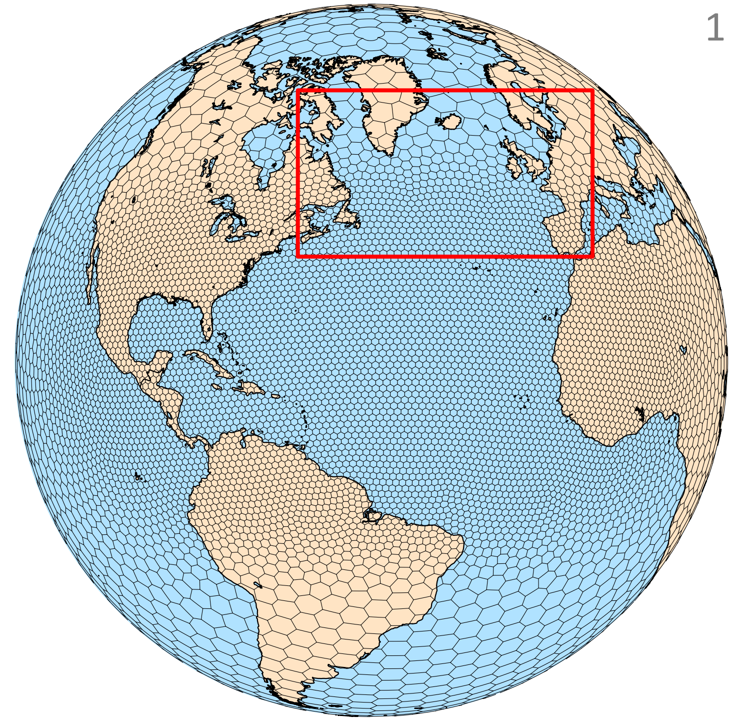
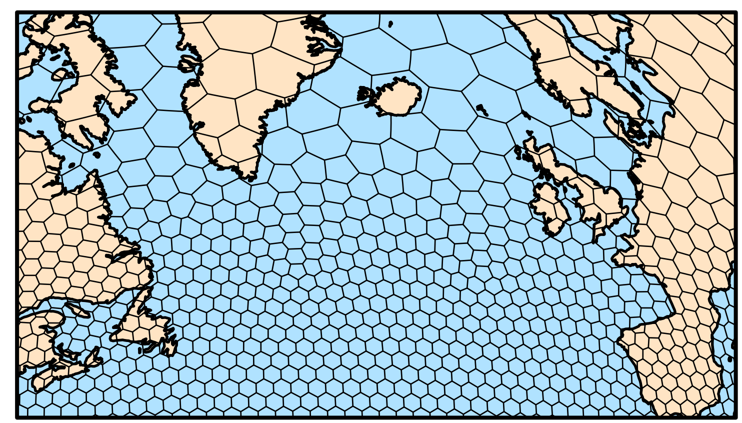
MPAS
Model for Prediction Across Scales

MPAS for WRF Users
or
A Gentle Introduction to Atmospheric Modeling with MPAS





NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Tutorial Outline

Introduction

The MPAS mesh representation

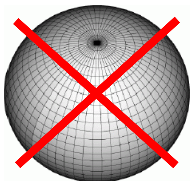
Obtaining, compiling, and running MPAS

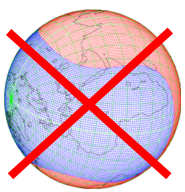
Physics and dynamics configuration

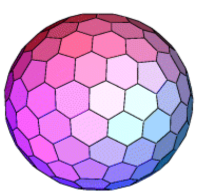
Post-processing and visualization

MPAS
Model for Prediction Across Scales

MPAS Development







2005	Global lat-lon (WRF) problematic.
2006	Triangles - problems with divergence.
2007	Yin-Yang: local conservation past 1st-order accuracy? Cubed-sphere: Corner point problems? Hex grid: C-grid problem solved for perfect hex mesh.
2008	C-grid problem solved for general Voronoi mesh.
2009	Unstructured-mesh MPAS SW eqns. solver. MPAS hydrostatic eqns. solver.
2010	MPAS nonhydrostatic eqns. solver. Hydrostatic MPAS in CAM/CESM.
2011	WRF-NRCM physics in MPAS.
2012	DART data assimilation.
2013	3km global mesh tests on Yellowstone. MPAS V1.0 release (atmosphere, ocean) MPAS-Atmosphere real-time TC forecast testing.
2014	Scale-aware physics testing begins.
2015	15-3 km convective forecast HWT experiments begin Release of *mesoscale reference* physics suite
2017	MPAS V5.0 release, Scale-aware convection permitting physics suite released.
2019	MPAS V7.0 release, Regional MPAS-Atmosphere capability

MPAS
Model for Prediction Across Scales

What is MPAS?

MPAS Version 7.0:

MPAS infrastructure - NCAR, LANL, others.

- Infrastructure for the Voronoi mesh and solvers (data structures; mesh generation, manipulation; operators on the mesh).

MPAS - Atmosphere (NCAR)

- Nonhydrostatic atmospheric solver; pre- and post-processors

MPAS - Ocean (LANL)

- Hydrostatic ocean solver, pre- and post-processors

MPAS - Ice, etc. (LANL and others)

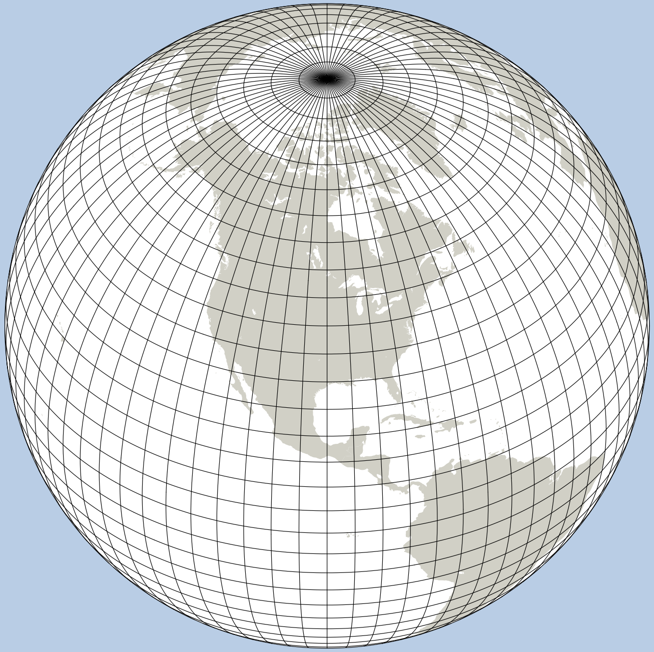
- Land-ice model, pre- and post-processors

These are all stand-alone models – there is no coupler in MPAS

MPAS
Model for Prediction Across Scales

Why MPAS?

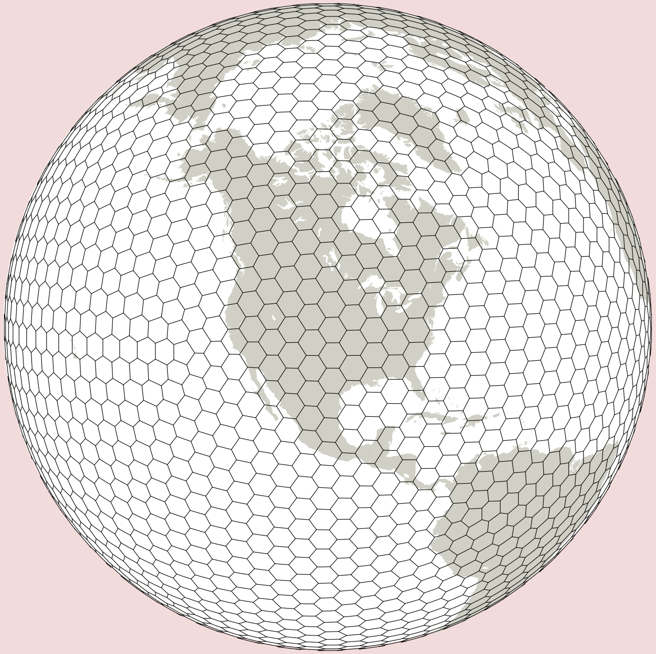
Significant differences between WRF and MPAS



WRF

Lat-Lon global grid

- Anisotropic grid cells
- Polar filtering required
- Poor scaling on massively parallel computers



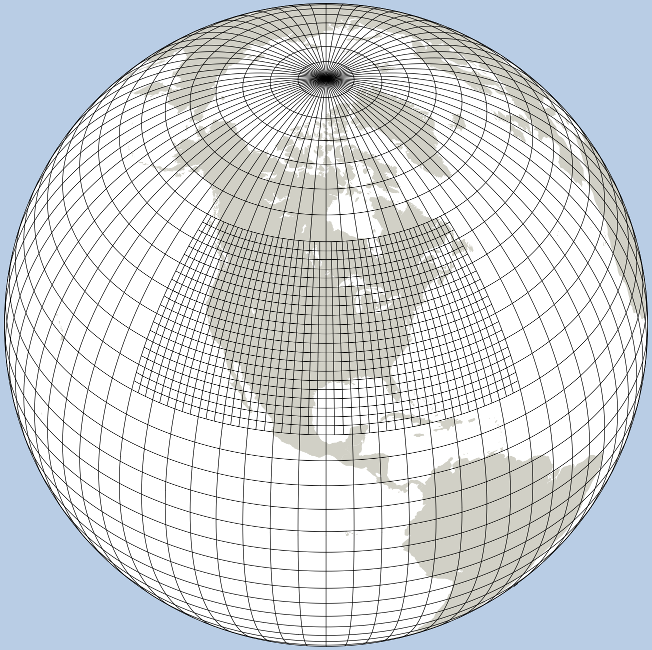
MPAS

Unstructured Voronoi (hexagonal) grid

- Good scaling on massively parallel computers
- No pole problems

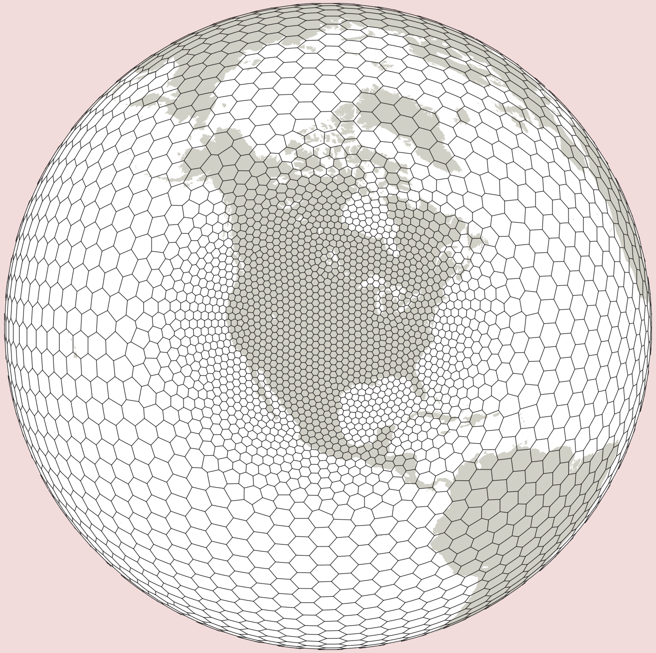
MPAS
Model for Prediction Across Scales

Why MPAS?
Significant differences between WRF and MPAS



WRF
Grid refinement through domain nesting

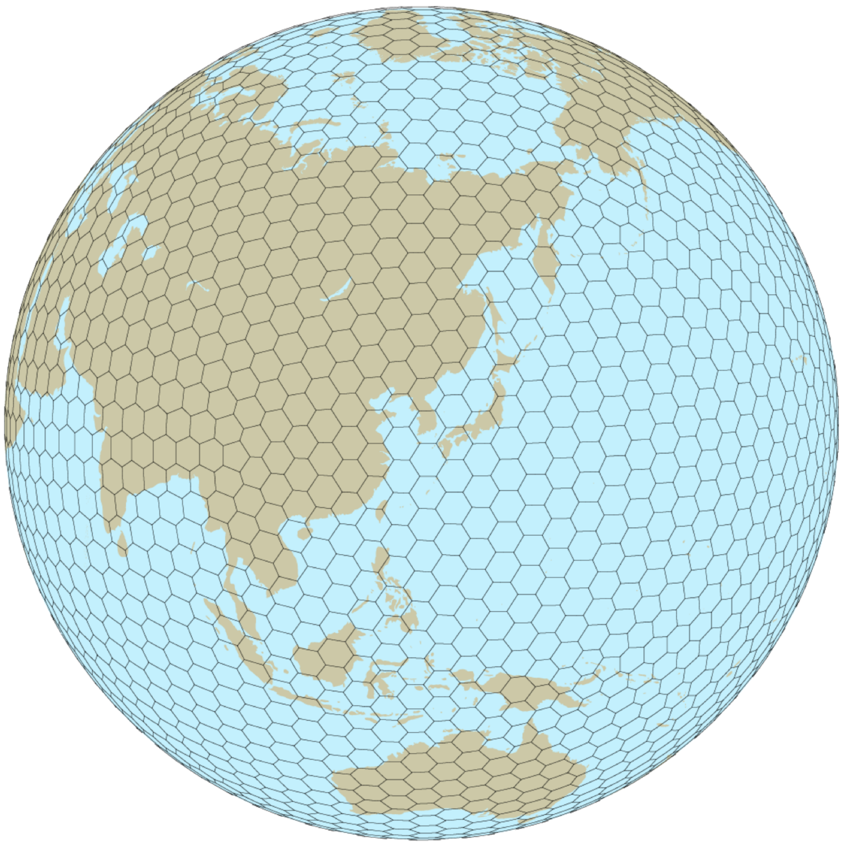
- Flow distortions at nest boundaries



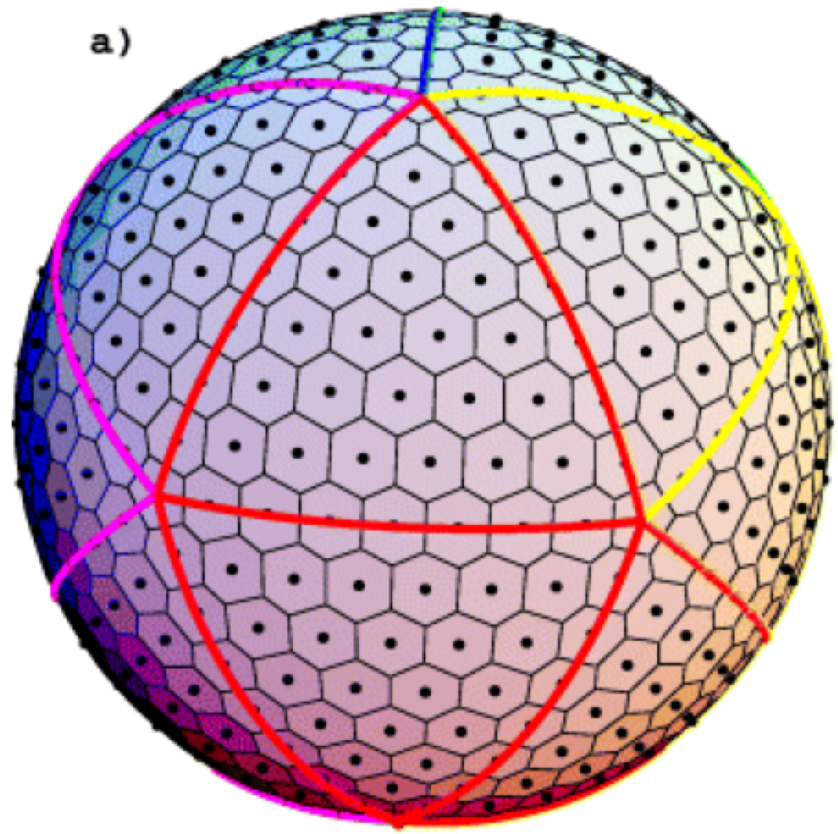
MPAS
Smooth grid refinement on a conformal mesh

- Increased accuracy and flexibility for variable resolution applications
- No abrupt mesh transitions.

Global Meshes



Global Quasi-Uniform Mesh
(SCVT)



Many models use an icosahedral mesh
(NICAM, BUGS, FIM, NIM, OLAM, etc.)

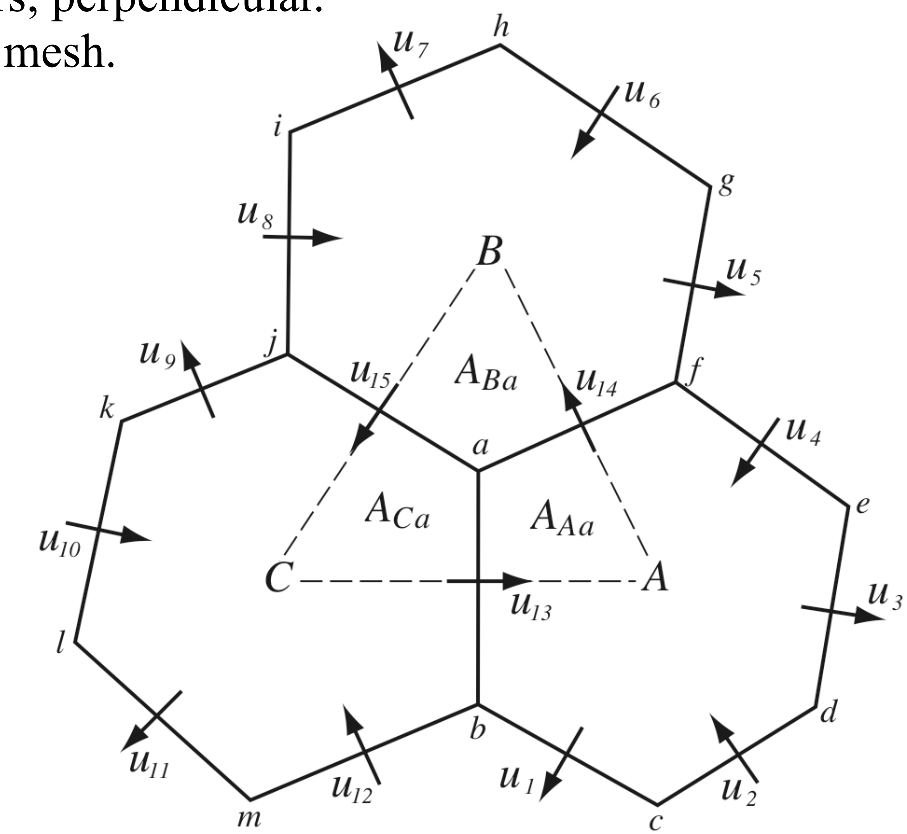
Centroidal Voronoi Meshes

<u>Unstructured spherical centroidal Voronoi meshes</u>

- Mostly *hexagons*, some pentagons and 7-sided cells
- Cell centers are at cell center-of-mass (centroidal).
- Cell edges bisect lines connecting cell centers; perpendicular.
- Uniform resolution – traditional icosahedral mesh.

<u>C-grid</u>

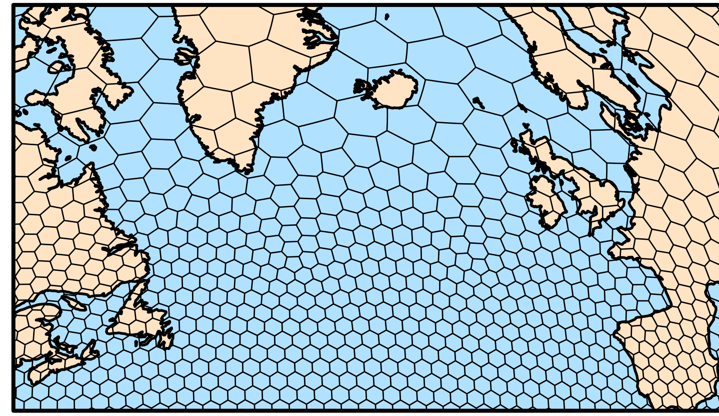
- Solve for normal velocities on cell edges.
- Gradient operators in the horizontal momentum equations are 2nd-order accurate.
- Velocity divergence is 2nd-order accurate for edge-centered velocities.
- Reconstruction of full velocity requires care.



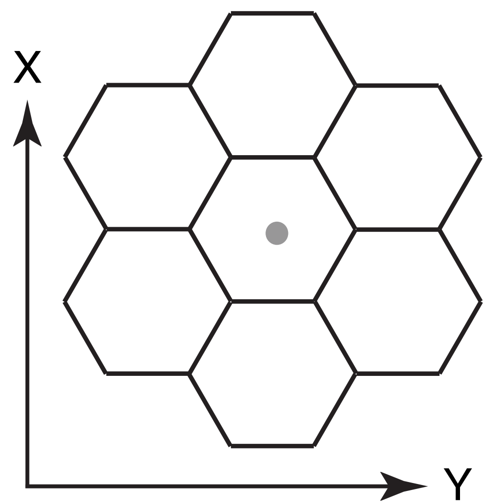
MPAS
Model for Prediction Across Scales

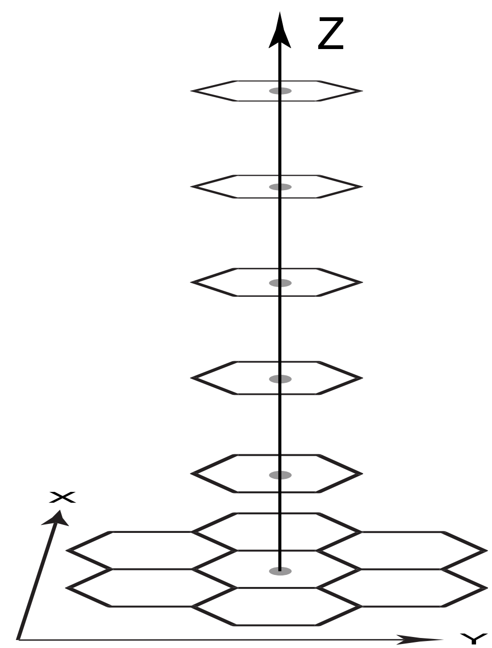
Centroidal Voronoi Meshes

The 2D (horizontal) mesh is *unstructured* – there is no global coordinate



The mesh is *structured* in the vertical

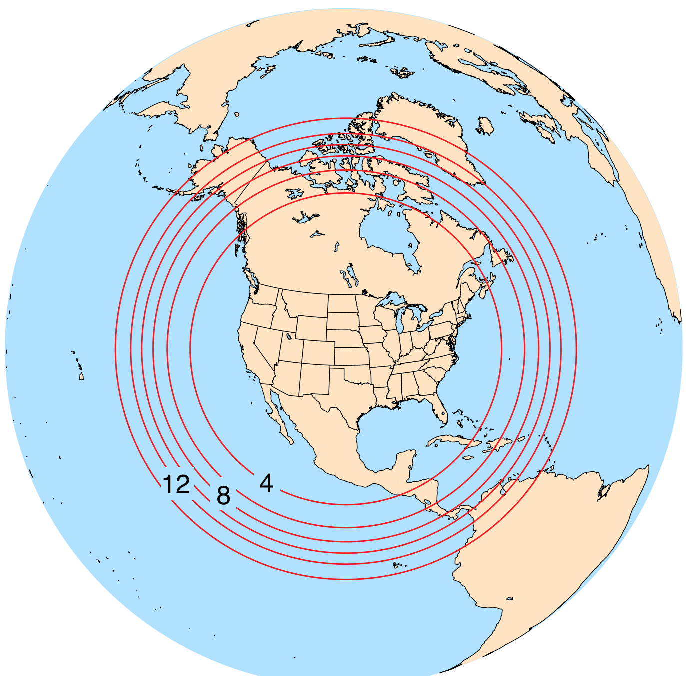




MPAS
Model for Prediction Across Scales

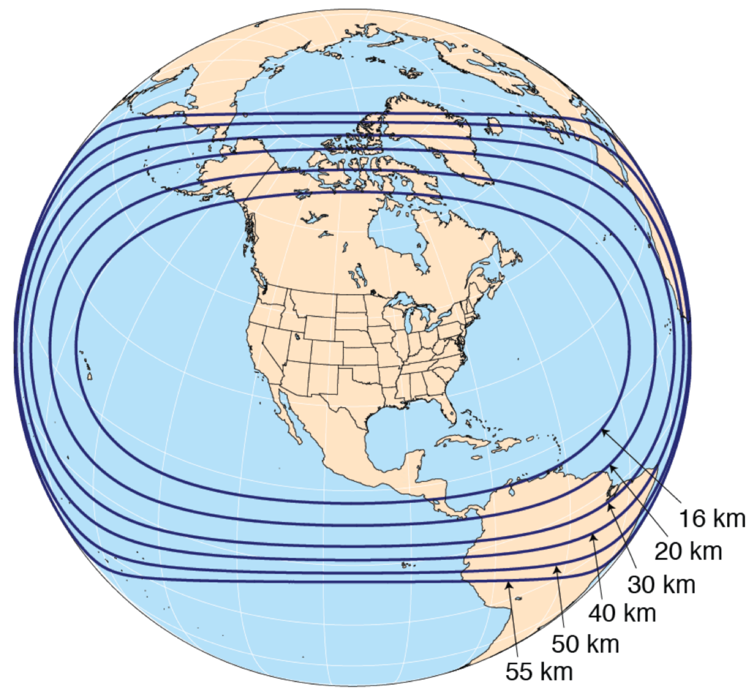
Some Forecast Experiment Meshes

NOAA SPC/NSSL HWT
May 2015, 2016 and 2017
Convective Forecast Experiment
Daily 5-day MPAS forecasts
00 UTC GFS analysis initialization



3-15 km mesh, δx contours 4, 6, 8, 10, 12, 14 km
approximately 6.49 million cells (columns)
50% have < 4 km spacing
(194 pentagons, 182 septagons)

NCAR real-time forecasts
November 2016-June 2017
Daily 10-day MPAS forecasts
00 UTC GFS analysis initialization



60-15 km mesh, δx contours
535,554 cells (columns)
35% have < 16 km spacing
(218 pentagons, 206 septagons)

