram. This directory $(REPO) will contain the VRM grids relevant to their research, which supplements the supported grid resolutions in CESM.



The repository data can be readily shared by members of a research group using compatible CESM tags. Compatible tags are those for which there are no significant structural changes in the XML configuration files, namelist entries, or the internal format of input datasets. As CESM development progresses, the $(REPO) will need to be updated periodically to maintain compatibility, but not for every tag. An example that would have required an update to the $(REPO) datasets, was the addition of gravity wave parameterizations that require ridge data in the topography file. A CESM tag containing this revision requires different input data and also possibly revisions of the default namelist values established for $(REPO) grids.

There are currently two methods for installing a new grid into CESM. For older CESM tags, the old method requires that the user edits 7-9 XML files to install a new grid and set defaults namelist values for a given tag. This same installation process must be carried out for each tag used. The new installation method for the next release requires only the execution of an installation script, which adds grid configuration and user namelist files for the grid into the $(REPO) directory. CIME then reads all of the grid-dependent information for the new grid from these files when a new case is created.

Installation for NEW tags:
First, create the installation script:

```
cheyenne% cd $(VRM_tools)/Install_grid/NEWTAGS/
cheyenne% ./MAKE_install_script.csh
cheyenne% ls
CONFIG-TEMPLATES Install_grid_China01_ne30x8.csh MAKE_install_script.csh
```

Edit the install grid to verify that the grid-dependent variables are correctly set and also to add a grid description for the new VRM grid in the variable GRIDDESC.

```
set GRID_TAG  = "China01_ne30x8"
set GRID_NAME = "ne0np4.China01.ne30x8"
set GRIDDESC  = "An ne30 grid with refinement to ne240 over China"
set REPO_PATH = "/path/to/my/repo/"
set DATE      = "190813"
set NCOL      = "251534"
```

Make the script executable and run it:

```
cheyenne% chmod 744 Install_grid_China01_ne30x8.csh
cheyenne% ./Install_grid_China01_ne30x8.csh
```

When complete, the installation script will have added a several files to the $(REPO) directory for this grid. These include *config_grids.xml*, *shell_commands*, and two namelist files *user_nl_cam* and *user_nl_clm*. The config_grid.xml file contains the grid definitions, domain, and mapping information the model needs. The shell_commands script contains xmlchange commands to set the pelayout for the case, a tolerance limit that must be relaxed for VRM grids, and most importantly the value of ATM_NCPL which sets the timestep for the new grid.

```
#
# shell commands to execute xmlchange commands

#-----------------------------------------
# Leave'em alone.
#-----------------------------------------
./xmlchange --append CLM_BLDNML_OPTS="-no-chk_res"
./xmlchange EPS_AAREA=1.e-4

#-----------------------------------------
# Set the default time step here:
#
#    ATM_NCPL=48  <--> 1800 sec timestep
#    ATM_NCPL=96  <-->  900 sec timestep
#    ATM_NCPL=144 <-->  600 sec timestep
#    ATM_NCPL=288 <-->  300 sec timestep
# notice a pattern?
#-----------------------------------------
./xmlchange ATM_NCPL=144

#-----------------------------------------
# Set the default pelayout:
#-----------------------------------------
./xmlchange NTASKS_ATM=1080
./xmlchange NTASKS_CPL=1080
./xmlchange NTASKS_LND=1080
./xmlchange NTASKS_ICE=1080
./xmlchange NTASKS_OCN=1080
./xmlchange NTASKS_ROF=1080
./xmlchange NTASKS_GLC=1080
./xmlchange NTASKS_WAV=1080
```

The user_nl_clm file sets the flanduse_timeseries and fsurdat datasets to use for the model run. The user must select the appropriate fsurdat file (1850 or 2000) for their needs.

```
flanduse_timeseries = '/path/to/my/repo/ne0np4.China01.ne30x8/clm_surfdata_5_0/
                landuse.timeseries_ne0np4.China01_ne30x8_hist_16pfts_Irrig_CMIP6_simyr1850-20015_c190813.nc'

!===================================
! Comment out the fsurdat file you need to use:
!===================================
!  fsurdat='/path/to/my/repo/ne0np4.China01.ne30x8/clm_surfdata_5_0/
                    surfdata_ne-np4.China01.e30x8_hist_16pfts_Irrig_CMIP6_simyr2000_c190813.nc'
fsurdat='/path/to/my/repo/ne0np4.China01.ne30x8/clm_surfdata_5_0/
                  surfdata_ne-np4.China01.e30x8_hist_16pfts_Irrig_CMIP6_simyr1850_c190813.nc'
!===================================
```