Tutorial Outline

Introduction

The MPAS mesh representation

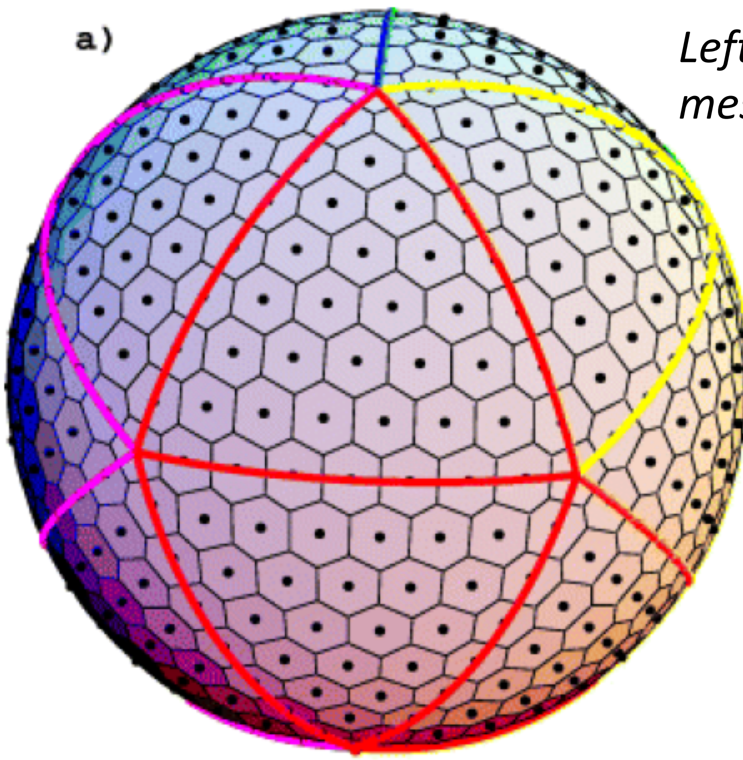
Obtaining, compiling, and running MPAS

Physics and dynamics configuration

Post-processing and visualization

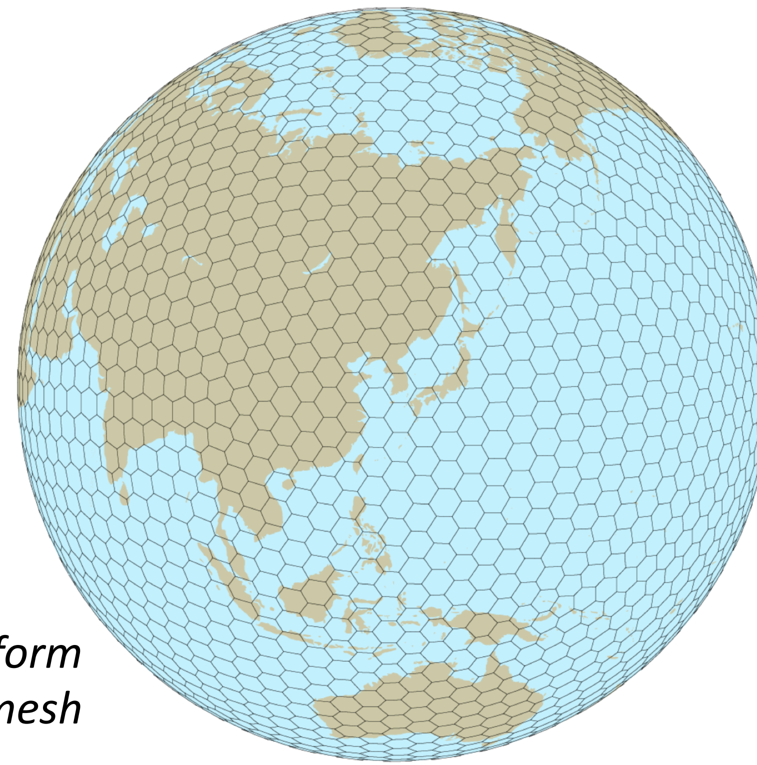
How do MPAS *and you* keep track of this unstructured Voronoi mesh?

Quasi-uniform MPAS meshes look just like icosahedral meshes...



Left: An icosahedral mesh

Right: A quasi-uniform MPAS mesh

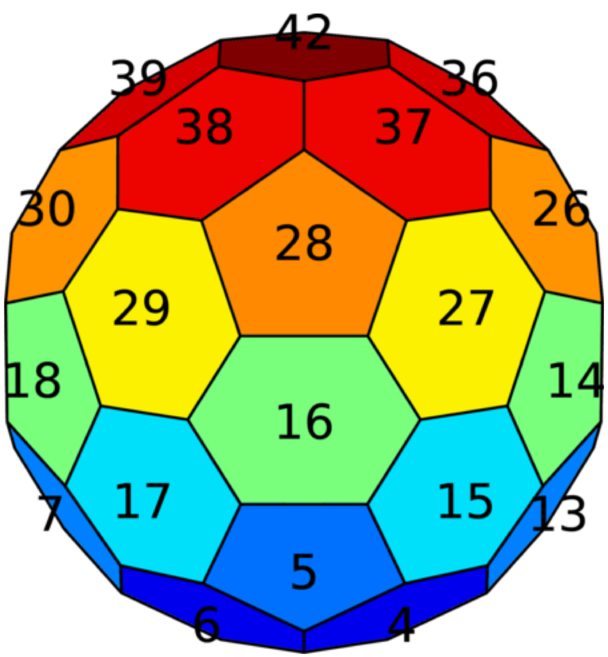


... but the MPAS solver considers every mesh as a completely general, unstructured mesh: there are no special cases!

MPAS
Model for Prediction Across Scales

How do MPAS *and you* keep track of this unstructured Voronoi mesh?

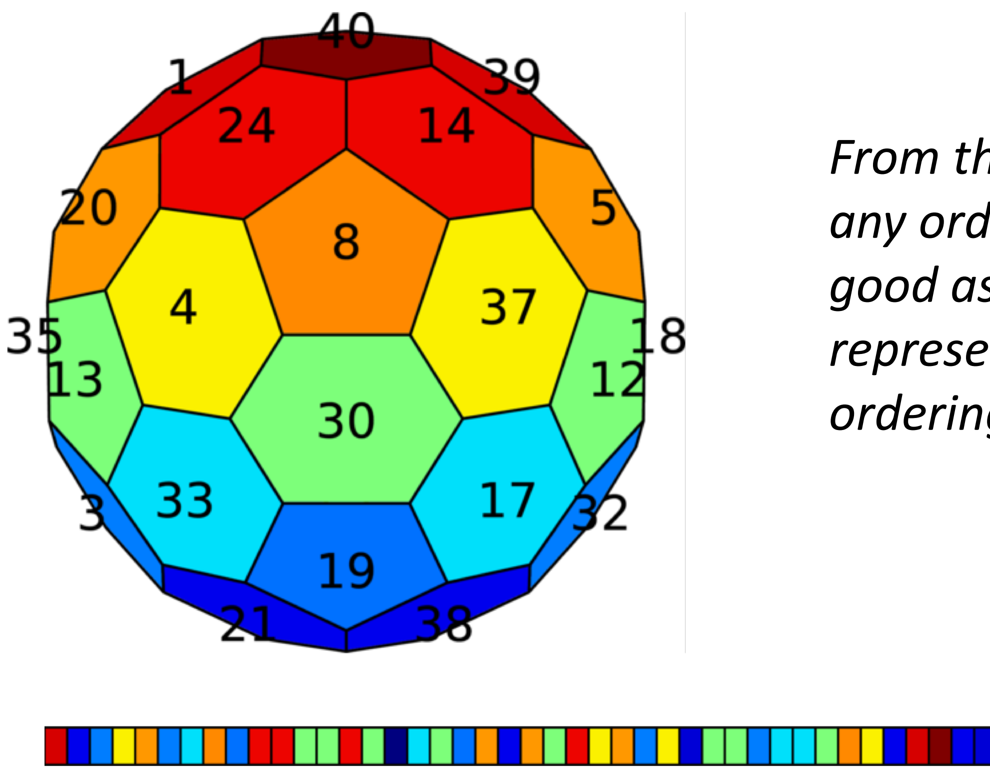
For the unstructured, horizontal dimension there is nothing to be gained from using 2-d arrays...
...hence, the horizontal dimension is collapsed into a single array dimension: we then have a simple list of elements



Example: For some 2-d field (shown in color) defined on mesh cells, that field is stored in a 1-d array (bottom) that is indexed by cell number (labeled in black).

How do MPAS *and you* keep track of this unstructured Voronoi mesh?

For the unstructured, horizontal dimension there is nothing to be gained from using 2-d arrays...
...hence, the horizontal dimension is collapsed into a single array dimension: we then have a simple list of elements

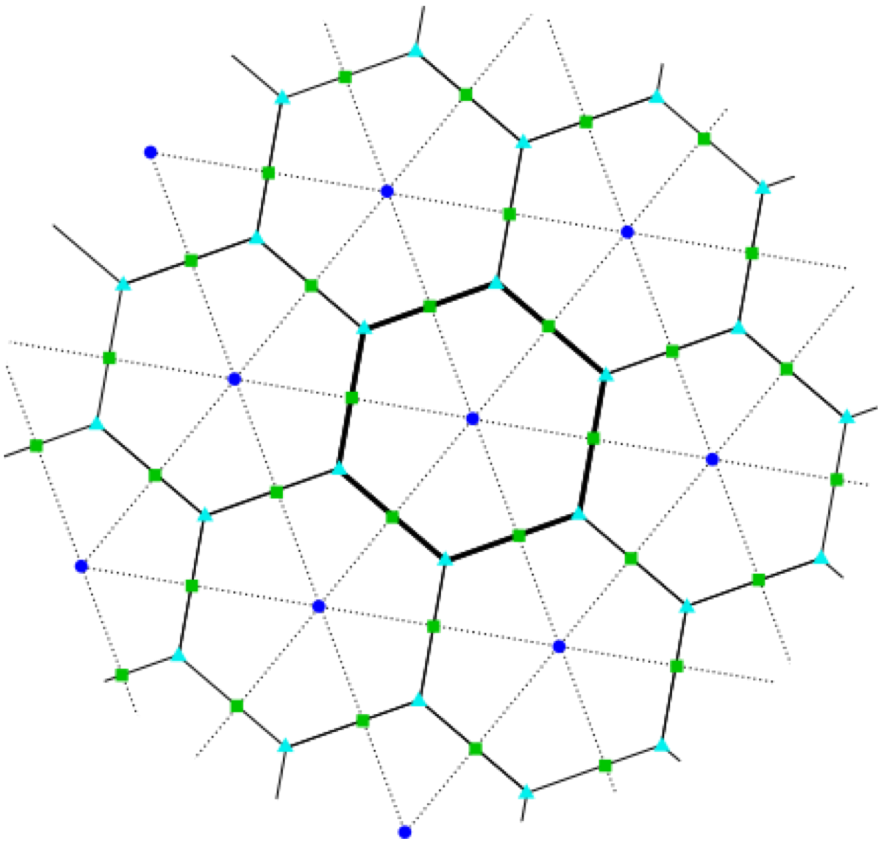


From the perspective of the MPAS solver, any ordering of cells in the mesh is as good as any other[1], as long as the mesh representation is consistent with this ordering.

[1]*Though some orderings may give better performance, e.g., due to better cache reuse.*

How do MPAS *and you* keep track of this unstructured Voronoi mesh?

Schemes for implicitly finding the indices/identities (the "IDs") of neighboring mesh elements (i.e., cells, edges, vertices) are bound to fail...
... so we must find them explicitly through connectivity fields that are the foundation of the MPAS mesh representation.



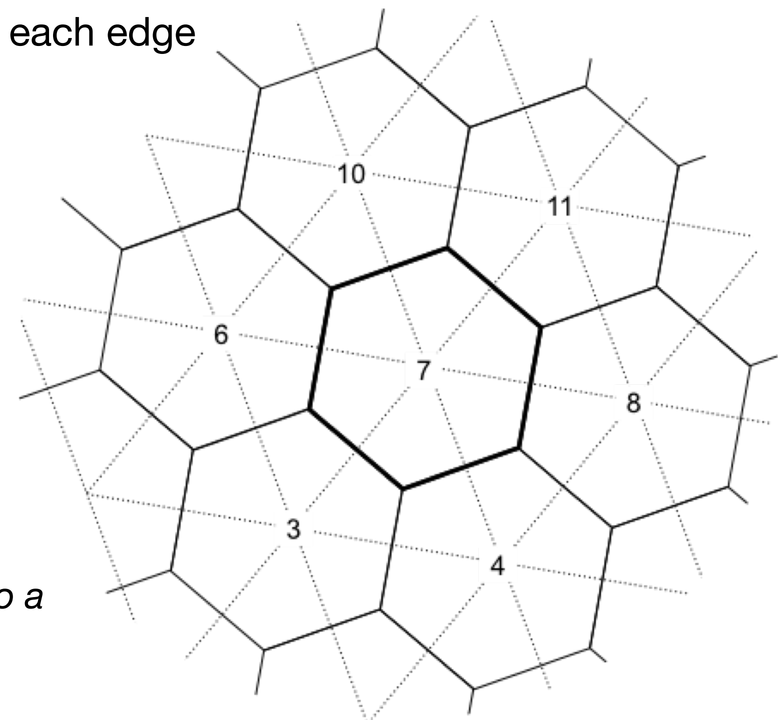
Three types of mesh elements are tracked in the mesh representation:

- **Cell** locations (blue circles) - the generating points of the Voronoi mesh
- **Vertex** locations (cyan triangles) - the corners of primal mesh cells
- **Edge** locations (green squares) - the points where the dual mesh edges intersect the primal mesh edges

Explicit connectivity fields describe the structure of a mesh

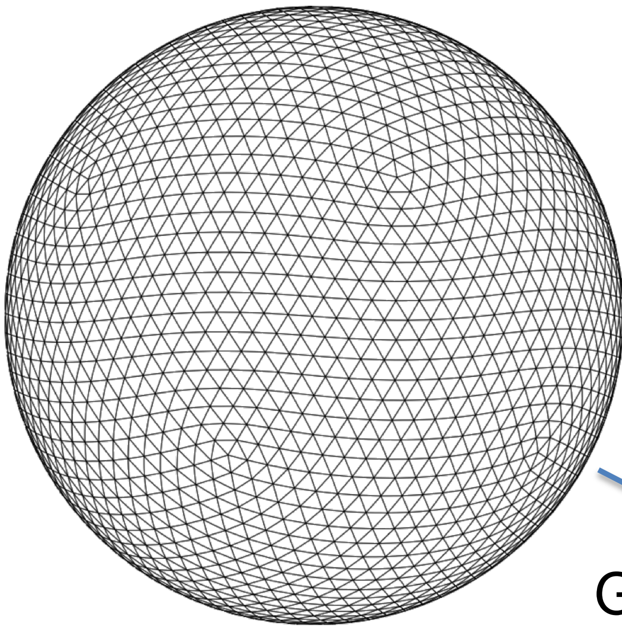
- **nEdgesOnCell(nCells)** – the number of neighbors for each cell
- **cellsOnCell(maxEdges, nCells)** – the indices of neighboring cells for each cell
- **edgesOnCell(maxEdges, nCells)** – the indices of bounding edges for each cell
- **verticesOnCell(maxEdges, nCells)** – the indices of corner vertices for each cell
- **edgesOnVertex(vertexDegree,nVertices)** – the indices of edges incident with each vertex
- **verticesOnEdge(2,nEdges)** – the indices of endpoint vertices for each edge
- **cellsOnVertex(vertexDegree,nVertices)** – the indices of cells meeting at each vertex
- **cellsOnEdge(2,nEdges)** – the indices of cells separated by each edge

```
nEdgesOnCell(7)=6   cellsOnCell(1,7)=8
                    cellsOnCell(2,7)=11
                    cellsOnCell(3,7)=10
                    cellsOnCell(4,7)=6
                    cellsOnCell(5,7)=3
                    cellsOnCell(6,7)=4
```



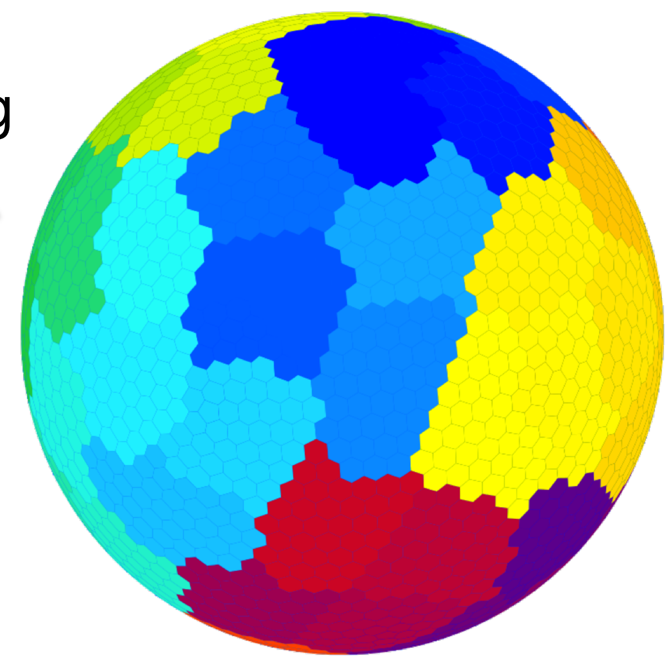
At model start-up, all indices in these arrays are re-numbered to a local indexing scheme.

How are meshes partitioned into blocks for parallelization?



- The *dual* mesh of a Voronoi tessellation is a Delaunay triangulation – essentially the connectivity graph of the cells
- Parallel decomposition of an MPAS mesh then becomes a graph partitioning problem*:* ***equally distribute nodes among partitions (give each process equal work) while minimizing the edge cut (minimizing parallel communication)***

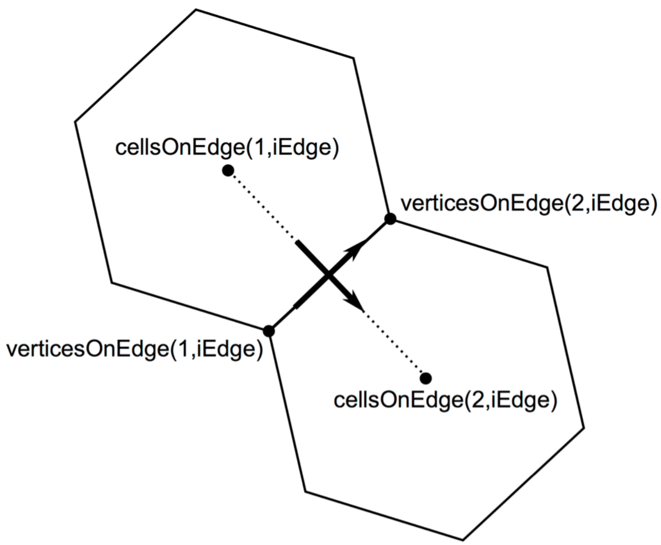
Graph partitioning



We use the Metis package for parallel graph decomposition

- Currently done as a pre-processing step, but could be done "on-line"
- Fortunately, Metis runs quickly, and a partitioning into *n* pieces only needs to be done once for a given mesh

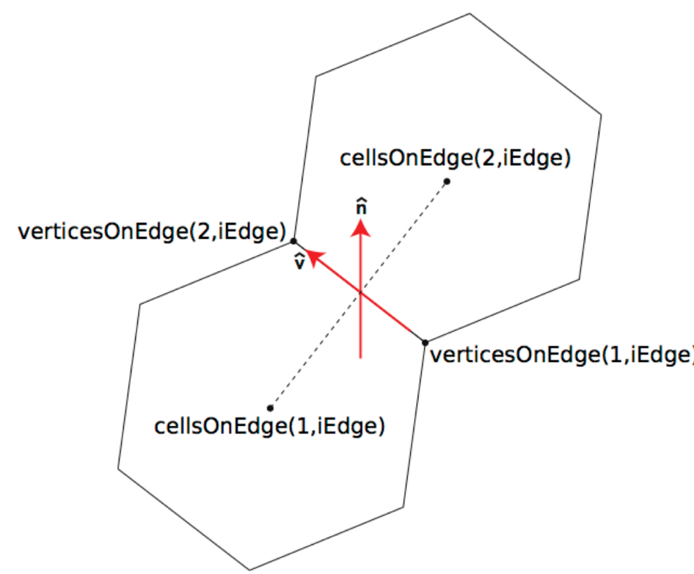
Horizontal wind vectors: how are these defined?



Positive u (normal) velocity is always defined as flow from cellsOnEdge(1,iEdge) to cellsOnEdge(2,iEdge) for edge iEdge

Positive v (tangential) velocity is always defined as flow from verticesOnEdge(1,iEdge) to verticesOnEdge(2,iEdge) for edge iEdge

The cross product of the positive *u* and *v* vectors always points upward (out of the plane)



Earth-relative horizontal winds, u_{zonal} and $u_{meridional}$, can be calculated using u and v:

$$\begin{bmatrix} u_\lambda \\ u_\phi \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

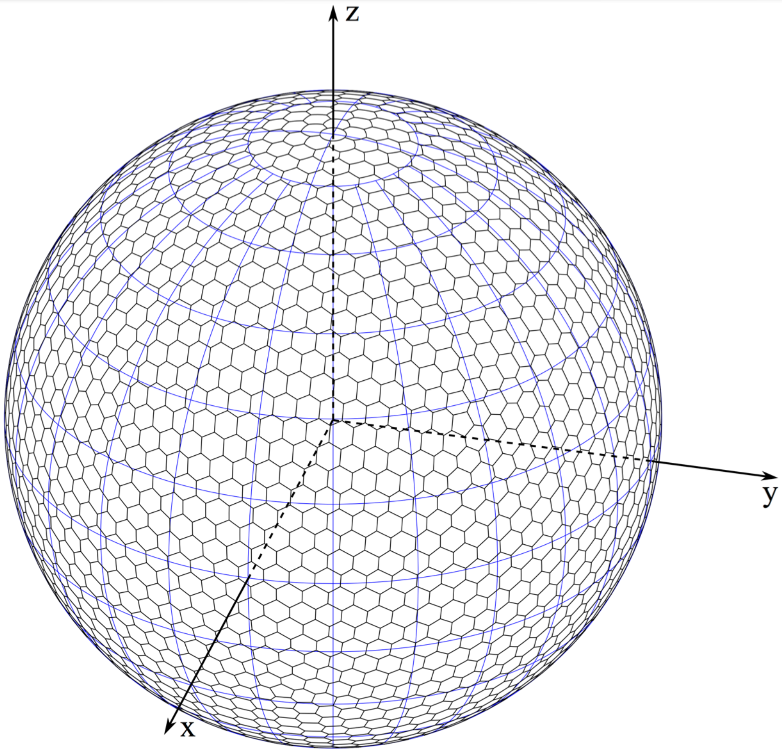
where α is angleEdge.

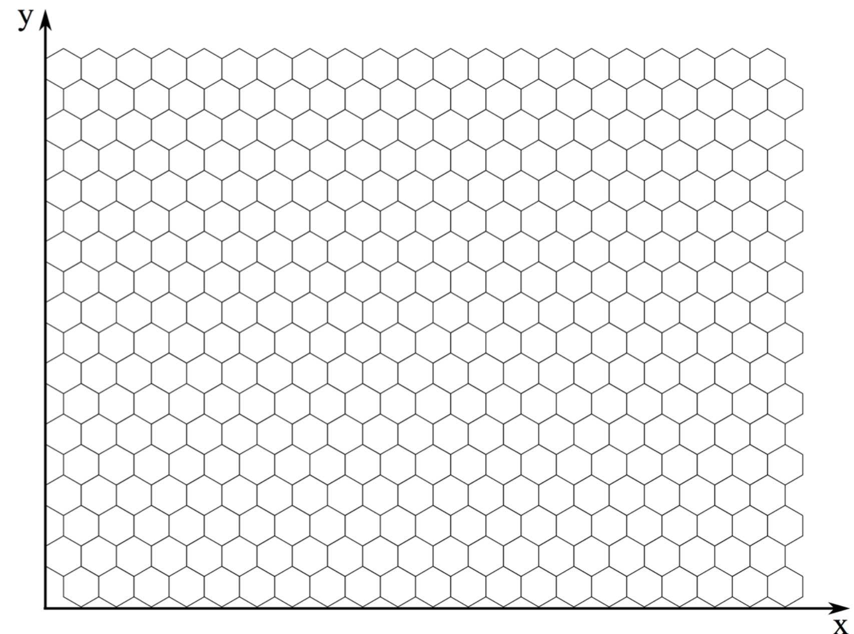
angleEdge(nEdges) – angle between east and positive u

$$\text{angleEdge} = \arcsin \|\hat{\mathbf{n}} \times \hat{\mathbf{v}}\|$$

MPAS
Model for Prediction Across Scales

Which mesh geometries does MPAS support?





On the surface of the sphere: all distances and areas are computed in spherical geometry.

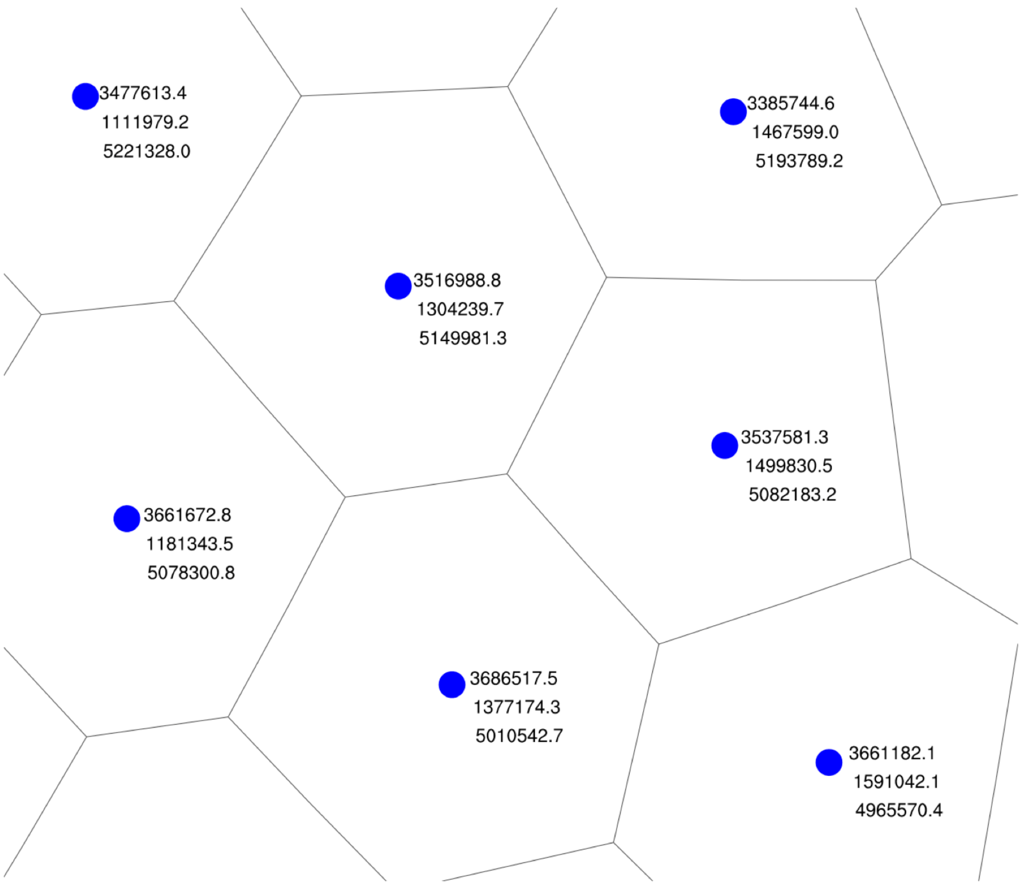
In the Cartesian plane: all distances and areas are computed in Euclidean geometry.

$$x = r\cos(\lambda)\cos(\phi)$$
$$y = r\sin(\lambda)\cos(\phi)$$
$$z = r\sin(\phi)$$

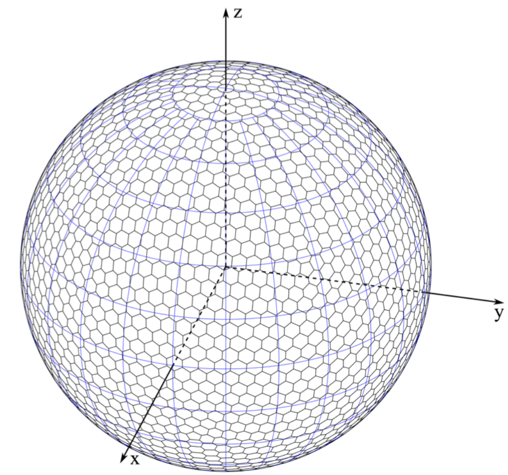
$$\phi = \arcsin\left(\frac{z}{r}\right)$$
$$\lambda = \arctan\left(\left|\frac{y}{x}\right|\right)$$

In the plane, only doubly periodic boundaries are currently supported.

Notes about MPAS mesh geometry



Above: Cartesian coordinates for cell locations near (52.9°N lat, 20.8°E lon) in a variable-resolution spherical mesh with radius 6371229 m.



Global Cartesian coordinates are computed for each element

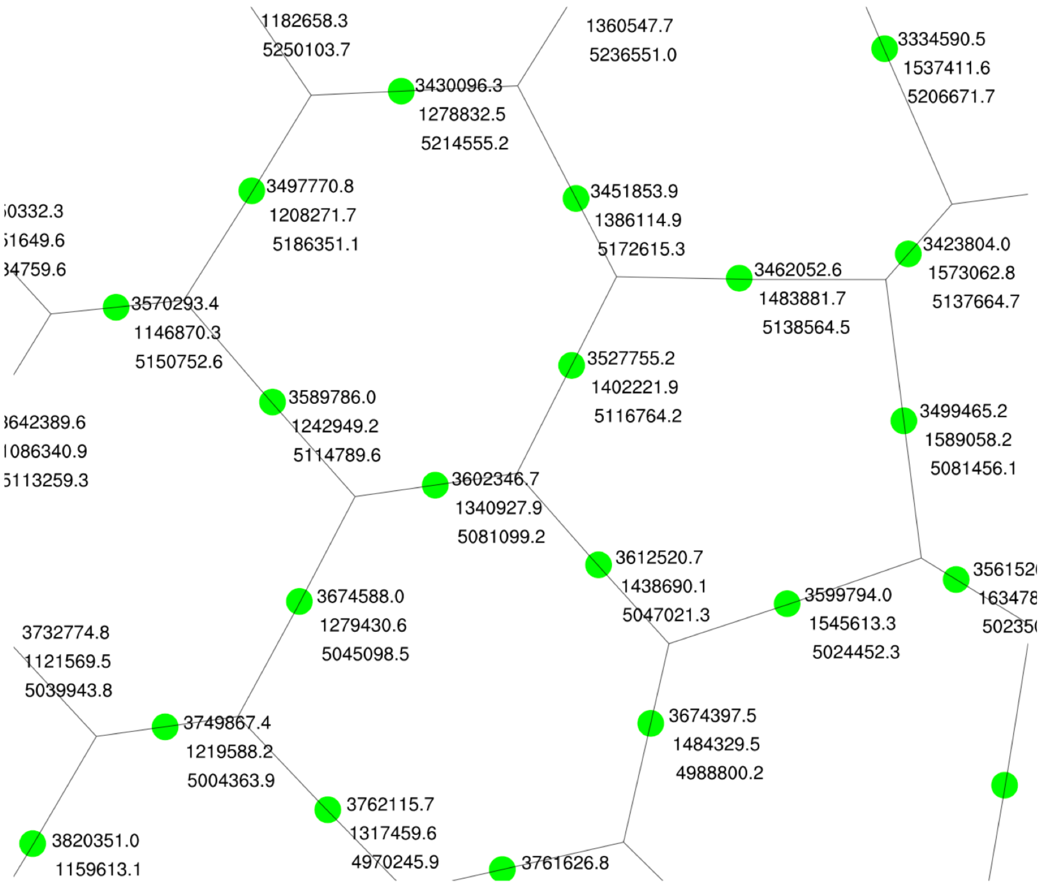
- For planar meshes, coordinates lie in the plane z=0
- For spherical meshes, coordinates lie on the surface of the sphere

For cells: **xCell**, **yCell**, **zCell**

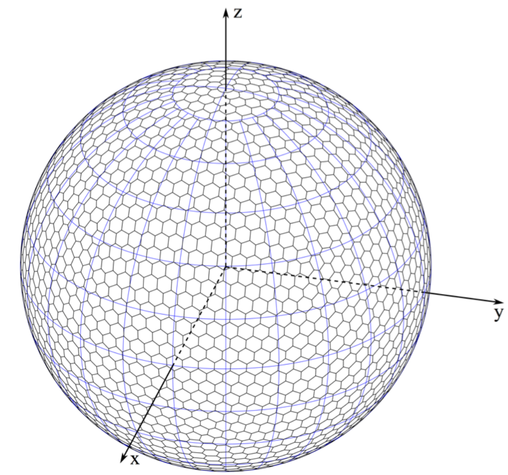
Latitudes and longitudes are computed from Cartesian coordinates as described earlier

- positive x-axis through 0° longitude
- positive y-axis through 90° longitude
- positive z-asix through 90° latitude

Notes about MPAS mesh geometry



Above: Cartesian coordinates for cell locations near (52.9°N lat, 20.8°E lon) in a variable-resolution spherical mesh with radius 6371229 m.



Global Cartesian coordinates are computed for each element

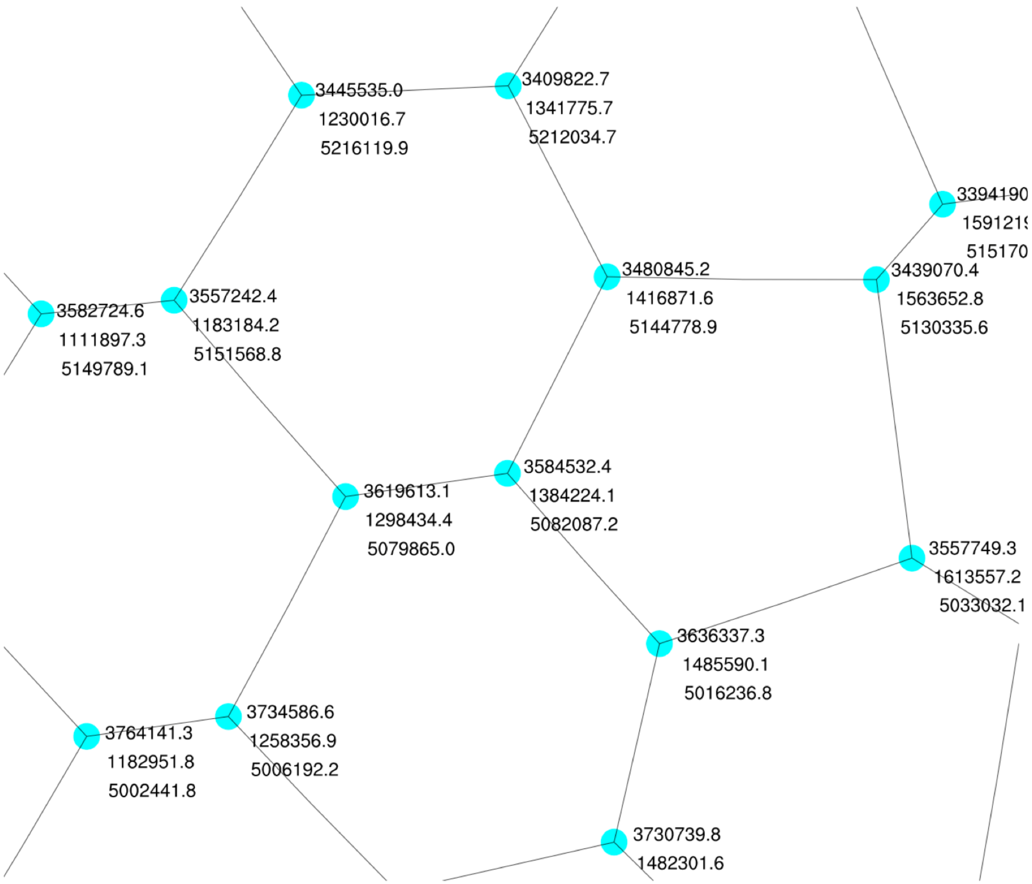
- For planar meshes, coordinates lie in the plane z=0
- For spherical meshes, coordinates lie on the surface of the sphere

For cells: **xEdge**, **yEdge**, **zEdge**

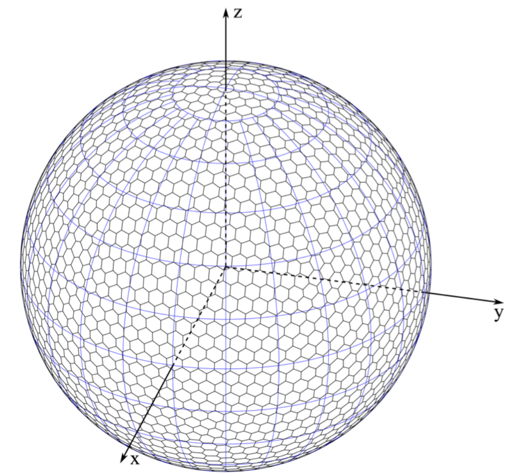
Latitudes and longitudes are computed from Cartesian coordinates as described earlier

- positive x-axis through 0° longitude
- positive y-axis through 90° longitude
- positive z-asix through 90° latitude

Notes about MPAS mesh geometry



Above: Cartesian coordinates for cell locations near (52.9°N lat, 20.8°E lon) in a variable-resolution spherical mesh with radius 6371229 m.



Global Cartesian coordinates are computed for each element

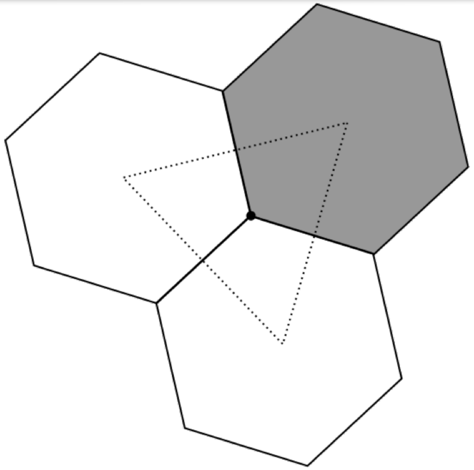
- For planar meshes, coordinates lie in the plane z=0
- For spherical meshes, coordinates lie on the surface of the sphere

For cells: **xVertex**, **yVertex**, **zVertex**

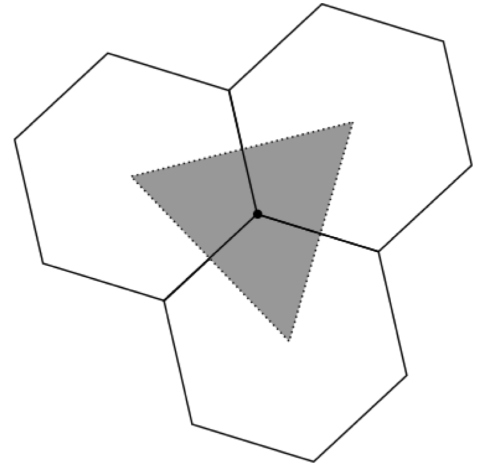
Latitudes and longitudes are computed from Cartesian coordinates as described earlier

- positive x-axis through 0° longitude
- positive y-axis through 90° longitude
- positive z-asix through 90° latitude

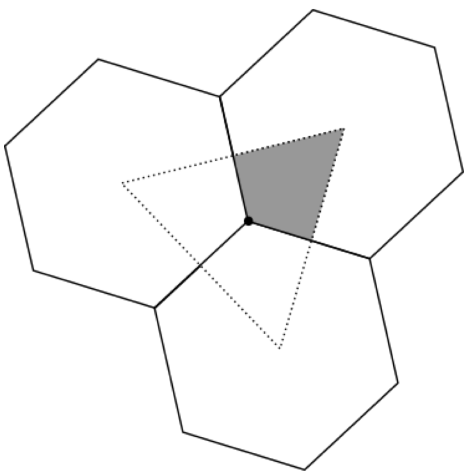
Mesh geometry fields in MPAS



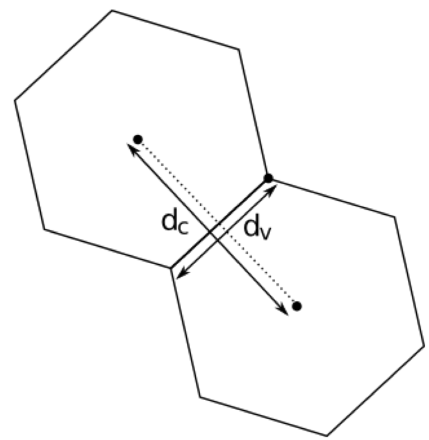
areaCell(nCells) – area of each cell



areaTriangle(nVertices) – area of each dual-grid cell



kiteAreasOnVertex(vertexDegree,nVertices) – area of intersection between dual- and primal-mesh cells



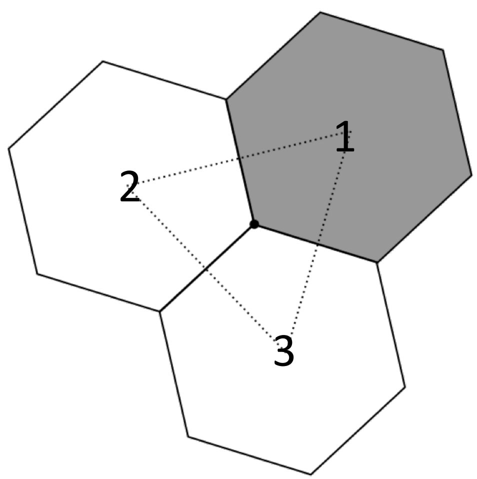
dcEdge(nEdges) – distances between cell centers
dvEdge(nEdges) – length of each edge

MPAS
Model for Prediction Across Scales

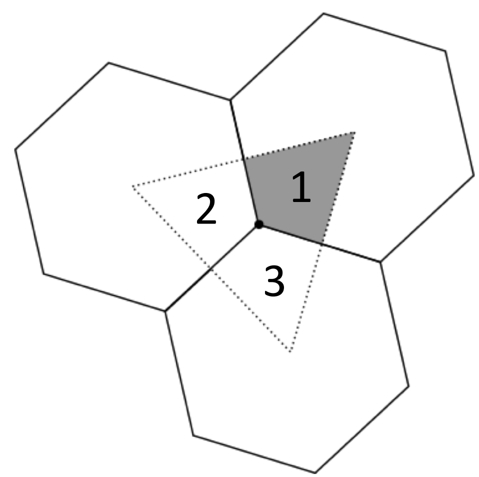
Indexing of mesh fields, both connectivity and geometry

- For *On* arrays (e.g., cellsOnCell), elements are listed in anti-clockwise order
  - Whenever possible, starting points are consistent between indexing arrays (e.g., cellsOnVertex and kiteAreasOnVertex)
  - E.g., the first edgeOnCell separates a given cell from the first cellOnCell

- All indices are 1-based
  - (MPAS is written in Fortran, after all...)



cellsOnVertex(:,V)



kiteAreasOnVertex(:,V)

What do MPAS meshes look like in netCDF files?

When stored in netCDF files ("*grid.nc*"), MPAS meshes have at least the following dimensions:

```
dimensions:
   nCells = 40962 ;
   nEdges = 122880 ;
   nVertices = 81920 ;
   maxEdges = 10 ;
   maxEdges2 = 20 ;
   TWO = 2 ;
   vertexDegree = 3 ;
```

The number of cells, edges, and vertices in the mesh.

For global, spherical meshes:
nVertices = 2 * (nCells - 2)
nEdges = 3 * (nCells - 2)

For *doubly-periodic* planar meshes:
nEdges = nCells + nVertices

For *limited-area* meshes:
nEdges + 1 = nCells + nVertices

When stored in netCDF files ("*grid.nc*"), MPAS meshes have at least the following dimensions:

```
dimensions:
   nCells = 40962 ;
   nEdges = 122880 ;
   nVertices = 81920 ;
   maxEdges = 10 ;
   maxEdges2 = 20 ;
   TWO = 2 ;
   vertexDegree = 3 ;
```

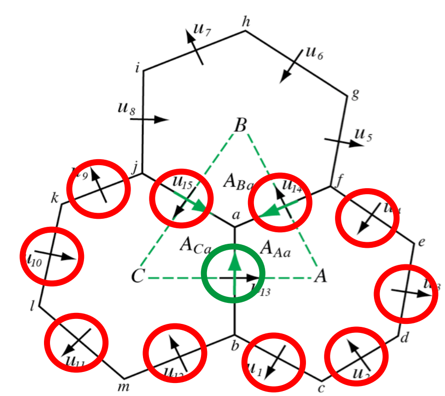
The maximum number of faces (edges) any cell can have; equivalent to the maximum number of cell neighbors or vertices that a cell can have.

What do MPAS meshes look like in netCDF files?

When stored in netCDF files (“*grid.nc*”), MPAS meshes have at least the following dimensions:

```
dimensions:
   nCells = 40962 ;
   nEdges = 122880 ;
   nVertices = 81920 ;
   maxEdges = 10 ;
   maxEdges2 = 20 ;
   TWO = 2 ;
   vertexDegree = 3 ;
```

The maximum number of edges that participate in the reconstruction of tangential velocities at cell faces (edges).



What do MPAS meshes look like in netCDF files?

When stored in netCDF files ("*grid.nc*"), MPAS meshes have at least the following dimensions:

```
dimensions:
    nCells = 40962 ;
    nEdges = 122880 ;
    nVertices = 81920 ;
    maxEdges = 10 ;
    maxEdges2 = 20 ;
    TWO = 2 ;
    vertexDegree = 3 ;
```

Always 2 (every dimension must have a name in netCDF). Used for, e.g., the number of vertices forming the endpoints of edges and the number of cells separated by an edge.

What do MPAS meshes look like in netCDF files?

When stored in netCDF files ("*grid.nc*"), MPAS meshes have at least the following dimensions:

```
dimensions:
   nCells = 40962 ;
   nEdges = 122880 ;
   nVertices = 81920 ;
   maxEdges = 10 ;
   maxEdges2 = 20 ;
   TWO = 2 ;
   vertexDegree = 3 ;
```

The number of cells/edges that meet at each vertex.

In principle, quadrilateral meshes could be represented by setting vertexDegree = 4

Tutorial Outline

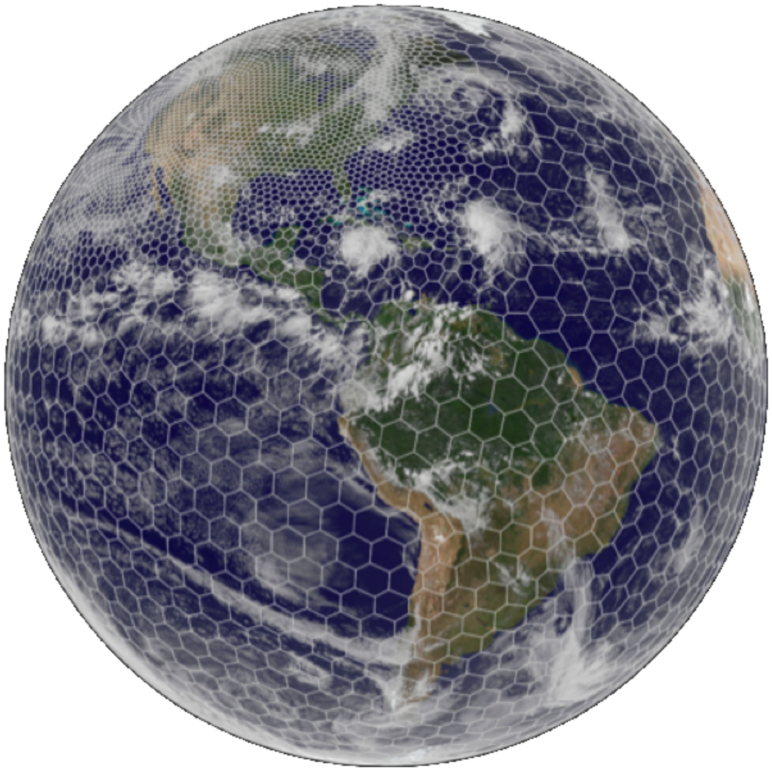
Introduction

The MPAS mesh representation

Obtaining, compiling, and running MPAS

Physics and dynamics configuration

Post-processing and visualization



1. **Obtaining MPAS source code**
2. Preliminary requirements
3. Compiling MPAS-Atmosphere
4. Creating initial conditions
5. Running the model

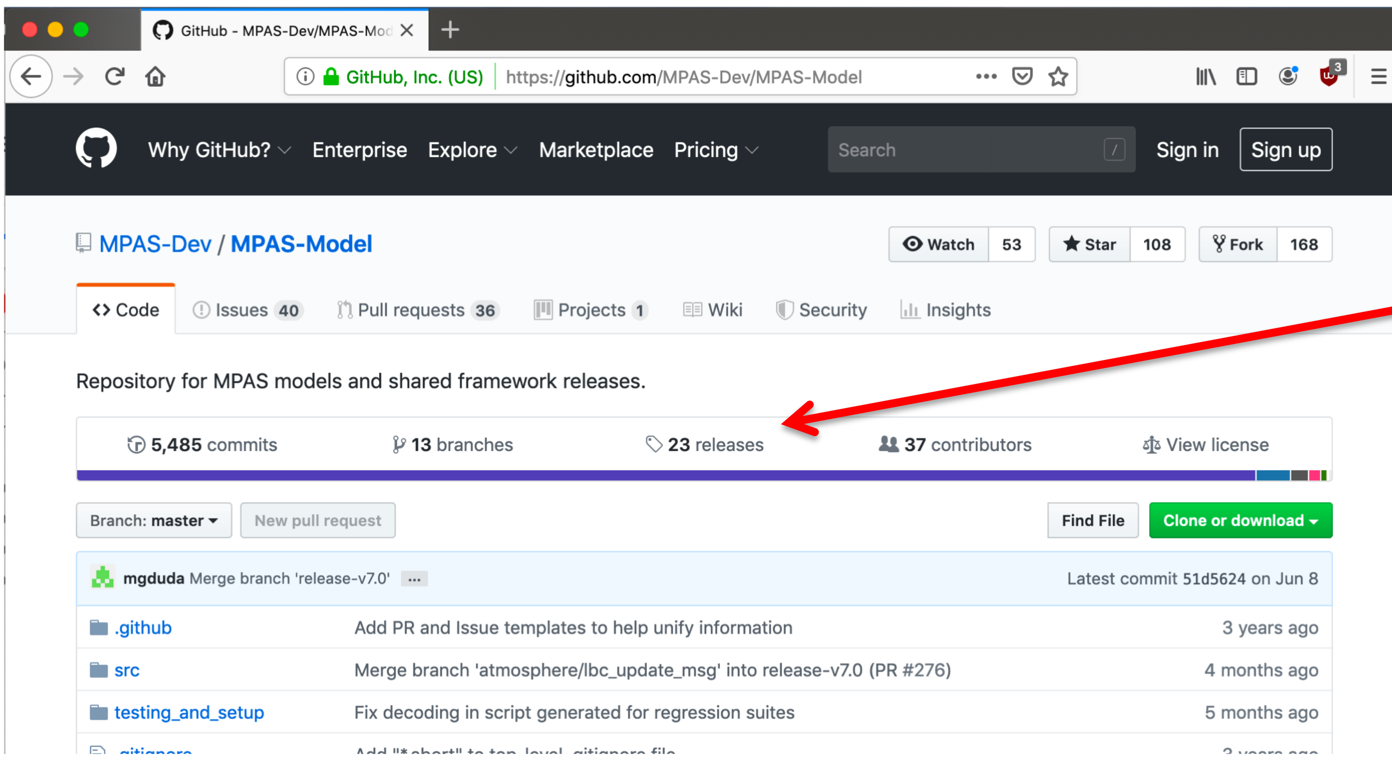
https://mpas-dev.github.io/

Obtaining the MPAS source code

Option 1: Simply clone the MPAS-Model repository from GitHub

```
git clone https://github.com/MPAS-Dev/MPAS-Model.git
```

Option 2: Download a tar file from GitHub: https://github.com/MPAS-Dev/MPAS-Model

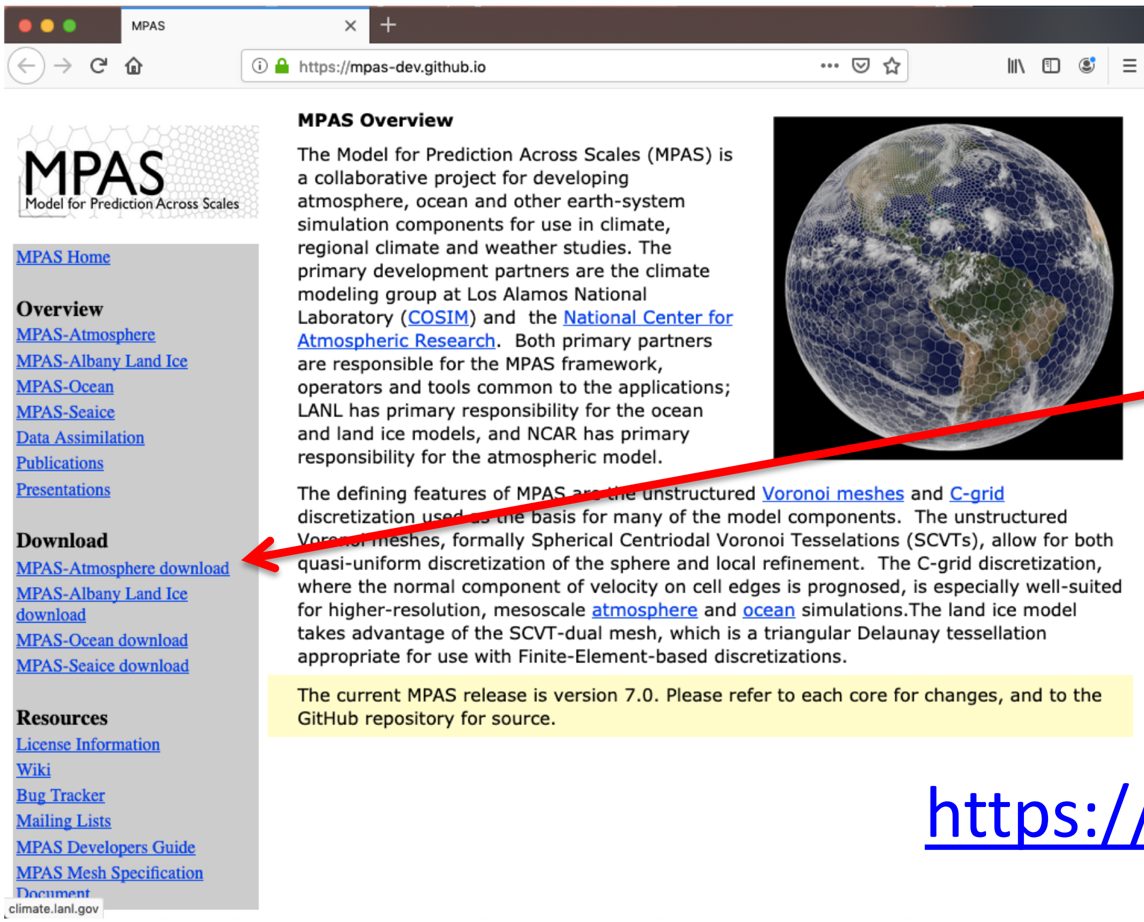


Click here!