The user_nl_cam file contains the entire *&dyn_se_inparm* namelist with the current default values for the ne120np4 resolution grid, the *&cam_initfile_nl* entries for initial values and topography, and the drydep_srf_file in *&chem_inparm*.

```
! Users should add all user specific namelist changes below in the form of
! namelist_var = new_namelist_value

&dyn_se_inparm
 se_mesh_file              = '/path/to/my/repo//ne0np4.China01.ne30x8//grids/China01_ne30x8_EXODUS.nc'
 se_hypervis_scaling       =  3.0D0
 se_hypervis_subcycle      =  3
 se_nsplit                 =  6
 se_rsplit                 =  4
 se_qsplit                 =  1
 se_nu_top                 =  1.0e5
 se_hypervis_power         =  0
 se_hypervis_subcycle_q    =  1
 se_hypervis_subcycle_sponge =  1
 se_refined_mesh           = .true.
 se_ne                     =  0
 se_nu                     = -1
 se_nu_div                 = -1
 se_nu_p                   = -1
 se_ftype                  =  2
 se_fv_nphys               =  0
 se_fvm_supercycling       = -1
 se_fvm_supercycling_jet   = -1
 se_horz_num_threads       =  0
 se_hypervis_dynamic_ref_state =  .false.
 se_kmax_jet               = -1
 se_kmin_jet               = -1
 se_large_courant_incr     =  .true.
 se_lcp_moist              =  .true.
 se_limiter_option         =  8
 se_phys_dyn_cp            =  1
 se_qsize_condensate_loading  =  5
 se_rayk0                  =  2
 se_raykrange              =  0.5
 se_raytau0                =  0.0
 se_statediag_numtrac      =  3
 se_statefreq              =  0
 se_tracer_num_threads     =  0
 se_tstep_type             =  4
 se_variable_nsplit        = .false.
 se_vert_num_threads       =  0
 se_vert_remap_q_alg       =  1
/

&chem_inparm
drydep_srf_file = '/path/to/my/repo//ne0np4.China01.ne30x8//atmsrf/atmsrf_ne0np4.China01.ne30x8_190813.nc'
/
&cam_initfiles_nl
 bnd_topo      = '/path/to/my/repo//ne0np4.China01.ne30x8//topo/topo_ne0np4.China01.ne30x8_blin_190813.nc'
 ncdata        = '/path/to/my/repo//ne0np4.China01.ne30x8//inic/
                                                cami-mam4_0000-01-01_ne0np4.China01.ne30x8_L32_c190813.nc'
/
```

Some *&dyn_se_inparm* values are modified from the ne120np4 defaults, they set the model to user tensor hyper-viscosity, adjust the strength of the damping, set the model to use a variable resolution grid, and provide the Exodus file for the grid.

The contents of these installation files set the definitions and default values for the model grid. As the model is tuned for optimal behavior, the namelist parameters in these files should be updated to set the optimal values as defaults.

To create a new case for a compatible tag using the new variable resolution grid, the user just needs to issue a newcase command with an additional –user-mods-dir argument that points to the repository.

```
cheyenne% ./create_newcase --case ./path/to/case_name --res ne0np4.China01.ne30x8_mt12
                           --compset FHIST --machine cheyenne  --run-unsupported
                           --user-mods_dir /path/to/my/repo/ne0np4.China01.ne30x8
cheyenne%
```

The datasets were processes for both gx1v7 and tx0.1v2, so both of those masks are supported resolution options: ne0np4.China01.ne30x8_mt12 or ne0np4.China01.ne30x8_mg17.