Obtaining the MPAS source code

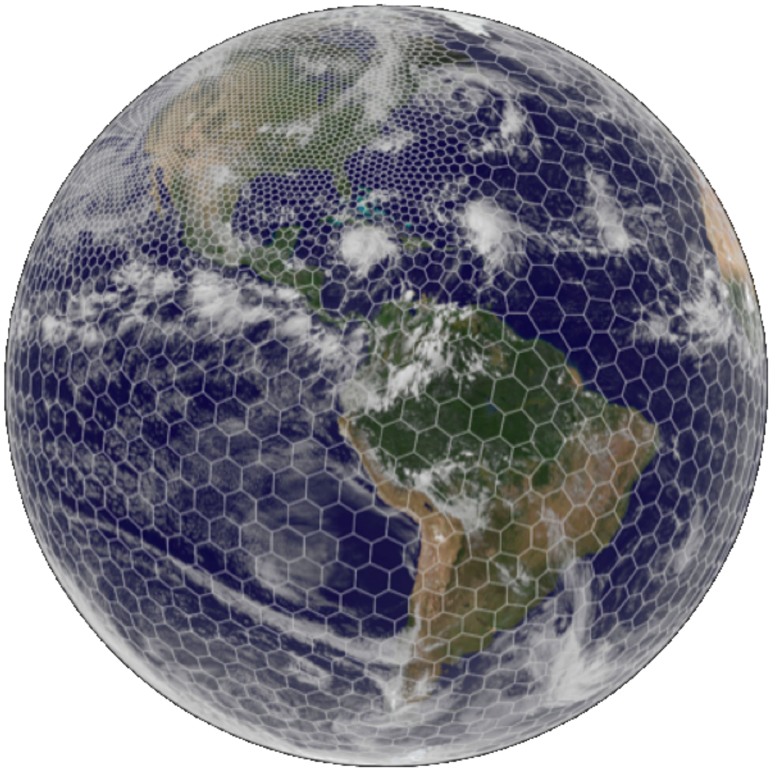
<u>Option 3</u>: Go to the MPAS homepage, register as a new user, join the MPAS-Atmosphere Users mailing list, get a copy of the users' guide, and download the latest release as in Options 1 or 2.

- *<u>This is the recommended (and kindly requested) option for new users!</u>*



Click on "MPAS-Atmosphere download" to get to everything mentioned above

https://mpas-dev.github.io



1. Obtaining MPAS source code
2. **Preliminary requirements**
3. Compiling MPAS-Atmosphere
4. Creating initial conditions
5. Running the model

https://mpas-dev.github.io/

Preliminary requirements

In order to compile MPAS and its required libraries, working C and Fortran compilers are necessary

- The Fortran compiler should be recent enough to support the ISO_C_BINDING module from the Fortran 2003 standard and procedure pointer components of derived types
- Most versions of common compilers from the last couple of years should be fine

Building MPAS requires *at least* the following libraries:

- Any implementation of MPI-2, e.g., MPICH, MVAPICH, OpenMPI
  - Ensure that `mpif90` and `mpicc` commands are in your path
- Parallel-netCDF (http://trac.mcs.anl.gov/projects/parallel-netcdf/)
  - Set `PNETCDF` environment variable to base installation directory
- PIO (https://github.com/NCAR/ParallelIO/)
  - Set `PIO` environment variable to base installation directory

MPAS
Model for Prediction Across Scales

Preliminary requirements: the easiest route

Assuming Fortran and C compilers are available, and a working MPI installation is also available, installing Parallel-netCDF and PIO should take less than 10 minutes:

Parallel-netCDF 1.8.1

```
$ setenv CC gcc
$ setenv FC gfortran
$ setenv F77 gfortran
$ setenv MPICC mpicc
$ setenv MPIF90 mpif90
$ setenv MPIF77 mpif90
$ setenv PNETCDF /home/duda/pnetcdf
$ cd parallel-netcdf-1.8.1
$ ./configure \
  --prefix=$PNETCDF \
  --disable-cxx
$ make
$ make install
```

PIO 1.7.1

(Assuming environment variables from Parallel-NetCDF installation)

```
$ setenv MPIFC mpif90
$ setenv PNETCDF_PATH $PNETCDF
$ setenv PIO /home/duda/pio
$ cd pio1_7_1/pio
$ ./configure \
  --prefix=$PIO \
  --disable-netcdf \
  --disable-mpiio
$ make
$ make install
```

Which versions of PIO should I try?

The PIO library is undergoing rapid development, and many different versions of the library are available; *which versions are supported and recommended?*

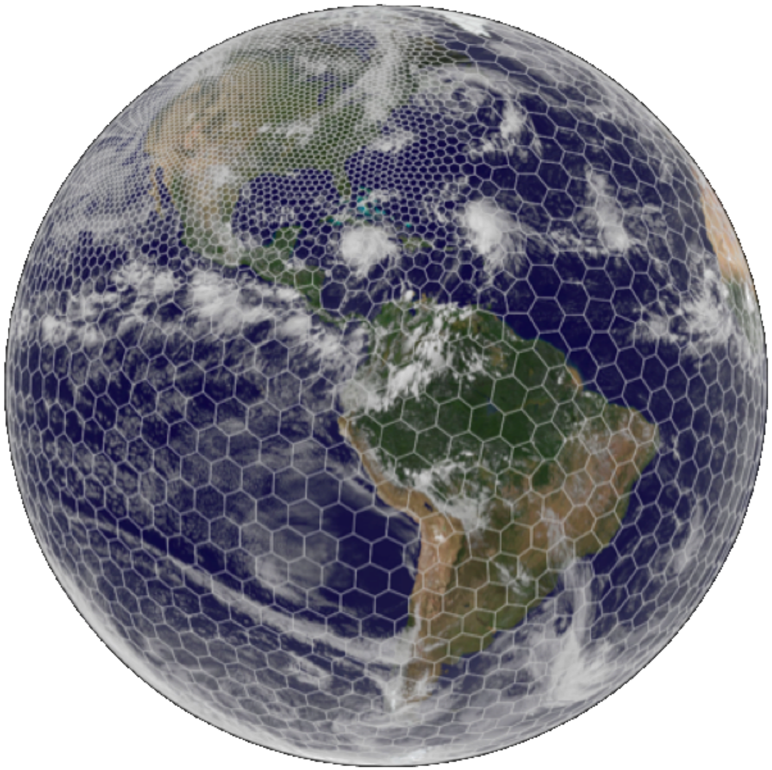
1) For ease of installation, try PIO 1.7.1

- Can be installed using only standard 'configure' and 'make' tools
- Supports NetCDF-3 and Parallel-netCDF I/O
- Download: https://github.com/NCAR/ParallelIO/releases/tag/pio1_7_1

2) If netCDF-4 I/O is needed or desirable, try the latest PIO release

- Requires recent versions of *'cmake'*, plus standard 'make'
- Supports netCDF-3, netCDF-4 (in parallel via PHDF5), and Parallel-netCDF I/O
- Download: https://github.com/NCAR/ParallelIO/
- See iolib_installation.sh at http://www2.mmm.ucar.edu/people/duda/files/mpas/sources/

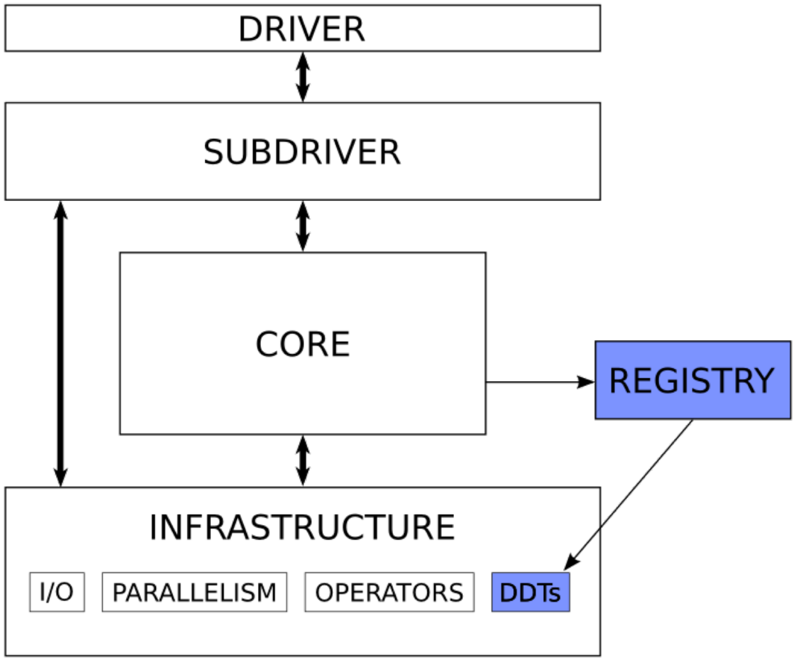
Obtaining, compiling, and running MPAS



1. Obtaining MPAS source code
2. Preliminary software requirements
3. **Compiling MPAS-Atmosphere**
4. Creating initial conditions
5. Running the model

https://mpas-dev.github.io/

Model Organization

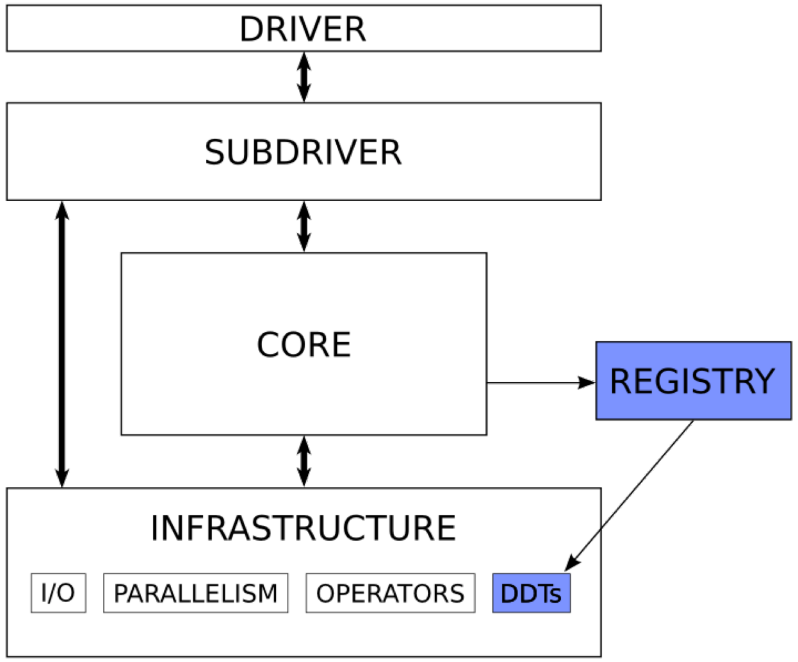


Checking out the MPAS code provides all MPAS models, not just MPAS-Atmosphere

- All models share a common set of infrastructure modules
- Each MPAS model is implemented as a "core" that lives in its own directory
- User must select which "core" to compile
- Each "core" is associated with a source code subdirectory under src/ and has a Registry file (similar to WRF)

```
MPAS-Model/
    src/
        core_atmosphere/
        core_init_atmosphere/
        core_landice/
        core_ocean/
        driver/
        external/
        framework/
        operators/
```

Model Organization



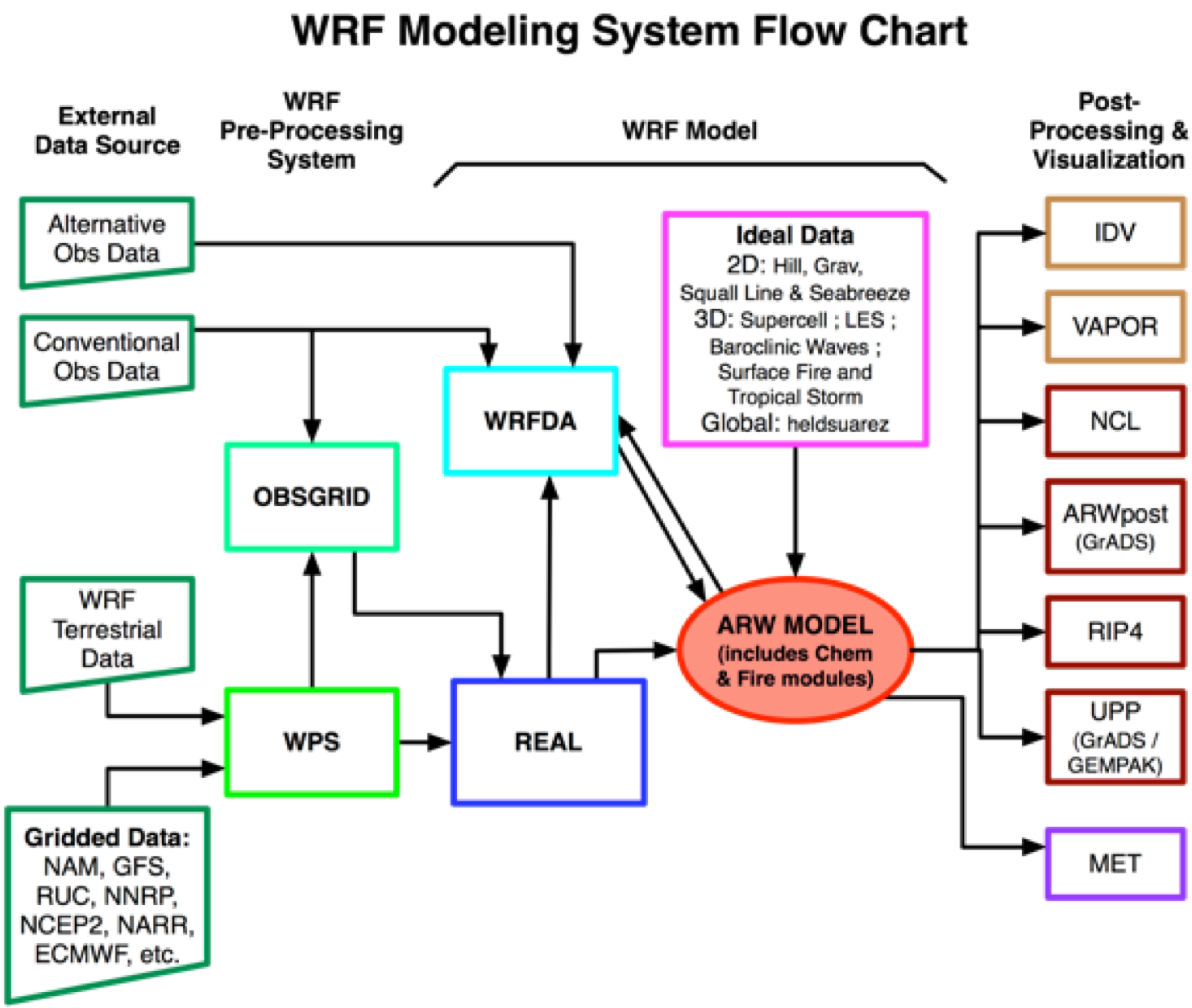
Checking out the MPAS code provides all MPAS models, not just MPAS-Atmosphere

- All models share a common set of infrastructure modules
- Each MPAS model is implemented as a “core” that lives in its own directory
- User must select which “core” to compile
- Each “core” is associated with a source code subdirectory under `src/` and has a Registry file (similar to WRF)

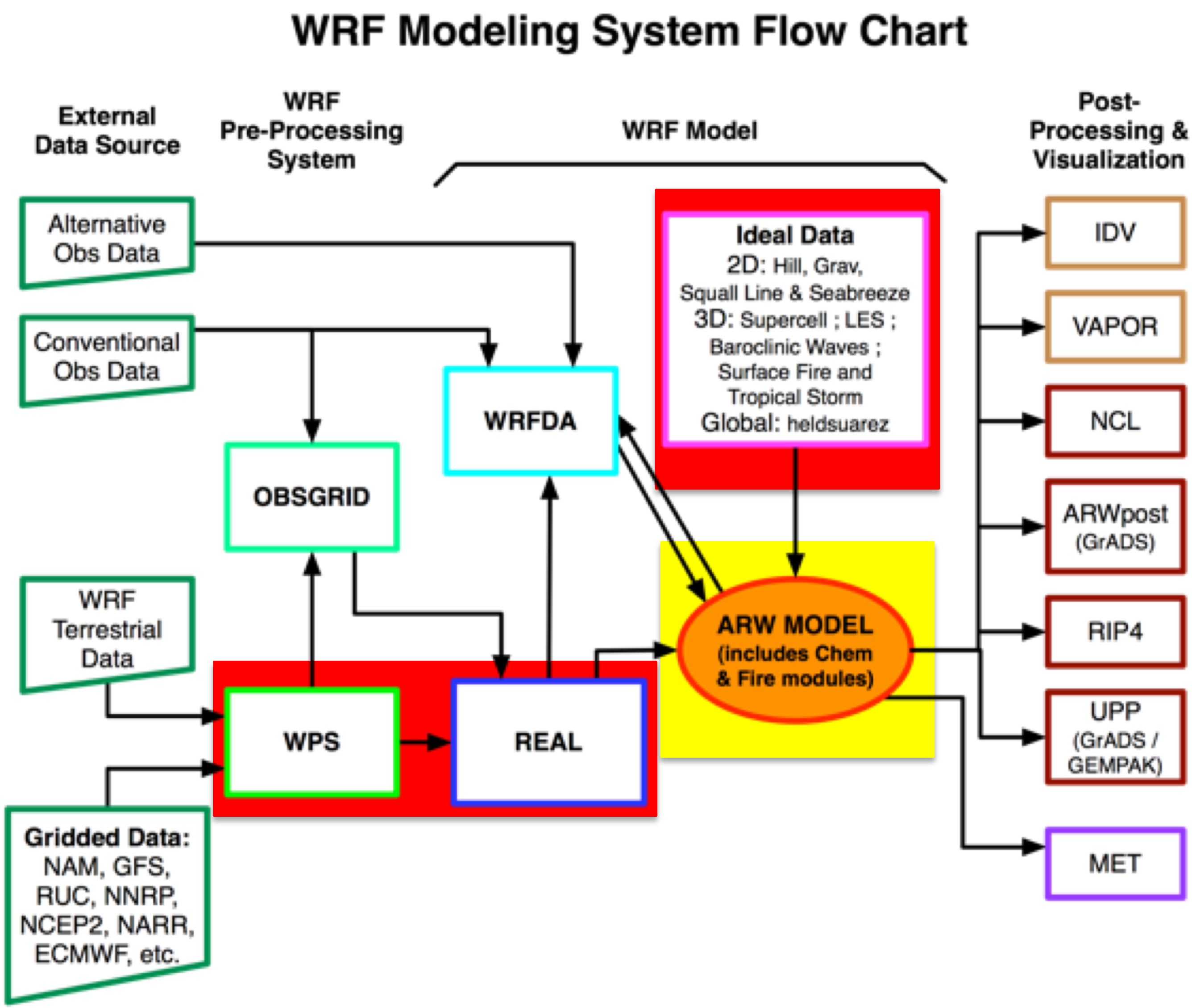
Running MPAS-Atmosphere involves two “cores”:

- The **init_atmosphere** core is responsible for
  - Interpolating static fields to the mesh (similar to geogrid.exe)
  - Generating a vertical grid (similar to real.exe)
  - Horizontally and vertically interpolating meteorological data to the 3-d grid (similar to metgrid.exe and real.exe)
  - *Where to we get meteorological data? From ungrib.exe!*
- The **atmosphere** core is the model itself, the equivalent of wrf.exe

Thinking about MPAS cores in terms of WRF components



Thinking about MPAS cores in terms of WRF components



Equivalent MPAS functionality handled by:

init_atmosphere

atmosphere

Compiling MPAS

There is no “configuration” step for MPAS, unlike, e.g., for the WRF model

- All build flags are either set in the top-level Makefile or on the command-line

General MPAS build command:

```
$ make target CORE=core <options>
```

target can be either

clean

or

xlf
gfortran
ifort
pgi
llvm
... plus a few others...

For MPAS-Atmosphere, core may be

atmosphere
init_atmosphere

<options> can be zero or more of

DEBUG=true
AUTOCLEAN=true
PRECISION=single
OPENMP=true

Compiling MPAS

Typical build of both the init_atmosphere and atmosphere cores involves:

$ make gfortran CORE=init_atmosphere *(build init_atmosphere_model)*

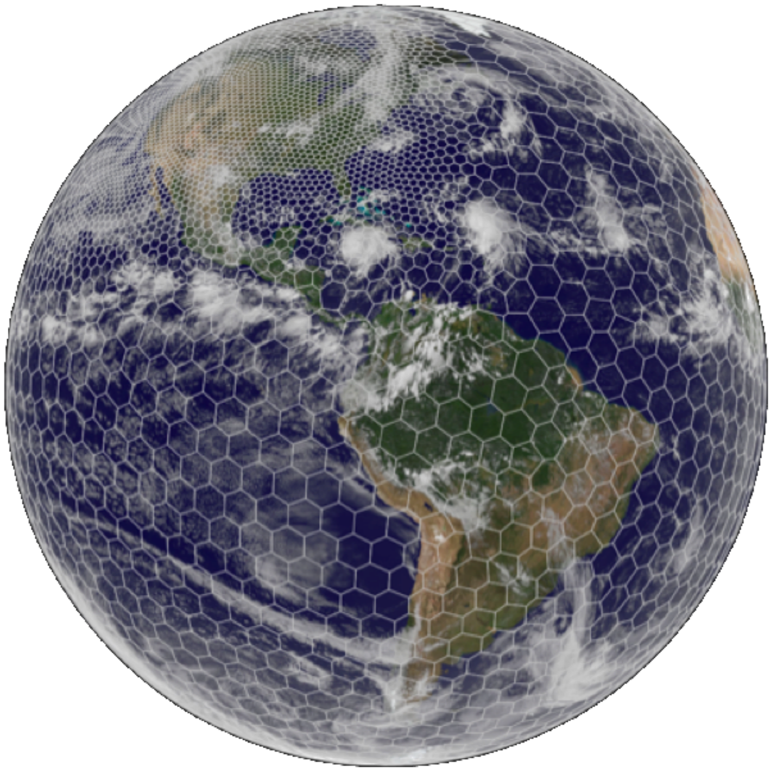
$ make clean CORE=atmosphere *(clean any infrastructure files used by both init_atmosphere and atmosphere)*

$ make gfortran CORE=atmosphere *(build atmosphere_model)*

By default, MPAS cores are built with double-precision reals

MPAS-Atmosphere can be built in single precision

- Add `PRECISION=single` to build commands for single-precision executables
- execution time ~35% less compared with double-precision
- output files approximately half as large
- Beginning with MPAS v3.0, it is possible to run the model in double precision while writing history files in single precision!



1. Obtaining MPAS source code
2. Preliminary requirements
3. Compiling MPAS-Atmosphere
4. **Creating initial conditions**

   But first, a few digressions...
5. Running the model

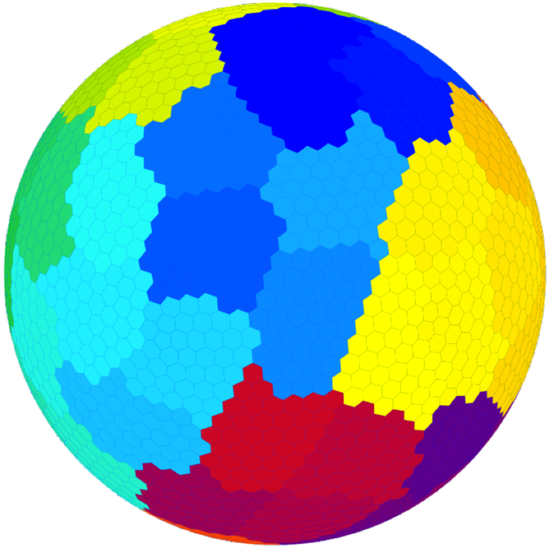
https://mpas-dev.github.io/

Digression: Mesh partition files

Recall that MPAS meshes must be partitioned using *Metis* in order for MPAS to be run in parallel

Conveniently, the meshes available from the MPAS-Atmosphere download page are provided with several pre-computed partition files

- **In many cases, it may not be necessary for you to run Metis yourself; just use a pre-computed partitioning**



For example, the x1.40962 mesh (about 120-km resolution) is provided with the following files:

`x1.40962.grid.nc` – the mesh itself
`x1.40962.graph.info` – the mesh connectivity graph
`x1.40962.graph.info.part.2` – pre-computed partitioning for 2 MPI tasks
`x1.40962.graph.info.part.8` – pre-computed partitioning for 8 MPI tasks
`x1.40962.graph.info.part.16` – pre-computed partitioning for 16 MPI tasks
...

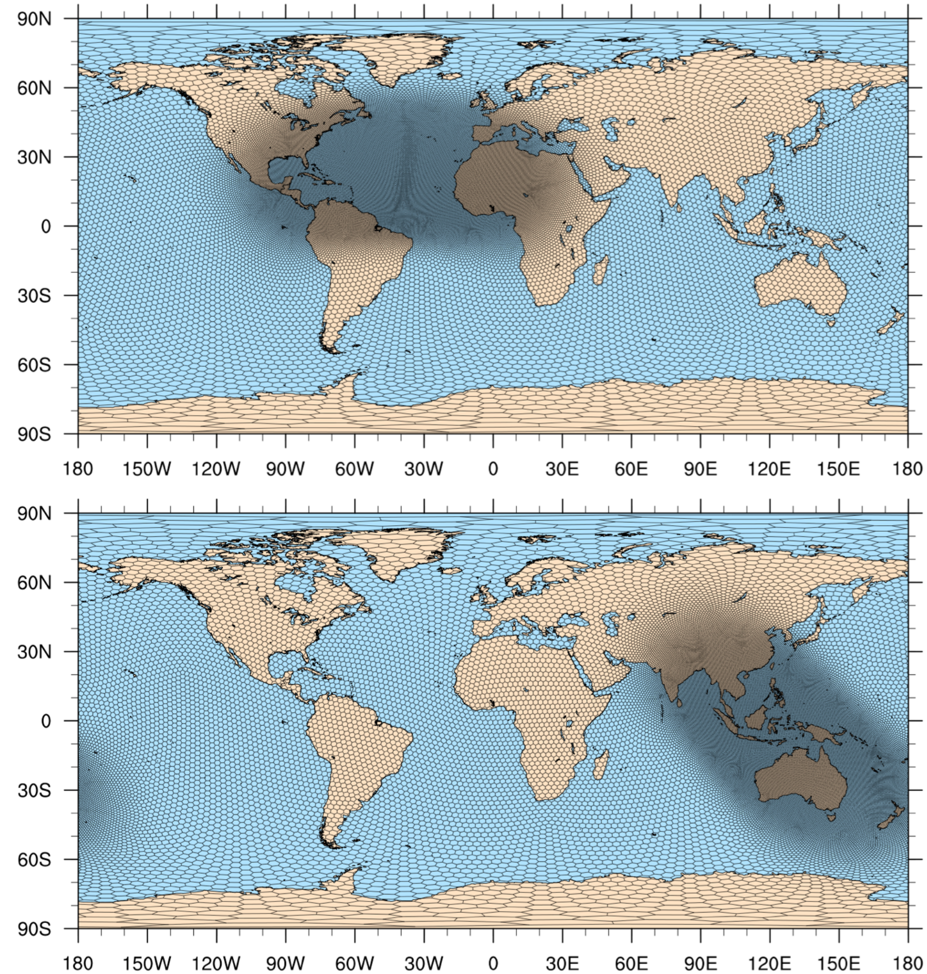
Digression: Preparing variable-resolution meshes

Before interpolating any fields to the mesh, the location of refinement can be moved:

Using the *grid_rotate* utility available through the MPAS-Atmosphere mesh download page, specify:

- Reference point location (lat,lon) in the original grid – usually the center of the refinement region
- Reference point location (lat,lon) in the rotated mesh
- A counter-clockwise rotation of the mesh about the vector from the center of the sphere through the reference point

NB: The original graph decomposition files can still be used, since cell connectivity does not change!



Above: A refinement region originally centered at 25N, 40W has been shifted to 7S, 125E and rotated by -45 degrees.

Digression: Preparing limited-area meshes

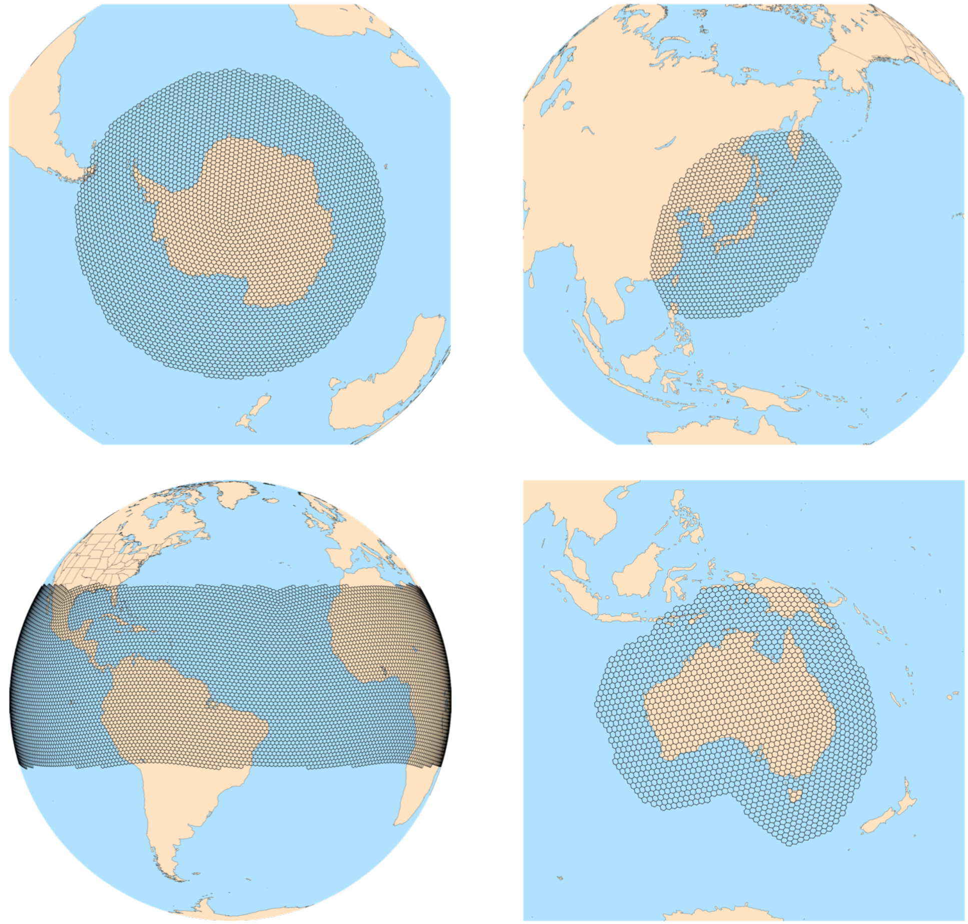
For limited-area simulations, use the MPAS-Limited-Area tool

Limited-area meshes may be created by subsetting any existing MPAS meshes

- After creating a limited-area mesh, use Metis to create *partition* files for the mesh

- In addition to initial conditions, limited-area simulations require lateral boundary conditions

- **See Sections 4.3 and 8.2 in the User's Guide for more details…**



https://github.com/MiCurry/MPAS-Limited-Area.git

Digression: Controlling MPAS input and output

Unlike in WRF, the netCDF files (input, history, restart, user-defined) that are read and written by MPAS are defined in a “streams” file

- For the *init_atmosphere* core, the file is named `streams.init_atmosphere`
- For the *atmosphere* core, the file is named `streams.atmosphere`

Example:

```
<stream name="diagnostics"
        type="output"
        filename_template="diag.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="3:00:00" >

        <var name=”mslp"/>
        <var name=”height_500hPa"/>
        <var name=”rainc"/>
        <var name=”rainnc"/>

</stream>
```

Refer to Chapter 5 of the MPAS-Atmosphere Users’ Guide for a complete description of options for controlling input and output.

Digression: Controlling MPAS input and output

Example:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

This stream is named “input”

This stream is named “restart”

This stream is named “output”

Digression: Controlling MPAS input and output

Example:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

This stream is only read by MPAS

This stream is both read and written

This stream is only written

Digression: Controlling MPAS input and output

Example:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

This stream reads from a file named "x1.4002.init.nc"

This stream reads and writes from files with names of this form

This stream writes to files with names of this form

Digression: Controlling MPAS input and output

Example:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

This stream is read only at the start of execution

This stream is read only at the start of execution

This stream is written every 1 day

This stream is written every 6 hours

Digression: Controlling MPAS input and output

Example:

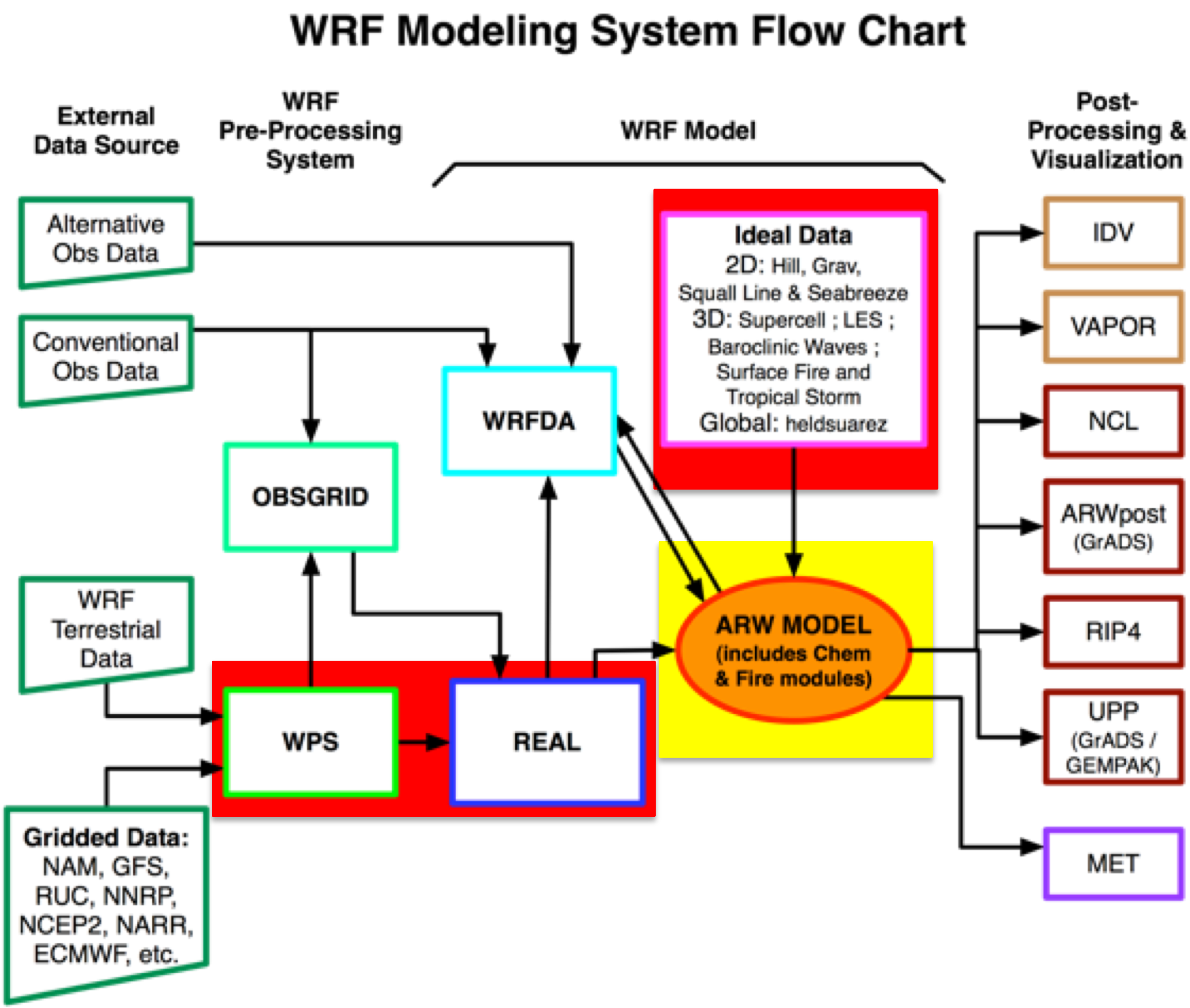
```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

The variables read/written by immutable streams may not be changed at runtime

This is the set of variables written by this stream

MPAS
Model for Prediction Across Scales

Initial conditions (real and idealized) are all created using the init_atmosphere core



Equivalent MPAS functionality handled by:

init_atmosphere

atmosphere

How does the ‘init_atmosphere’ core manage to combine all of the functionality of the WPS, real.exe, and ideal.exe programs into one!?

- The key idea is that `init_atmosphere_model` may be run in stages using different options

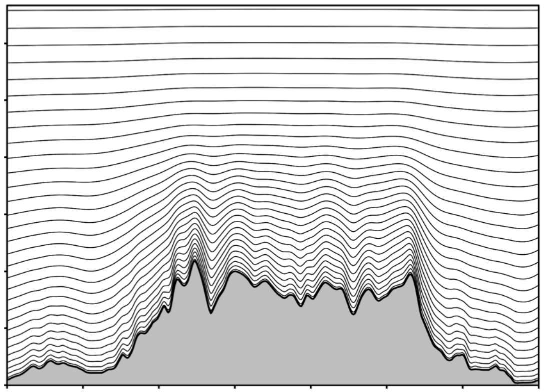
```
&nhyd_model
    config_init_case = 7
    ...
/

&preproc_stages
    config_static_interp = false
    config_native_gwd_static = false
    config_vertical_grid = true
    config_met_interp = true
    config_input_sst = false
    config_frac_seaice = true
/
```

2 = ideal baroclinic wave
4 = ideal squall line
5 = ideal supercell
6 = ideal mountain wave
7 = real-data initialization
8 = surface update file creation (like sst_update=1 in WRF)
9 = lateral boundary conditions

Only used for real-data cases to control, e.g., whether we interpolate static fields, compute fields for GWDO scheme, interpolate meteorological data, etc.

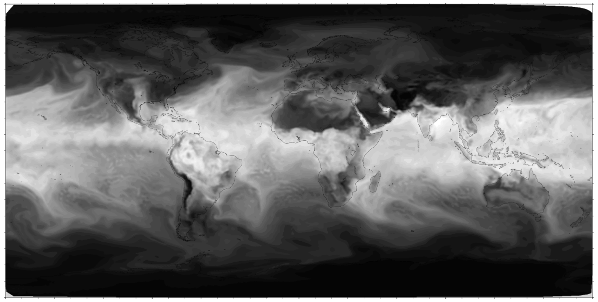
Overview of steps to run global MPAS-Atmosphere

1. Interpolate geographical fields to the mesh: terrain, landuse, soil type, etc.

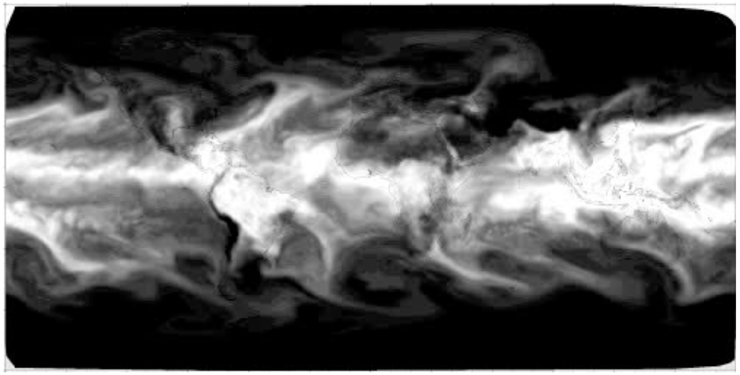


2. Generate the vertical grid

3. Prepare GRIB data with the WPS *ungrib.exe*



4. Interpolate initial conditions to the 3d mesh



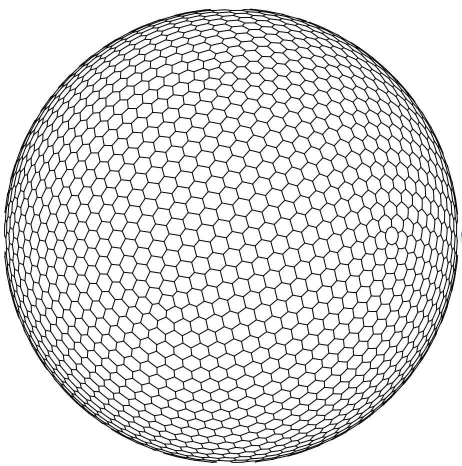
5. Run the MPAS-Atmosphere model itself

init_atmosphere_model   ungrib.exe   atmosphere_model

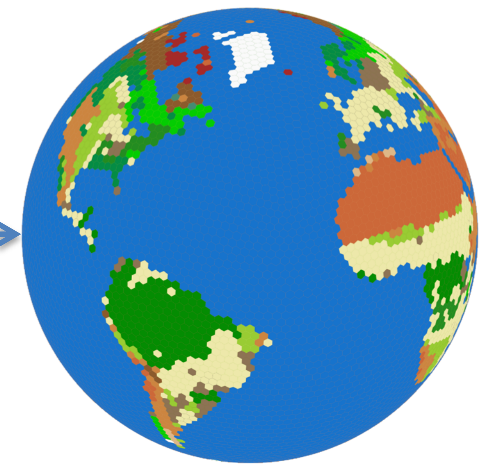
Creating real-data initial conditions: geographic fields

For real-data initial conditions, the first step is to interpolate static, geographic fields to the mesh

- Geographic datasets come from the WRF Pre-processing System
- Interpolation typically takes about 45 minutes
- *Must be run with just one MPI task!*



init_atmosphere_model



See MPAS-Atmosphere Users' Guide, section 7.2.1 for important namelist settings...

"input" stream filename_template:

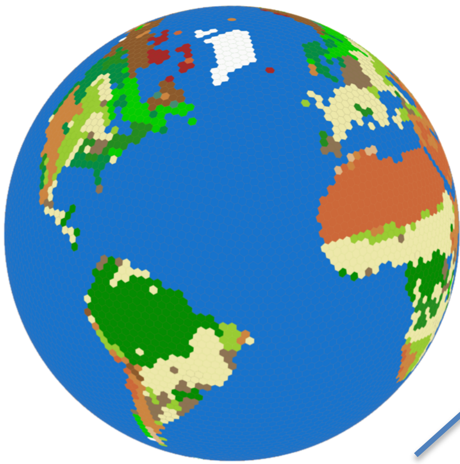
x1.40962.grid.nc

"output" stream filename_template:

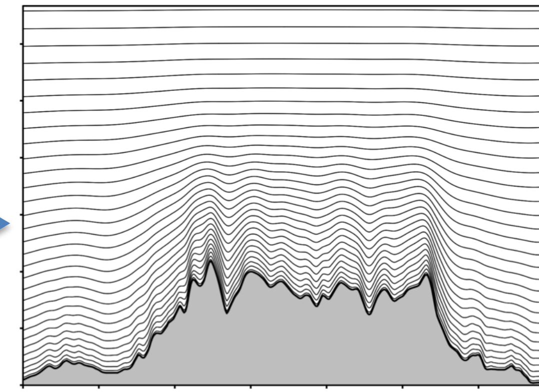
x1.40962.static.nc

Creating real-data initial conditions: meteorological fields

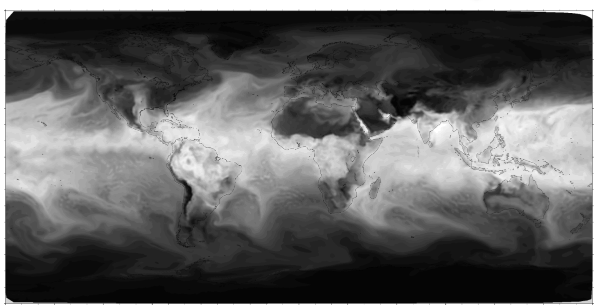
After interpolating geographic fields to the horizontal mesh, we can generate a vertical grid and interpolate meteorological initial conditions



init_atmosphere_model

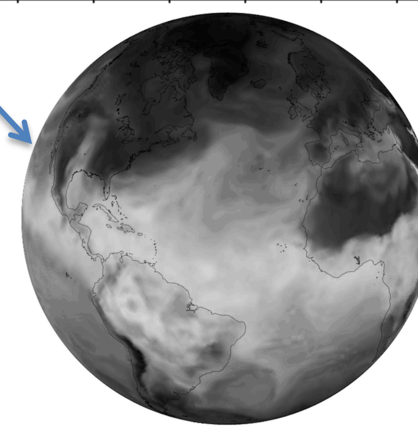


See MPAS-Atmosphere Users' Guide, section 7.2.2 for important namelist settings...



"input" stream filename_template:

x1.40962.static.nc



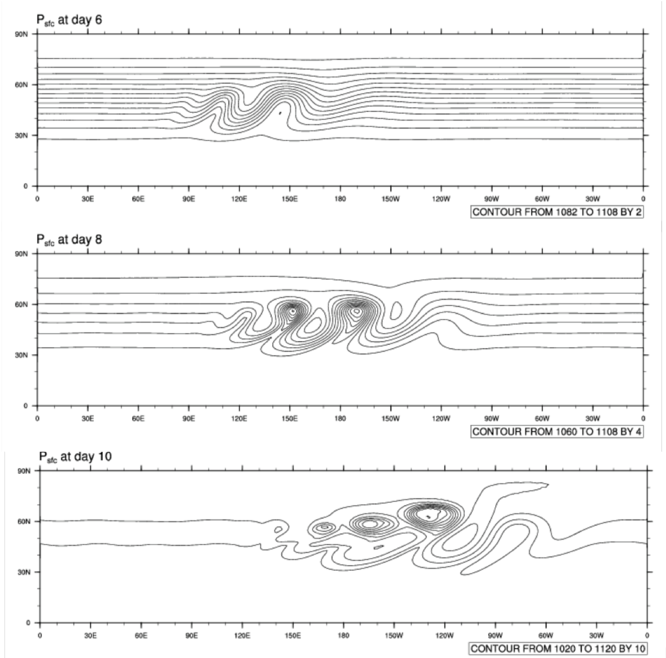
"output" stream filename_template:

x1.40962.init.nc

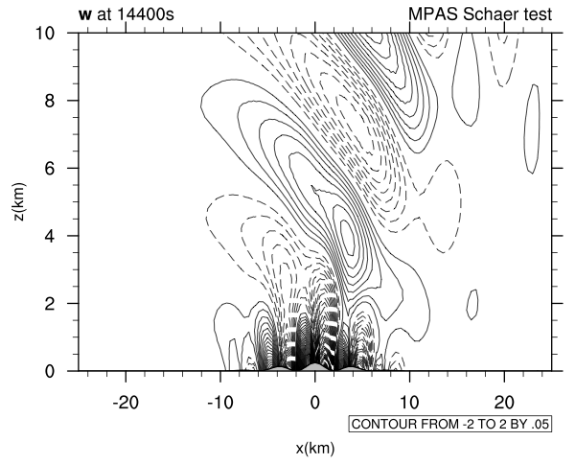
Creating idealized initial conditions

Besides real-data cases, there are several idealized cases that MPAS-Atmosphere supports “out-of-the-box”

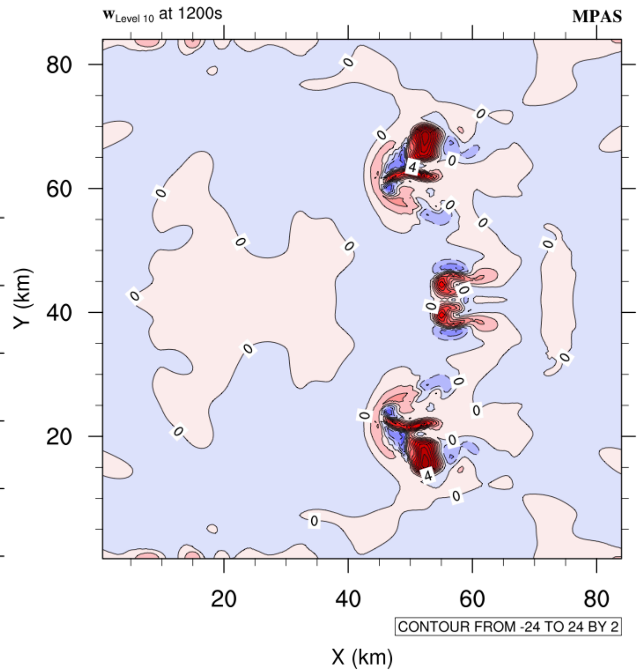
- Meshes and namelist files for these cases can be found through the MPAS-Atmosphere download page



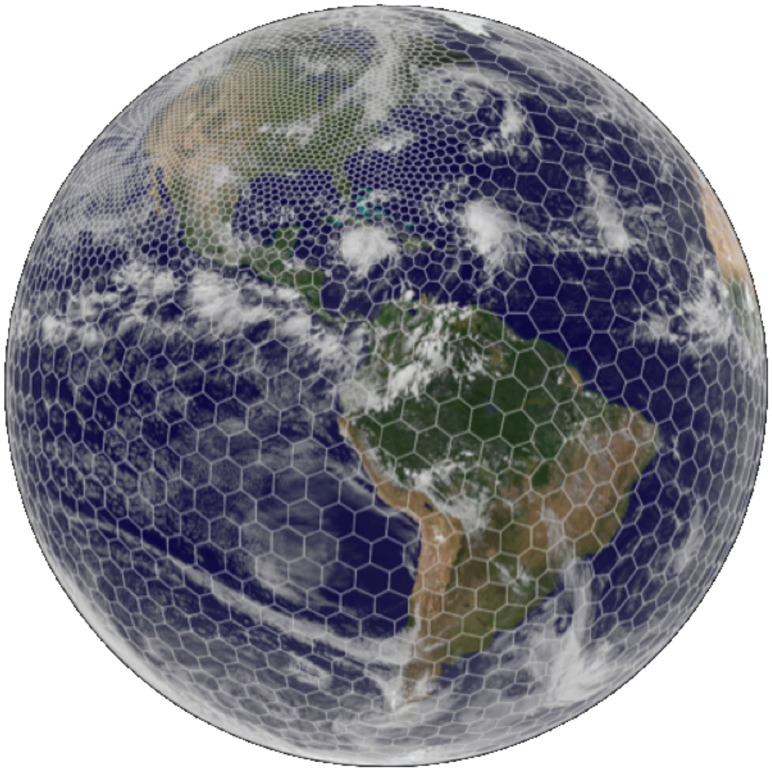
Jablonowski and Williamson baroclinic wave



Schaer mountain wave



Supercell thunderstorm



1. Obtaining MPAS source code
2. Preliminary software requirements
3. Compiling MPAS-Atmosphere
4. Creating initial conditions
5. **Running the model**

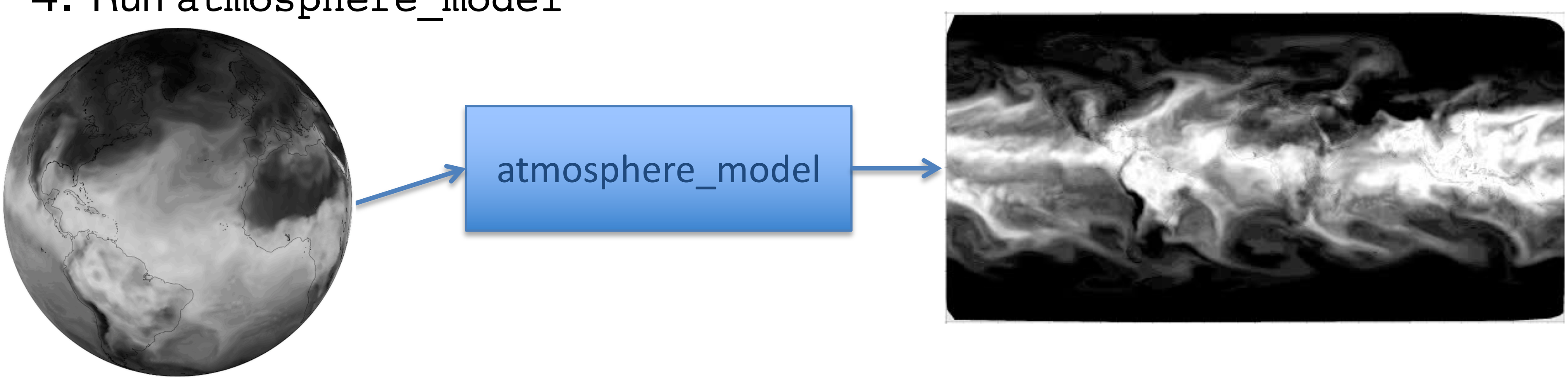
https://mpas-dev.github.io/

Running the MPAS-Atmosphere model

The same `atmosphere_model` executable can be used for either real-data or idealized simulations

Given initial conditions (e.g., `x1.40962.init.nc`), all that is needed to run the model is to:

1. Edit the `namelist.atmosphere` file to set model timestep, mixing and damping parameters, physics options, etc.
2. Edit the `streams.atmosphere` file to specify the name of the input initial conditions file and the frequency of model history files
3. Ensure that the proper mesh partition file (e.g., `x1.40962.graph.info.part.64`) is present
4. Run `atmosphere_model`



Running the MPAS-Atmosphere model (2)

Before running the model itself (`atmosphere_model`), verify that the following namelist options have been properly set:

- **config_dt** – The model timestep, in seconds; with MPAS v7.0, try starting with a timestep of between 5 and 6 times the minimum model grid spacing in kilometers; also ensure that model output interval is evenly divided by the timestep
- **config_len_disp** – The length-scale for explicit horizontal mixing; set this to the minimum grid distance (in meters) in the mesh

Besides these crucial namelist options, ensure that the names of input and output files are correctly set in the `streams.atmosphere` file!

Running the MPAS-Atmosphere model (3)

As the model runs, information about the progress of the model is written to the file `log.atmosphere.0000.out`

- This is the equivalent of the WRF `rsl.error.0000` file

One can tail this file to check on model progress, e.g.,