Installation for OLD tags:

For older tags, the user must edit 7-9 XML files to set the grid information, input datasets, and namelist default values for a new grid. This method is tried and true, but the process varies as the structure and entries in the XML files varies continuously with CESM development. In the OLDTAG subdirectory, there is VRM_INSTALL_NOTES.txt file which goes through the files that need to be edited. These notes are valid for current tags, but it the XML files have differences, the best rule of thumb is to scan the XML files in <cime>, <cam/bld>, and <clm/bld> for an established grid like ne120np4. The entries for that grid provide a guide for adding values for a newly created grid. As with everything else, there are helper scripts.

```
cheyenne% cd $(VRM_tools)/Install_grid/OLDTAGS/
cheyenne% ./MAKE_VRM_Config_Entries_script.csh
cheyenne% ls
MAKE_VRM_Config_Entries_script.csh ne0np4.China01.ne30x8 TEMPLATES VRM_INSTALL_NOTES.txt
```

Run the script in the newly created directory and it will create five “VRMODS_*” files which contain the XML entries that will be needed. Follow the instructions in the installation notes and ‘mouse’ these entries into the XML files.

```
cheyenne% cd ne0np4.Cina01.ne30x8
cheyenne% ./MAKE_VRM_Config_Entries_China01_ne30x8.csh
cheyenne% ls
MAKE_VRM_Config_Entries_China01_ne30x8.csh VRMODS_config_grids.xml
VRMODS_config_component_cesm.xml VRMODS_horiz_grid.xml
VRMODS_namelist_defaults_cam.xml VRMODS_namelist_defaults_clm4_5.xml
```

This installation must be carried out for each tag and once completed, creating a new case is done in the usual manner (without the additional –user-mods-dir argument).

```
cheyenne% ./create_newcase --case ./path/to/case_name --res ne0np4.China01.ne30x8_mt12
                           --compset FHIST --machine cheyenne  --run-unsupported
cheyenne%
```

The user must add entries that need to be modified in the user_nl_cam files on their own. Once the optimal defaults for tuning parameters are sorted out, the corresponding values in the xml files need to be updated.

Adjust CESM Tuning Parameters

For a case with an ne120np4 refinement region, the settings should be nearly the same as for a uniform ne120np4 grid. Most grid require some iteration to get the settings right.

```
! Users should add all user specific namelist changes below in the form of
! namelist_var = new_namelist_value

&dyn_se_inparm
 se_mesh_file             = '/path/to/my/repo//ne0np4.China01.ne30x8//grids/China01_ne30x8_EXODUS.nc'
 se_hypervis_scaling      = 3.0D0
 se_hypervis_subcycle     = 3
 se_nsplit                = 6
 se_rsplit                = 4
 se_qsplit                = 1
 se_nu_top                = 1.0e5
 se_hypervis_power        = 0
 se_hypervis_subcycle_q   = 1
 se_hypervis_subcycle_sponge = 1
 se_refined_mesh          = .true.
 se_ne                    = 0
 se_nu                    = -1
 se_nu_div                = -1
 se_nu_p                  = -1
 se_ftype                 = 2
 se_fv_nphys              = 0
 se_fvm_supercycling      = -1
 se_fvm_supercycling_jet  = -1
 se_horz_num_threads      = 0
 se_hypervis_dynamic_ref_state = .false.
 se_kmax_jet              = -1
 se_kmin_jet              = -1
 se_large_courant_incr    = .true.
 se_lcp_moist             = .true.
 se_limiter_option        = 8
 se_phys_dyn_cp           = 1
 se_qsize_condensate_loading = 5
 se_rayk0                 = 2
 se_raykrange             = 0.5
 se_raytau0               = 0.0
 se_statediag_numtrac     = 3
 se_statefreq             = 0
 se_tracer_num_threads    = 0
 se_tstep_type            = 4
 se_variable_nsplit       = .false.
 se_vert_num_threads      = 0
 se_vert_remap_q_alg      = 1
/
```

VRM_Diagnostics

This directory contains dioagnostic tests and methods for validating a new variable resolution grid. Currently, there are only a few files in this directory. Users who develop tests that they find useful for their new grids are encouraged to add their methods here. Contributions here are in general unsupported and are intended to provide information/guidance for users who are just getting started with variable resolution grids.

Plotting Scripts

Contains NCL and python scripts for analysis and display of data on a variable resolution grid.

VisIt Plugin

This directory contains a basic plugin to for the **VisIt** visualization and analysis tool developed by Lawrence Livermore National Laboratory. It parses the 3D data values according to model level and tags them with a level index. If the input netCDF file contains PHIS topography values, the user has the option of plotting values on model levels or on 3D height surfaces.

SEview

A simple dataset browser like **ncview**. [Not Done Yet].

Aqual-Planet (DCMIP??) Tests

It would be nice to have a standard set of simple tests to evaluate new model grids.