```
$ tail -f log.atmosphere.0000.out
```

```
Begin timestep 2017-06-12_01:00:00
--- time to run the LW radiation scheme L_RADLW = T
--- time to run the SW radiation scheme L_RADSW = T
--- time to run the convection scheme L_CONV    = T
--- time to apply limit to accumulated rainc and rainnc L_ACRAIN   = F
--- time to apply limit to accumulated radiation diags. L_ACRADT   = F
--- time to calculate additional physics_diagnostics               = F
 split dynamics-transport integration            3

global min, max w  -0.4467210       1.098162
global min, max u   -89.13145       88.83957
Timing for integration step:    0.3368 s
```

Above: Example output for a timestep in the log file from a typical model run.

Model for Prediction Across Scales

Tutorial Outline

Introduction

The MPAS mesh representation

Obtaining, compiling, and running MPAS

Physics and dynamics configuration

Post-processing and visualization

MPAS
Model for Prediction Across Scales

Configuring the dynamics and the physics

(namelist.atmosphere)

```
&nhyd_model
  config_dt = 90.0
  config_start_time = '2010-10-23_00:00:00'
  config_run_duration = '5_00:00:00'
  config_split_dynamics_transport = true
  config_number_of_sub_steps = 2
  config_dynamics_split_steps = 3
  config_h_mom_eddy_visc2 = 0.0
  config_h_mom_eddy_visc4 = 0.0
  config_v_mom_eddy_visc2 = 0.0
  config_h_theta_eddy_visc2 = 0.0
  config_h_theta_eddy_visc4 = 0.0
  config_v_theta_eddy_visc2 = 0.0
  config_horiz_mixing = '2d_smagorinsky'
  config_h_ScaleWithMesh = true
  config_len_disp = 15000.0
  config_visc4_2dsmag = 0.05
  config_del4u_div_factor = 1.0
  config_w_adv_order = 3
  config_theta_adv_order = 3
  config_scalar_adv_order = 3
  config_u_vadv_order = 3
  config_w_vadv_order = 3
  config_theta_vadv_order = 3
  config_scalar_vadv_order = 3
  config_scalar_advection = true
  config_positive_definite = false
  config_monotonic = true
  config_coef_3rd_order = 0.25
  config_epssm = 0.1
  config_smdiv = 0.1
  config_apvm_upwinding = 0.5
```

Time and time-steps

```
&nhyd_model
  config_dt = 90
  config_start_time = ”2010-10-23_00:00:00"
  config_run_duration = ”5_00:00:00”
  config_split_dynamics_transport = true
  config_number_of_sub_steps = 2
  config_dynamics_split_steps = 3
```

Timestep in seconds

Dynamics substeps per split step

Number of acoustic steps per timestep

MPAS
Model for Prediction Across Scales

Time Integration
Dynamics and Scalar Transport Options

Split-transport integration

Call physics

Do dynamics_split_steps
    Do step_rk3 = 1, 3
        compute large-time-step tendency
        Do acoustic_steps
            update u
            update rho, theta and w
        End acoustic_steps
    End rk3 step
End dynamics_split_steps

Do scalar step_rk3 = 1, 3
    scalar RK3 transport
End scalar rk3 step

Call microphysics

Allows for smaller dynamics timesteps relative to scalar transport timestep and main physics timestep.

We can use any FV scheme here (we are not tied to RK3)
Scalar transport and physics are the expensive pieces in most applications.

MPAS
Model for Prediction Across Scales

Configuring the dynamics

(namelist.atmosphere)

```
&nhyd_model
   config_dt = 90.0
   config_start_time = '2010-10-23_00:00:00'
   config_run_duration = '5_00:00:00'
   config_split_dynamics_transport = true
   config_number_of_sub_steps = 2
   config_dynamics_split_steps = 3
   config_h_mom_eddy_visc2 = 0.0
   config_h_mom_eddy_visc4 = 0.0
   config_v_mom_eddy_visc2 = 0.0
   config_h_theta_eddy_visc2 = 0.0
   config_h_theta_eddy_visc4 = 0.0
   config_v_theta_eddy_visc2 = 0.0
   config_horiz_mixing = '2d_smagorinsky'
   config_h_ScaleWithMesh = true
   config_len_disp = 15000.0
   config_visc4_2dsmag = 0.05
   config_del4u_div_factor = 1.0
   config_w_adv_order = 3
   config_theta_adv_order = 3
   config_scalar_adv_order = 3
   config_u_vadv_order = 3
   config_w_vadv_order = 3
   config_theta_vadv_order = 3
   config_scalar_vadv_order = 3
   config_scalar_advection = true
   config_positive_definite = false
   config_monotonic = true
   config_coef_3rd_order = 0.25
   config_epssm = 0.1
   config_smdiv = 0.1
   config_apvm_upwinding = 0.5
```

Time and time-steps

&nhyd_model

config_epssm = 0.1
time-offcentering of the vertically implicit acoustic and gravity-wave integration.

config_smdiv = 0.1
3D divergence damping

config_apvm_upwinding = 0.5
Anticipated Potential Vorticity Method (APVM): upwind-biased estimate of edge PV; provides an enstrophy sink.

MPAS
Model for Prediction Across Scales

Configuring the dynamics and the physics

Advection

(namelist.atmosphere)

```
&nhyd_model
   config_dt = 90.0
   config_start_time = '2010-10-23_00:00:00'
   config_run_duration = '5_00:00:00'
   config_split_dynamics_transport = true
   config_number_of_sub_steps = 2
   config_dynamics_split_steps = 3
   config_h_mom_eddy_visc2 = 0.0
   config_h_mom_eddy_visc4 = 0.0
   config_v_mom_eddy_visc2 = 0.0
   config_h_theta_eddy_visc2 = 0.0
   config_h_theta_eddy_visc4 = 0.0
   config_v_theta_eddy_visc2 = 0.0
   config_horiz_mixing = '2d_smagorinsky'
   config_h_ScaleWithMesh = true
   config_len_disp = 15000.0
   config_visc4_2dsmag = 0.05
   config_del4u_div_factor = 1.0
   config_w_adv_order = 3
   config_theta_adv_order = 3
   config_scalar_adv_order = 3
   config_u_vadv_order = 3
   config_w_vadv_order = 3
   config_theta_vadv_order = 3
   config_scalar_vadv_order = 3
   config_scalar_advection = true
   config_positive_definite = false
   config_monotonic = true
   config_coef_3rd_order = 0.25
   config_epssm = 0.1
   config_smdiv = 0.1
   config_apvm_upwinding = 0.5
```

```
&nhyd_model
   config_w_adv_order = 3
   config_theta_adv_order = 3
   config_scalar_adv_order = 3
   config_u_vadv_order = 3
   config_w_vadv_order = 3
   config_theta_vadv_order = 3
   config_scalar_vadv_order = 3
   config_positive_definite = .false.
   config_monotonic = .true.
   config_coef_3rd_order = 0.25
```

Advection scheme order (2, 3, or 4)

PD/Mono options for scalar transport

Upwind coefficient (0 <-> 1), >0 increases damping

MPAS
Model for Prediction Across Scales

Configuring the dynamics and the physics

Dissipation

(namelist.atmosphere)

```
&nhyd_model
   config_dt = 90.0
   config_start_time = '2010-10-23_00:00:00'
   config_run_duration = '5_00:00:00'
   config_split_dynamics_transport = true
   config_number_of_sub_steps = 2
   config_dynamics_split_steps = 3
   config_h_mom_eddy_visc2 = 0.0
   config_h_mom_eddy_visc4 = 0.0
   config_v_mom_eddy_visc2 = 0.0
   config_h_theta_eddy_visc2 = 0.0
   config_h_theta_eddy_visc4 = 0.0
   config_v_theta_eddy_visc2 = 0.0
   config_horiz_mixing = '2d_smagorinsky'
   config_h_ScaleWithMesh = true
   config_len_disp = 15000.0
   config_visc4_2dsmag = 0.05
   config_del4u_div_factor = 1.0
   config_w_adv_order = 3
   config_theta_adv_order = 3
   config_scalar_adv_order = 3
   config_u_vadv_order = 3
   config_w_vadv_order = 3
   config_theta_vadv_order = 3
   config_scalar_vadv_order = 3
   config_scalar_advection = true
   config_positive_definite = false
   config_monotonic = true
   config_coef_3rd_order = 0.25
   config_epssm = 0.1
   config_smdiv = 0.1
   config_apvm_upwinding = 0.5
```

```
&nhyd_model
   config_h_mom_eddy_visc2 = 0
   config_h_mom_eddy_visc4 = 0
   config_v_mom_eddy_visc2 = 0
   config_h_theta_eddy_visc2 = 0
   config_h_theta_eddy_visc4 = 0
   config_v_theta_eddy_visc2 = 0
   config_horiz_mixing = "2d_smagorinsky"
   config_len_disp = 15000.
   config_visc4_2dsmag = 0.05
   config_h_ScaleWithMesh = .true.
   config_del4u_div_factor = 1.0
```

fixed viscosity $m^2 s^{-1}$

Fixed hyper-viscosity $m^4 s^{-1}$

Alternately "2d_fixed"

Δx_fine

Scale viscosities, hyperviscosities with local mesh spacing

4th order background filter coef, used with 2d_smagorinsky

υ_4 (m^4/s) = config_len_disp^3 x config_visc4_2dsmag

MPAS
Model for Prediction Across Scales

Configuring the dynamics

(namelist.atmosphere)

```
&damping
   config_zd = 22000.0
   config_xnutr = 0.2
/
&physics
   config_sst_update = false
   config_sstdiurn_update = false
   config_deepsoiltemp_update = false
   config_radtlw_interval = '00:30:00'
   config_radtsw_interval = '00:30:00'
   config_bucket_update = 'none'
   config_physics_suite = 'mesoscale_reference'
```

Gravity-wave absorbing layer

&damping

config_zd = 22000.

Bottom of the gravity-wave absorbing layer (meters)
Note: WRF defines this parameter as the depth of the layer.

config_xnutr = 0.2

Gravity-wave absorbing layer damping coefficient

This is the same formulation as in WRF

Configuring the Physics

(namelist.atmosphere)

```
&physics
   config_sst_update = false
   config_sstdiurn_update = false
   config_deepsoiltemp_update = false
   config_radtlw_interval = '00:30:00'
   config_radtsw_interval = '00:30:00'
   config_bucket_update = 'none'
   config_physics_suite = 'mesoscale_reference'
```

```
&physics
config_physics_suite = 'mesoscale_reference'
```

Mesoscale reference physics suite – MPAS V7.0

Surface Layer: (Monin Obukhov): module_sf_sfclay.F as in WRF 4.0.3
PBL: YSU as in WRF 4.0.3
Land Surface Model (NOAH 4-layers): as in WRF 4.0.3.
Gravity Wave Drag: YSU gravity wave drag scheme, as in WRF 4.0.3
Convection: new Tiedtke (nTiedtke), as in WRF 4.0.3
Microphysics: WSM6: as in WRF 4.1
Radiation: RRTMG sw as in WRF 3.8.1; RRTMG lw as in WRF 3.8.1
Cloud fraction for radiation: Xu-Randall
Ocean Mixed Layer: modified and extended from WRFV3.6

MPAS
Model for Prediction Across Scales

Configuring the Physics

(namelist.atmosphere)

```
&physics
   config_sst_update = false
   config_sstdiurn_update = false
   config_deepsoiltemp_update = false
   config_radtlw_interval = '00:30:00'
   config_radtsw_interval = '00:30:00'
   config_bucket_update = 'none'
   config_physics_suite = 'convection_permitting'
```

```
&physics
config_physics_suite = 'convection_permitting'
```

Convection-permitting physics suite – MPAS V7.0

Surface Layer: module_sf_mynn.F as in WRF 3.6.1
PBL: Mellor-Yamada-Nakanishi-Niino (MYNN) as in WRF 3.6.1
Land Surface Model (NOAH 4-layers): as in WRF 4.0.3.
Gravity Wave Drag: YSU gravity wave drag scheme, as in WRF 4.0.3
Convection: Grell-Freitas scale aware scheme (modified from WRF 3.6.1)
Microphysics: Thompson scheme (non-aerosol aware): as in WRF 3.8.1
Radiation: RRTMG sw as in WRF 3.8.1; RRTMG lw as in WRF 3.8.1
Cloud fraction for radiation: Xu-Randall

MPAS
Model for Prediction Across Scales

Configuring the Physics

Physics components can be individually selected, and a suite component can be overridden, by adding the appropriate configuration setting to the &physics section of the namelist.atmosphere.

Table 6.3: Possible options for individual physics parameterizations. Namelist variables should be added to the &physics namelist record.

Parameterization	Namelist variable	Possible options	Details
Convection	config_convection_scheme	cu_tiedtke	Tiedtke (WRF 3.8.1)
		cu_ntiedtke	New Tiedtke (WRF 4.0.3)
		cu_grell_freitas	Modified version of scale-aware Grell-Freitas (WRF 3.6.1)
		cu_kain_fritsch	Kain-Fritsch (WRF 3.2.1)
Microphysics	config_microp_scheme	mp_wsm6	WSM 6-class (WRF 4.1)
		mp_thompson	Thompson non-aerosol aware (WRF 3.8.1)
		mp_kessler	Kessler
Land surface	config_lsm_scheme	noah	Noah (WRF 4.0.3)
Boundary layer	config_pbl_scheme	bl_ysu	YSU (WRF 4.0.3)
		bl_mynn	MYNN (WRF 3.6.1)
Surface layer	config_sfclayer_scheme	sf_monin_obukhov	Monin-Obukhov (WRF 4.0.3)
		sf_mynn	MYNN (WRF 3.6.1)
Radiation, LW	config_radt_lw_scheme	rrtmg_lw	RRTMG (WRF 3.8.1)
		cam_lw	CAM (WRF 3.3.1)
Radiation, SW	config_radt_sw_scheme	rrtmg_sw	RRTMG (WRF 3.8.1)
		cam_sw	CAM (WRF 3.3.1)
Cloud fraction for radiation	config_radt_cld_scheme	cld_fraction	Xu and Randall (1996)
		cld_incidence	0/1 cloud fraction depending on $q_c + q_i$
Gravity wave drag by orography	config_gwdo_scheme	bl_ysu_gwdo	YSU (WRF 4.0.3)

'off' specifies no physics component for that parameterization
'suite' means take the suite definition (default)

Tutorial Outline

Introduction

The MPAS mesh representation

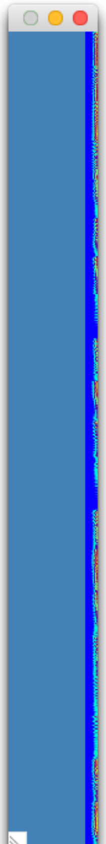
Obtaining, compiling, and running MPAS

Physics and dynamics configuration

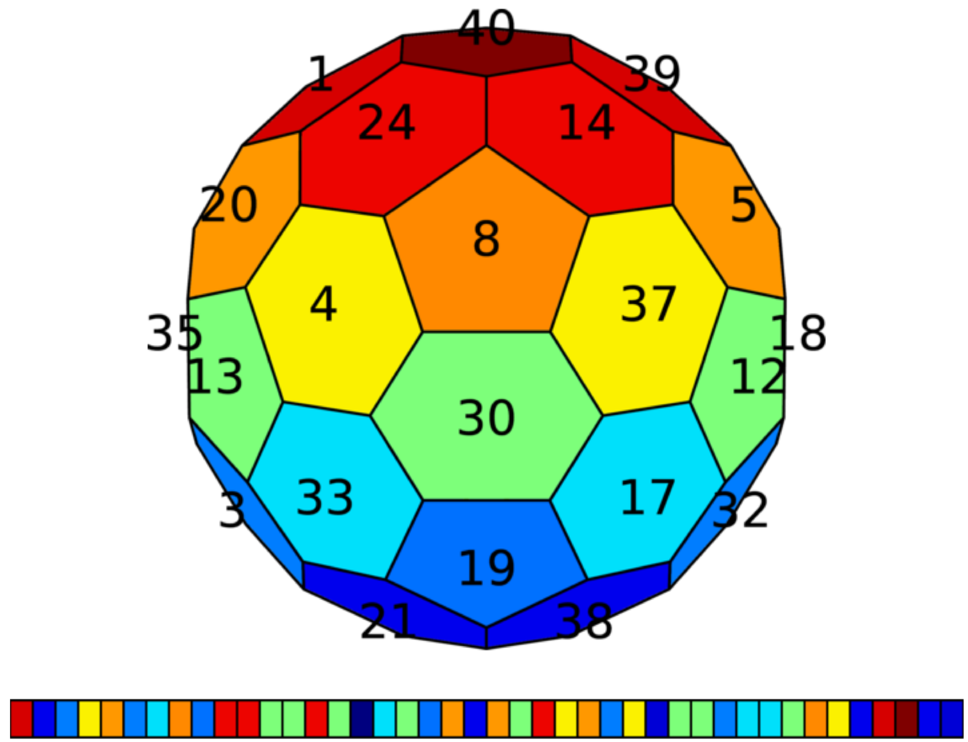
Post-processing and visualization

Now that you've got MPAS output, how can you do something useful with it?

Using 'ncview' directly on MPAS netCDF files doesn't work well...

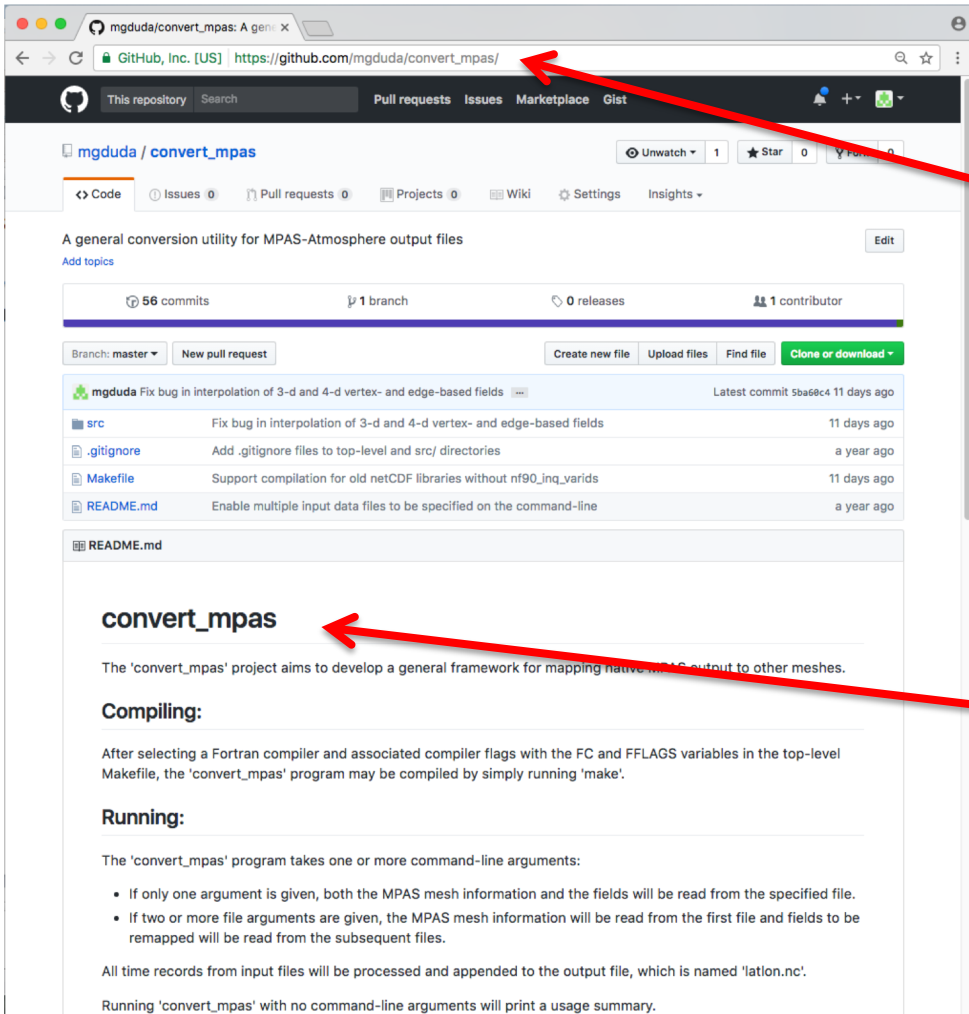


Left: Can you spot Hurricane Matthew in the MPAS 'qv' field seen in ncview?



Recall that MPAS stores 2-d horizontal fields in 1-d arrays; 3-d fields are 2-d arrays with the vertical (structured) dimension innermost, e.g., `theta(nVertLevels,nCells)`.

The 'convert_mpas' tool can quickly interpolate MPAS files to a specified lat-lon grid



Source code can be obtained from https://github.com/mgduda/convert_mpas/

The README.md file summarizes the key details of compiling and running

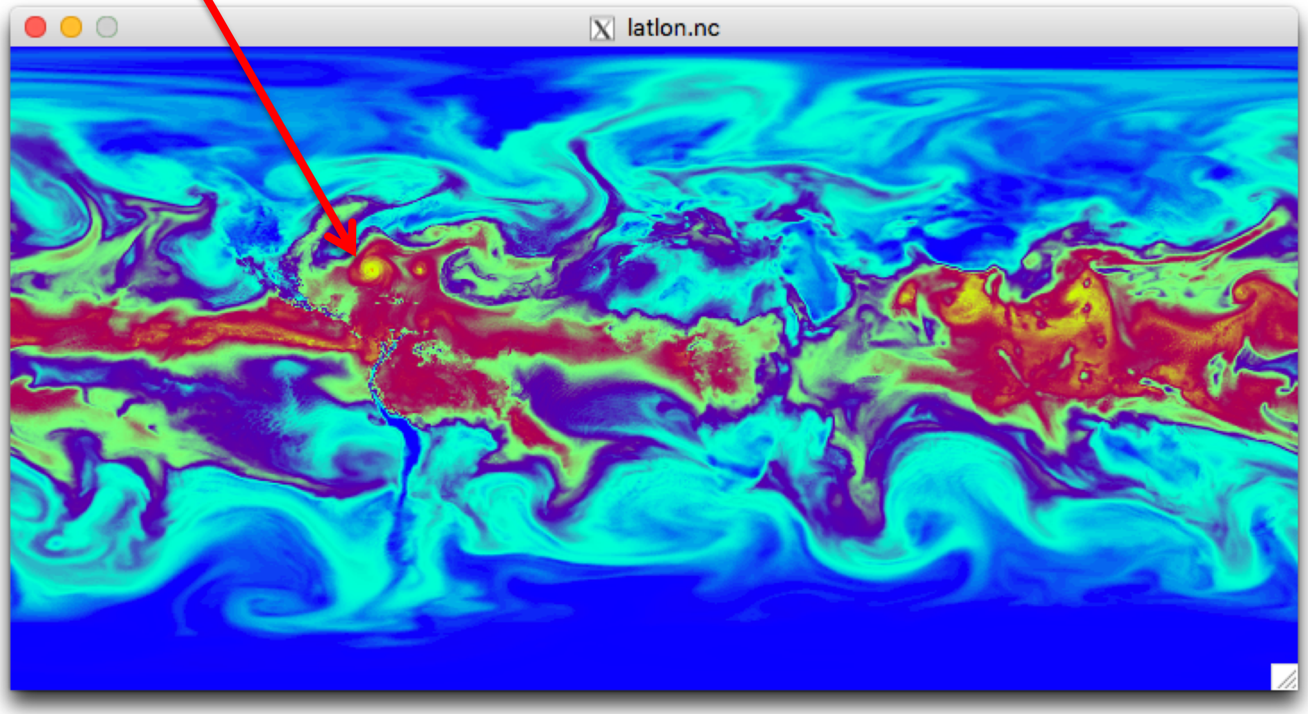
Now that you've got MPAS output, how can you do something useful with it?

Basic usage of 'convert_mpas':

- If just one argument is given, it specifies an MPAS file that has mesh information as well as fields to be interpolated
  - Ex: `convert_mpas x1.40962.init.nc`
- If more than one argument is given:
  - First argument is used *only to obtain mesh information*
  - All remaining arguments contain fields to be interpolated
  - Ex: `convert_mpas x1.40962.grid.nc diag*nc`
  - Ex: `convert_mpas history.2017-06-16_00.nc history*nc`
- Output file is always called `latlon.nc`
  - Probably best to remove this file before re-running 'convert_mpas'
- Default output grid is 0.5-degree lat-lon grid

Now that you've got MPAS output, how can you do something useful with it?

Now we can see Hurricane Matthew in our MPAS output

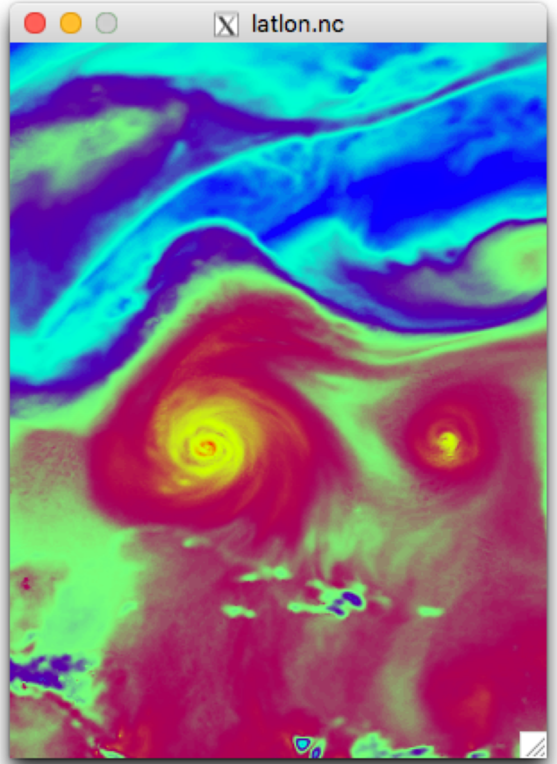


How can we interpolate to just the region of interest and at higher resolution?

Now that you've got MPAS output, how can you do something useful with it?

A text file named `target_domain` in your working directory may be used to specify parameters of the lat-lon grid:

```
startlat=10.0
endlat=50.0
startlon=-90.0
endlon=-60
nlat=400
nlon=300
```



A text file named `include_fields` in your working directory may also be used to list the fields that should be interpolated

MPAS
Model for Prediction Across Scales

Now that you've got MPAS output, how can you do something useful with it?

To plot fields directly from the native MPAS mesh, try NCL, python, Matlab, etc.

MPAS

https://mpas-dev.github.io

MPAS
Model for Prediction Across Scales

MPAS Atmosphere Public Releases

MPAS Home

Overview
- MPAS-Atmosphere
- MPAS-Albany Land Ice
- MPAS-Ocean
- MPAS-Seaice
- Data Assimilation
- Publications
- Presentations

Download
- MPAS-Atmosphere download
- MPAS-Albany Land Ice download
- MPAS-Ocean download
- MPAS-Seaice download

Resources
- License Information
- Wiki
- Bug Tracker
- Mailing Lists
- MPAS Developers Guide
- MPAS Mesh Specification Document

MPAS Atmosphere 7.0 was released on 8 June 2019.

***As of September 2018**, official support for MPAS-Atmosphere has migrated from the Google Groups forum to a web forum hosted by NCAR's Mesoscale and Microscale Meteorology. Users are encouraged to post any questions related to building and running MPAS-Atmosphere to the appropriate sub-topic in the MPAS-Atmosphere forum at http://forum.mmm.ucar.edu/phpBB3/. Posting to the forum requires the creation of an account, but no account is needed to browse the forum.*

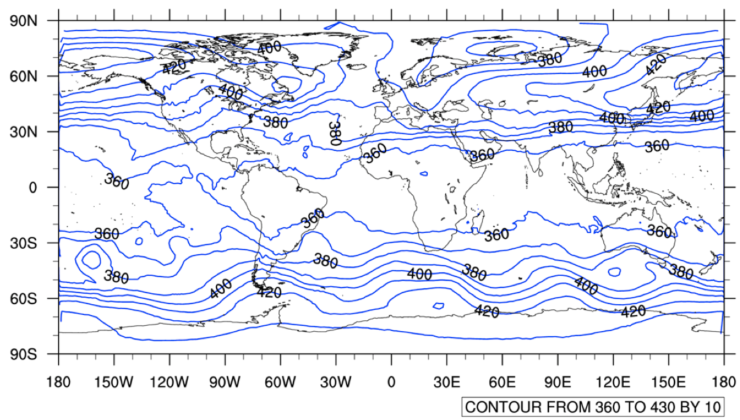
A variable resolution MPAS Voronoi mesh

- MPAS Atmosphere 7.0 release notes
- MPAS source code download
- MPAS-Atmosphere Users' Guide
- MPAS-Atmosphere tutorial presentations
- MPAS-Atmosphere meshes
- Configurations for idealized test cases
- Sample input files for real-data simulations
- Visualization and analysis tools

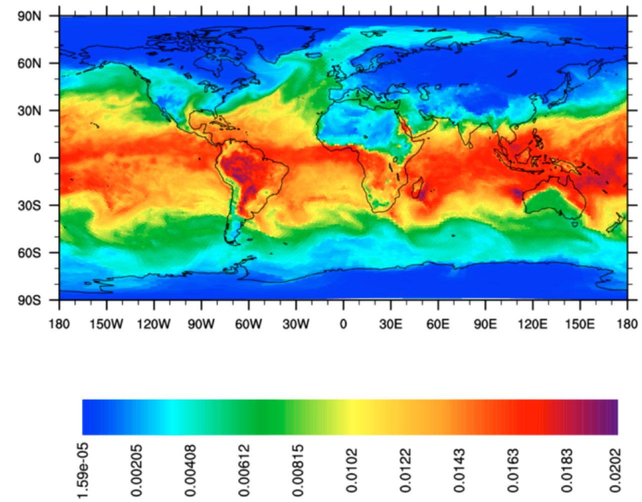
The MPAS-Atmosphere download page contains a collection of example NCL scripts

MPAS
Model for Prediction Across Scales

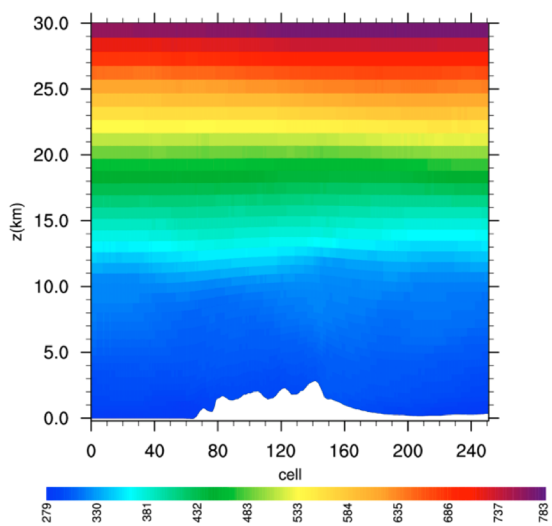
Example NCL scripts from the MPAS-Atmosphere downloads page



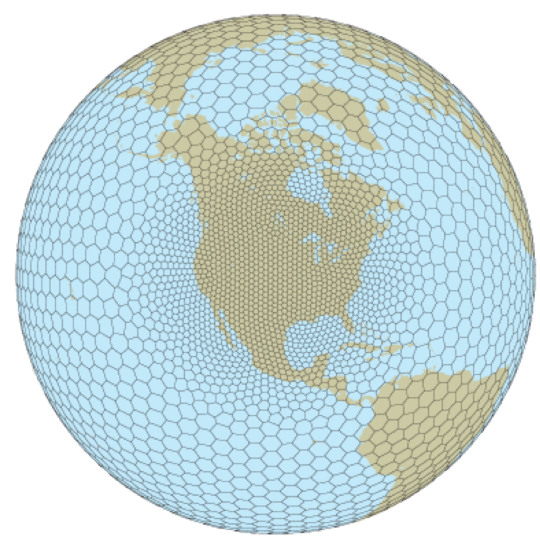
Contours – simple or color-filled



Individual grid cells as a color-filled polygons



Vertical cross-sections with specified endpoints



Voronoi mesh against a map background

Useful links

- The MPAS homepage:

https://mpas-dev.github.io

- The MPAS GitHub repository

https://github.com/MPAS-Dev/MPAS-Model

- The full MPAS v7.0 tutorial

http://www2.mmm.ucar.edu/projects/mpas/tutorial/

- The MPAS-Atmosphere Help forum:

http://forum.mmm.ucar.edu/

Good luck in your future use of MPAS!

Extra slides

Tutorial Outline

Introduction

The MPAS mesh representation

Obtaining, compiling, and running MPAS

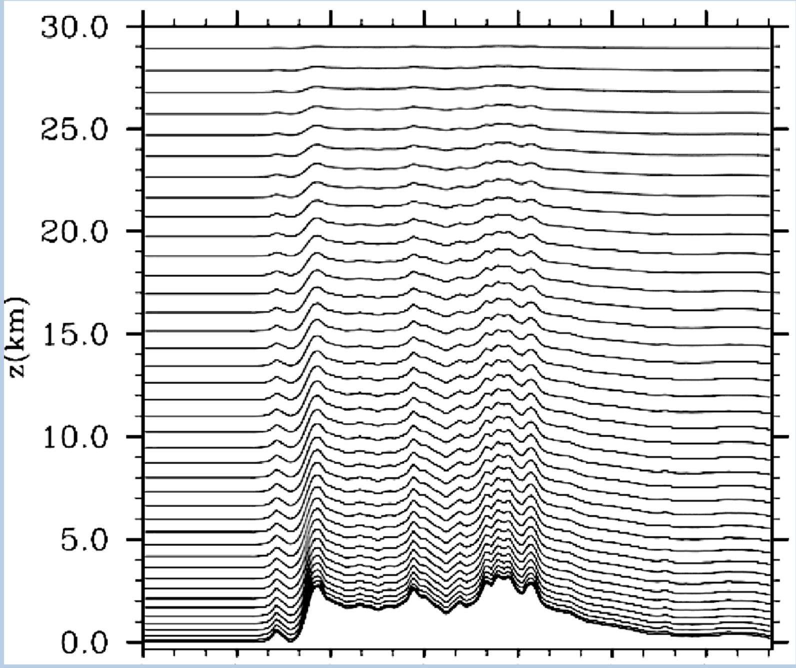
Physics and dynamics configuration

Post-processing and visualization

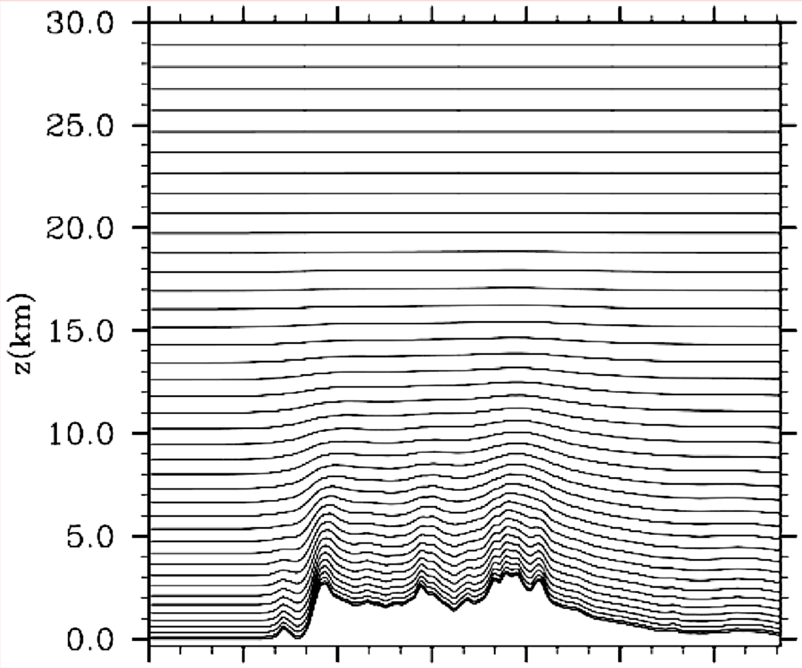
MPAS
Model for Prediction Across Scales

Why MPAS?

Significant differences between WRF and MPAS



WRF
Pressure-based terrain-following sigma vertical coordinate



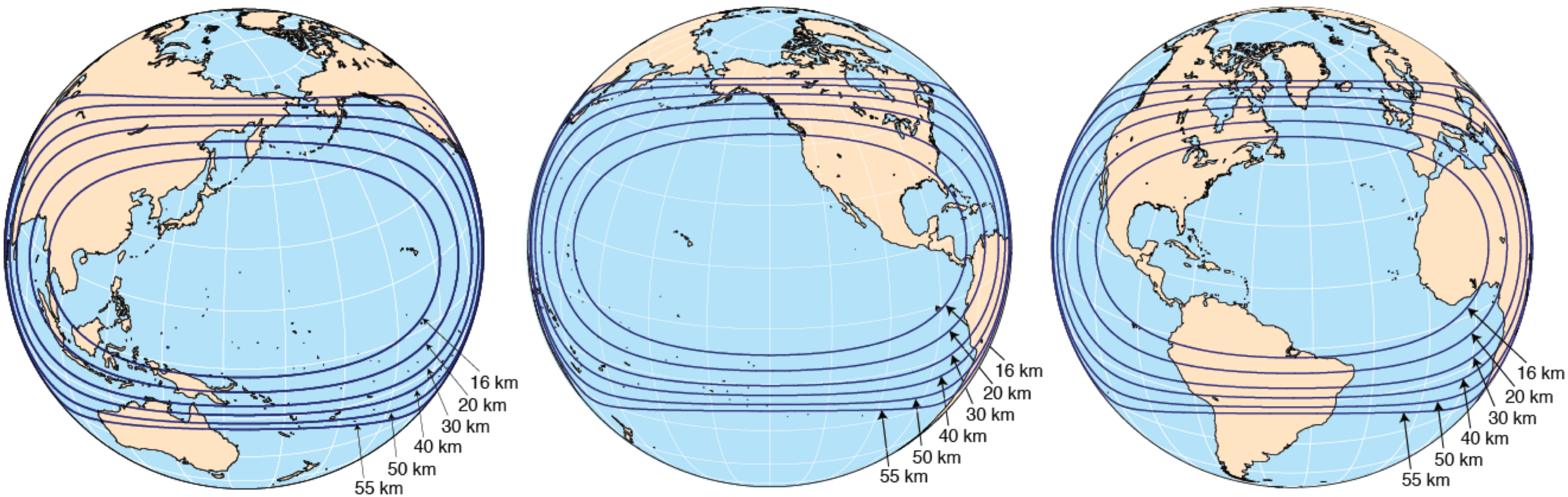
MPAS
Height-based hybrid smoothed terrain-following vertical coordinate

- Improved numerical accuracy

MPAS
Model for Prediction Across Scales

Some Forecast Experiment Meshes

NCAR real-time TC forecast Experiment
November 2014 - 2017
Daily 10-day MPAS forecasts
00 UTC GFS analysis initialization



60-15 km mesh, δx contours
535,554 cells (columns)
35% have < 16 km spacing
(218 pentagons, 206 septagons)

Tutorial Outline

Introduction

The MPAS mesh representation

Obtaining, compiling, and running MPAS

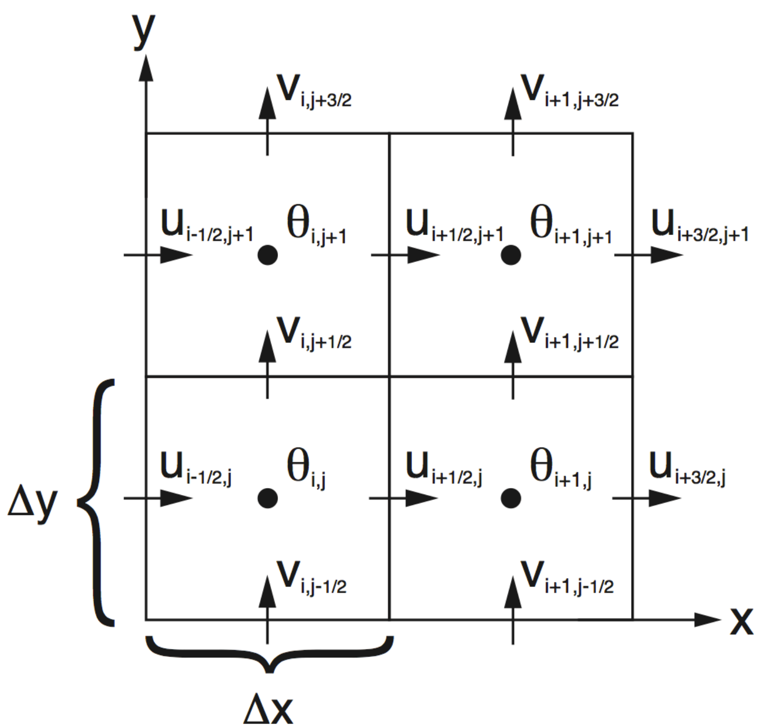
Physics and dynamics configuration

Post-processing and visualization

How do MPAS, *and you,* keep track of this unstructured Voronoi mesh?

One can start to imagine way to identify neighboring cells implicitly based on the index or location of each cell

- In WRF, our neighbors are at (i+1, j), (i-1, j), (i, j+1), (i, j-1)

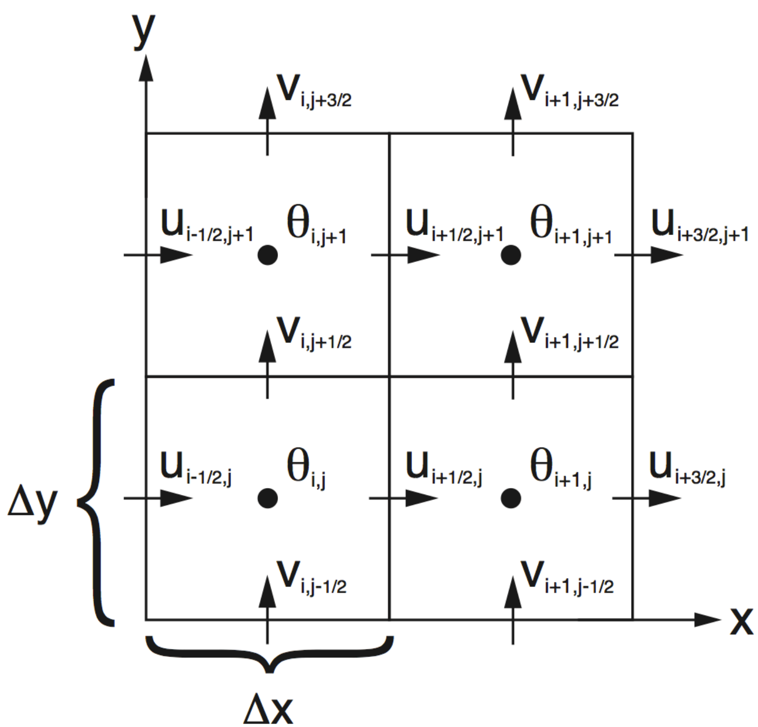


Above: A region from the ARW C-staggered grid, stored in a 2-d array.

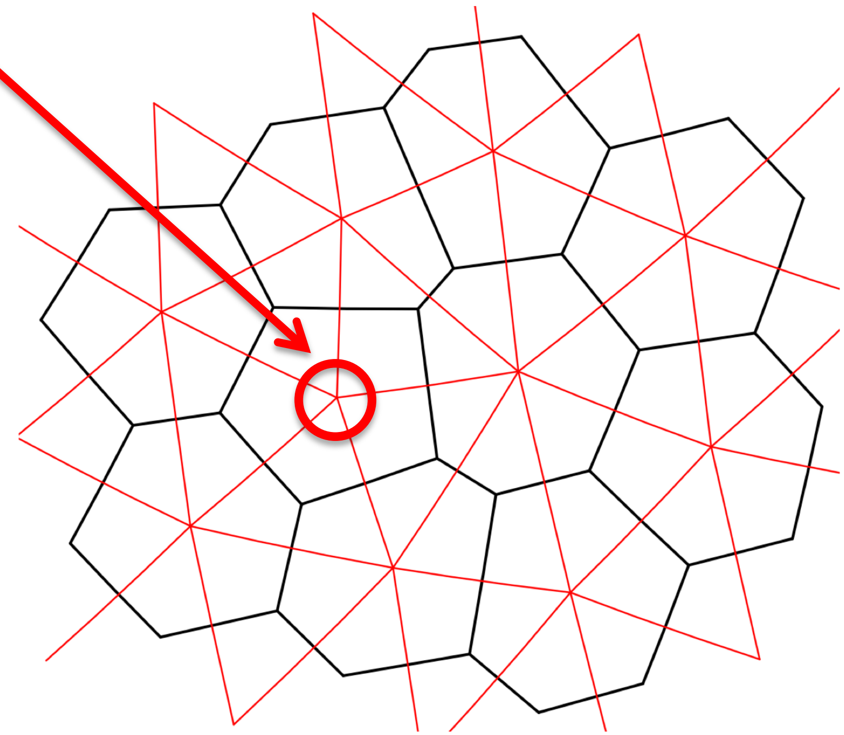
How do MPAS, *and you*, keep track of this unstructured Voronoi mesh?

One can start to imagine way to identify neighboring cells implicitly based on the index or location of each cell

- In WRF, our neighbors are at (i+1, j), (i-1, j), (i, j+1), (i, j-1)
- Who is the "next" cell after this one in any direction?



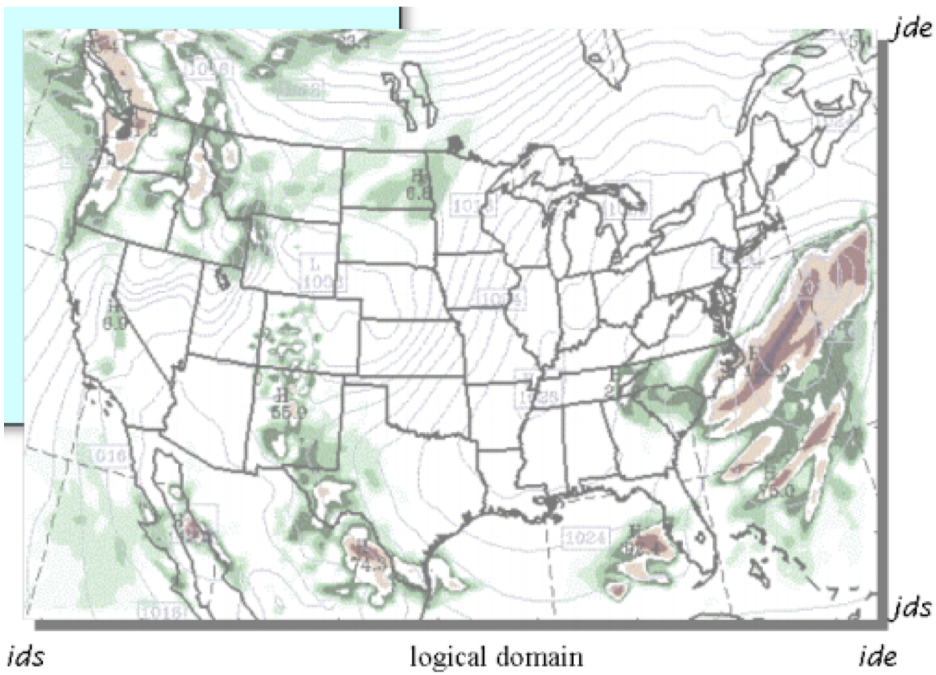
Above: A region from the ARW C-staggered grid, stored in a 2-d array.

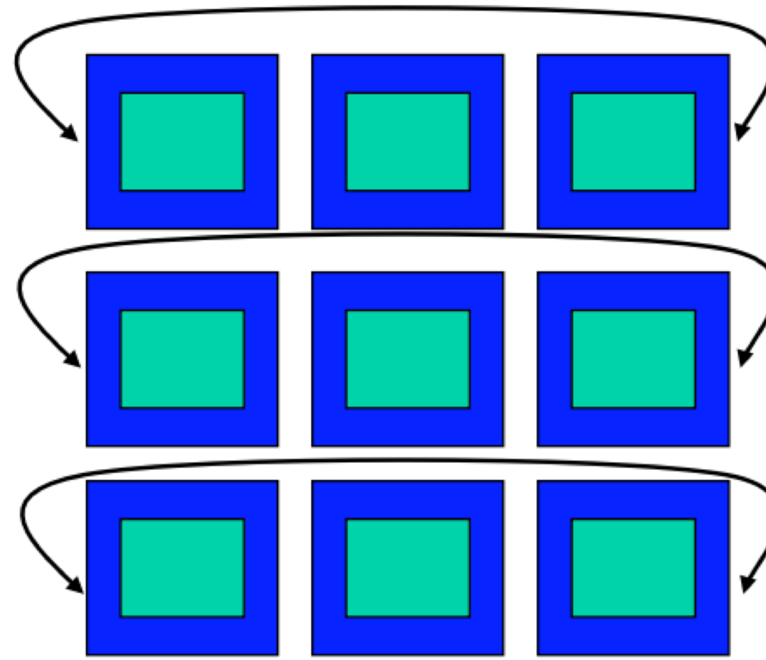


Above: A region from an MPAS mesh showing Voronoi regions (black) and Delaunay triangles (red).

One use of connectivity arrays: horizontal domain decomposition

In WRF, horizontal domain decomposition is simplified by the logically rectangular mesh and can take place at runtime with essentially no cost...



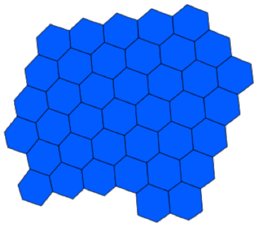


Above: A WRF domain decomposed into nine patches (taken from Dave Gill's WRF architecture tutorial slides).

How can we do the equivalent domain decomposition for an unstructured MPAS mesh?

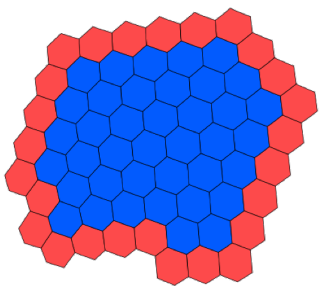
MPAS
Model for Prediction Across Scales

How are meshes partitioned into blocks for parallelization?

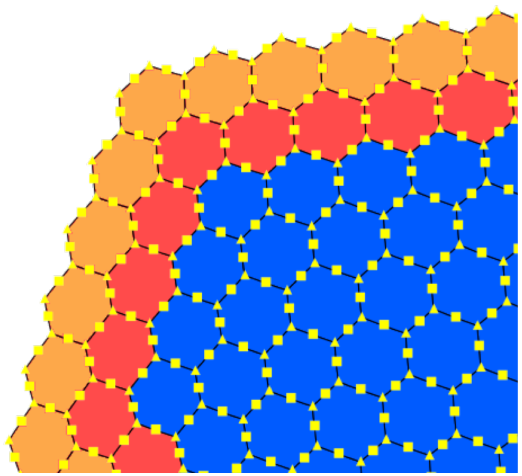


Block of cells owned by a process

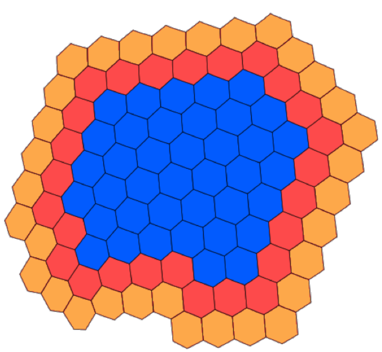
Given an assignment of cells to a process, any number of layers of halo (ghost) cells may be added



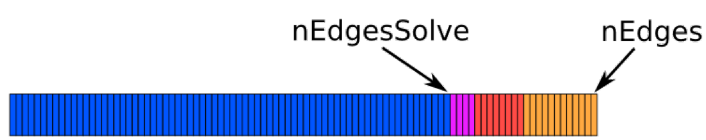
Block plus one layer of halo/ghost cells

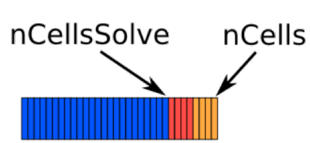


With a complete list of cells stored in a block, adjacent edge and vertex locations can be found; we apply a simple rule to determine ownership of edges and vertices adjacent to real cells in different blocks



Block plus two layers of halo/ghost cells



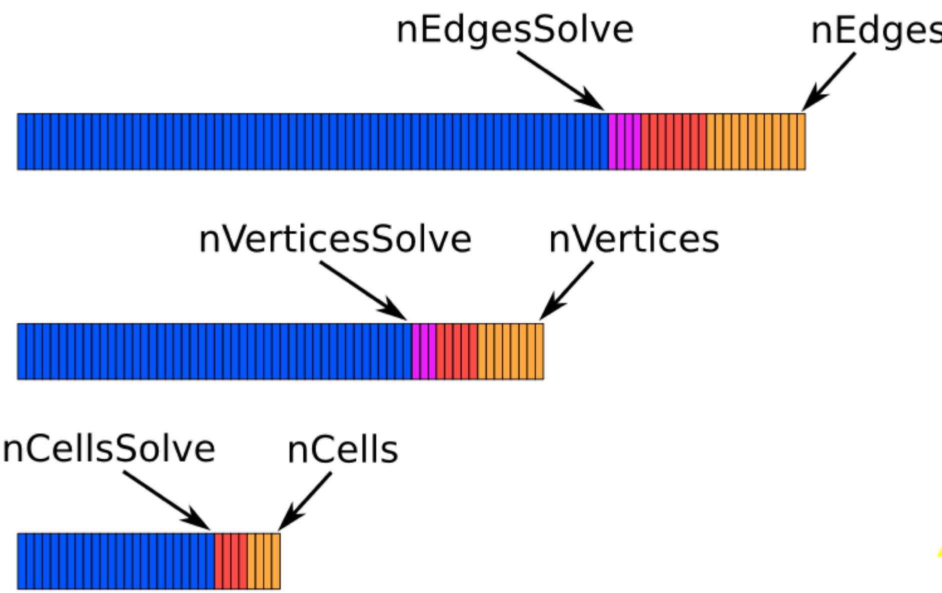


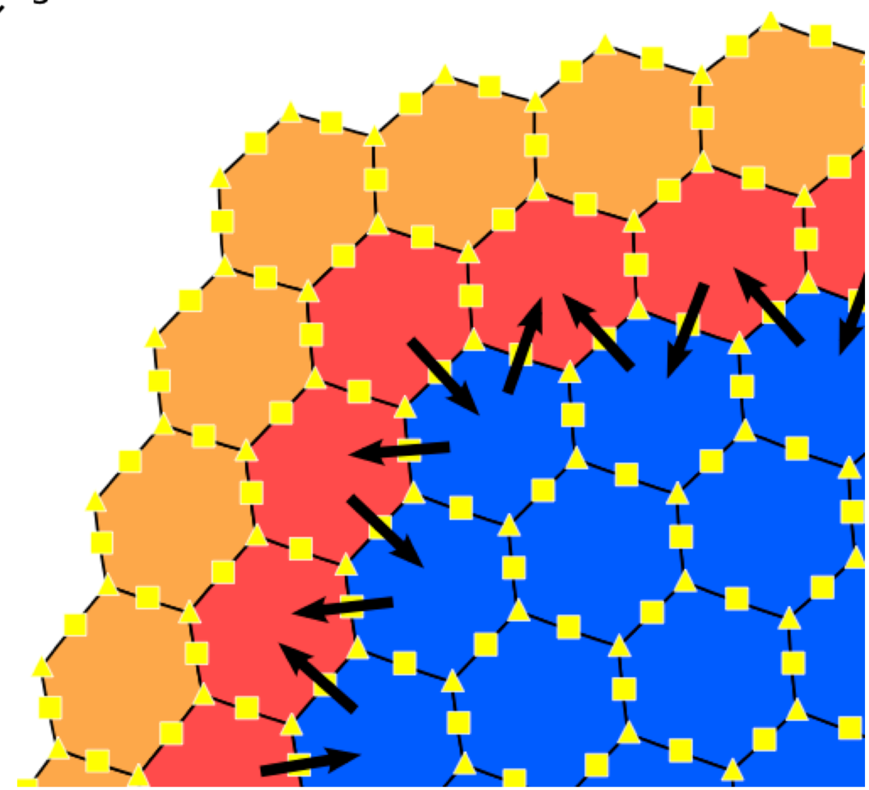
Cells, edges, and vertices are stored in a 1-d array, with halo cells at the end of the array

How are meshes partitioned into blocks for parallelization?

An edge *E* is an owned edge **iff** cellsOnEdge(1,*E*) is an owned cell

A vertex *V* is an owned vertex **iff** cellsOnVertex(1,*V*) is an owned cell

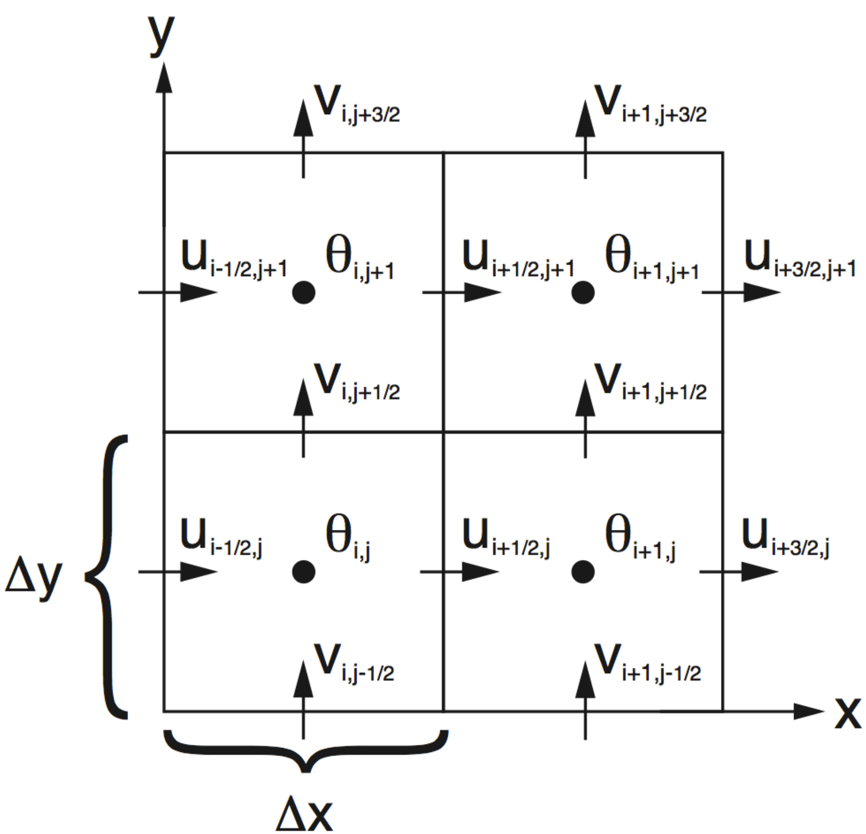


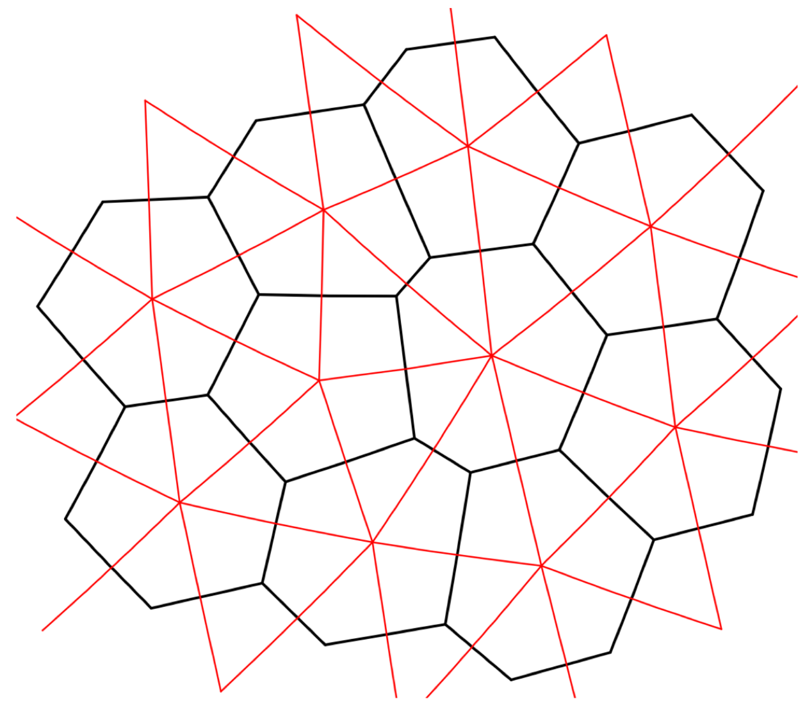


For *n* layers of ghost cells, we have *n+1* layers of ghost edges and ghost vertices.

Horizontal wind vectors: how are these defined?

In a WRF grid, which directions represent positive U and positive V?



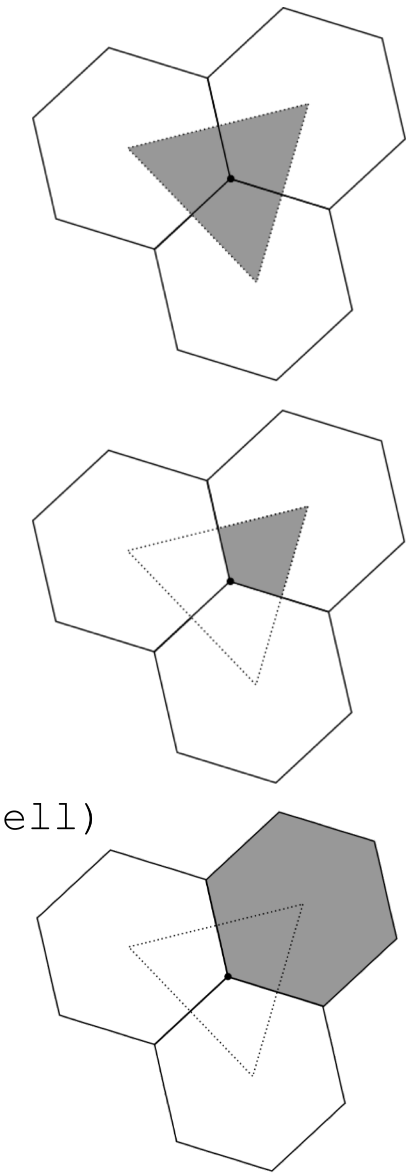


On a rectangular grid, one might say that positive U flows from left to right, and positive V flows from bottom to top when looking down on the xy-plane.

On a CVT mesh, one could introduce a similar definition, but we have only U, not V, so such a definition becomes more complicated...

An example of using mesh fields: Averaging a vertex-based field, `vorticity(nVertLevels,nVertices)`, to cells as `vortcell(nVertLevels,nCells)`:

```
vortcell(:,:) = 0.0
do iVtx = 1, nVertices
do j = 1, vertexDegree
    iCell = cellsOnVertex(j,iVtx)
    vortcell(:,j) = vortcell(:,j) +
          kiteAreasOnVertex(j,iVtx) * vorticity(:,iVtx)
end do
end do

do iCell = 1, nCells
    vortcell(:,iCell) = vortcell(:,iCell) / areaCell(iCell)
end do
```



Tutorial Outline

Introduction

The MPAS mesh representation

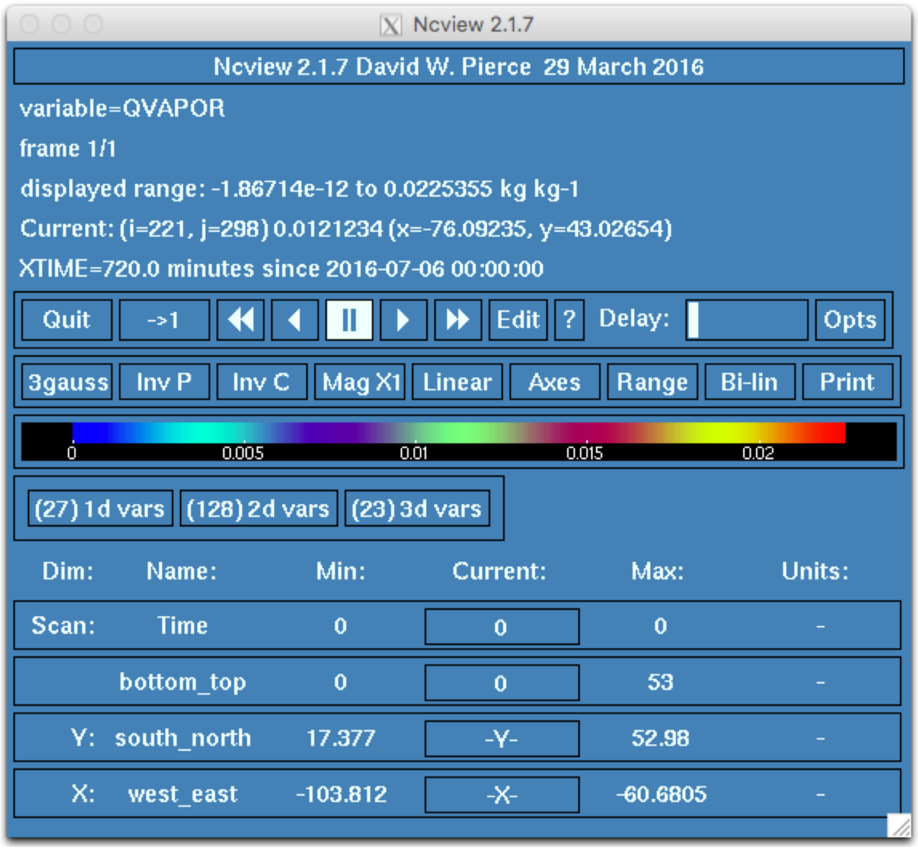
Obtaining, compiling, and running MPAS

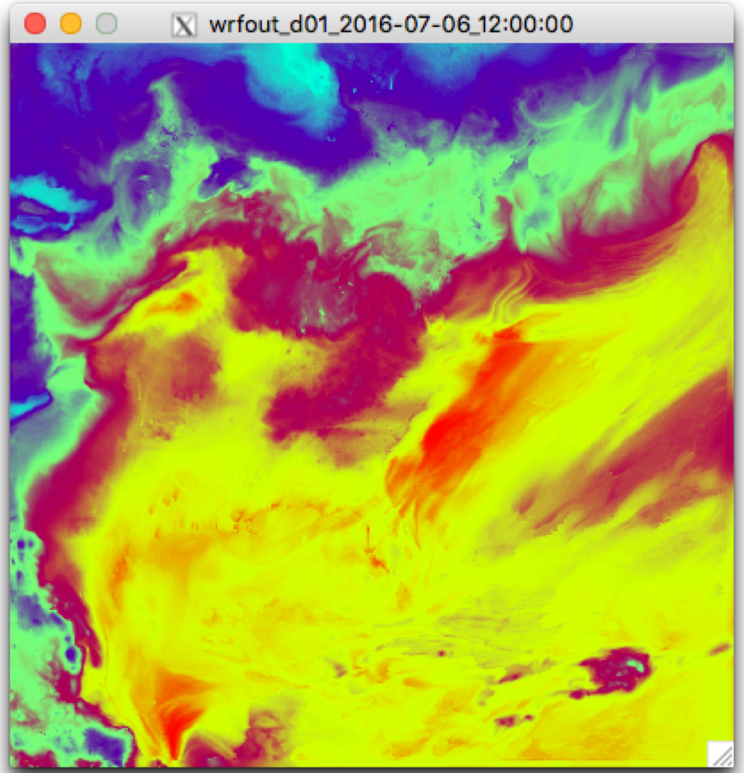
Physics and dynamics configuration

Post-processing and visualization

With WRF output, a quick look at model input/output is simple!

(Try making a mental estimate of the number of times you've run 'ncview' in the past year...)





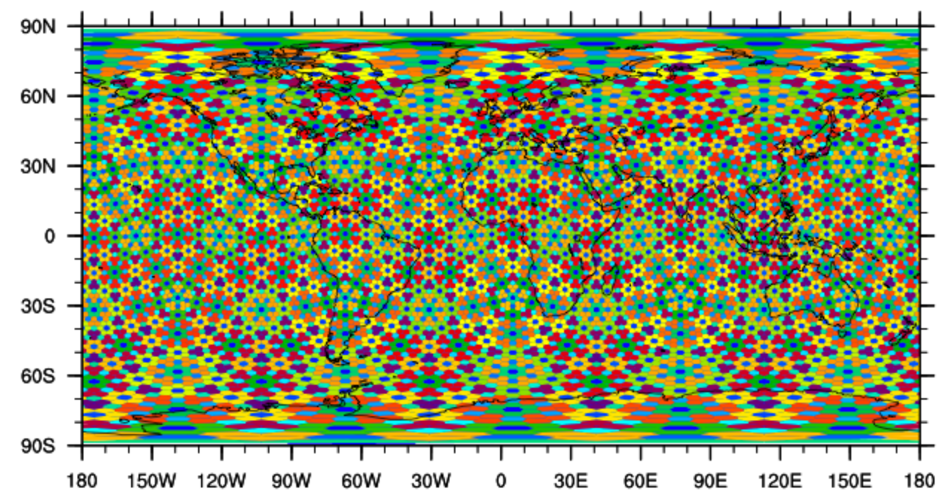
Making use of the MPAS mesh representation to more efficiently work with MPAS output

In WRF, finding the nearest grid cell to a given (lat,lon) location is a constant-time operation:

1. Using the map projection equations for the WRF grid projection, compute the real-valued (x,y) coordinates of the (lat,lon) location
2. Round the real-valued coordinates to the nearest integer

However, in MPAS, *there is no projection*, and the horizontal cells may be indexed in any order.

- We could just compute the distance from (lat,lon) to every cell center in the mesh and choose the nearest cell, or we could do something more efficient...



Above: Cells in the x1.10242 mesh colored according to their global index

Making use of the MPAS mesh representation to more efficiently work with MPAS output

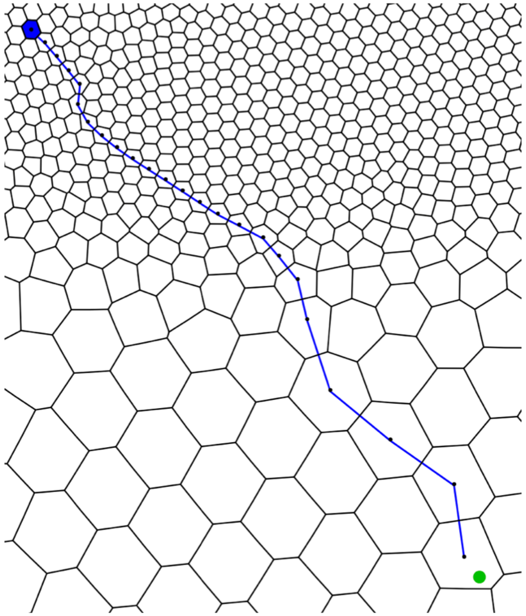
One solution would be to use search trees – perhaps a *kd*-tree – to store the cells in a mesh

- *O*(n log n) setup cost; each search takes O(log n) time, for a mesh with *n* cells

Alternatively, we can make use of the grid connectivity arrays nEdgesOnCell and cellsOnCell to navigate a path of monotonically decreasing distance to the (lat,lon) location

- No setup cost, at most $O(n^{1/2})$ cost* per search
- For repeated searches of "nearby" locations, almost constant cost!

$C_{nearest}$ = any starting cell
C_{test} = NULL
do while ($C_{nearest} \neq C_{test}$)
    $C_{test} = C_{nearest}$
    d = distance from C_{test} to (lat,lon)
    for i = 1 to nEdgesOnCell(C_{test})
        k = cellsOnCell(i, C_{test})
        d' = distance from k to (lat,lon)
        if (d' < d)
            d = d'; $C_{nearest}$ = k



Above: Path taken from starting cell (blue) to target location (green circle).

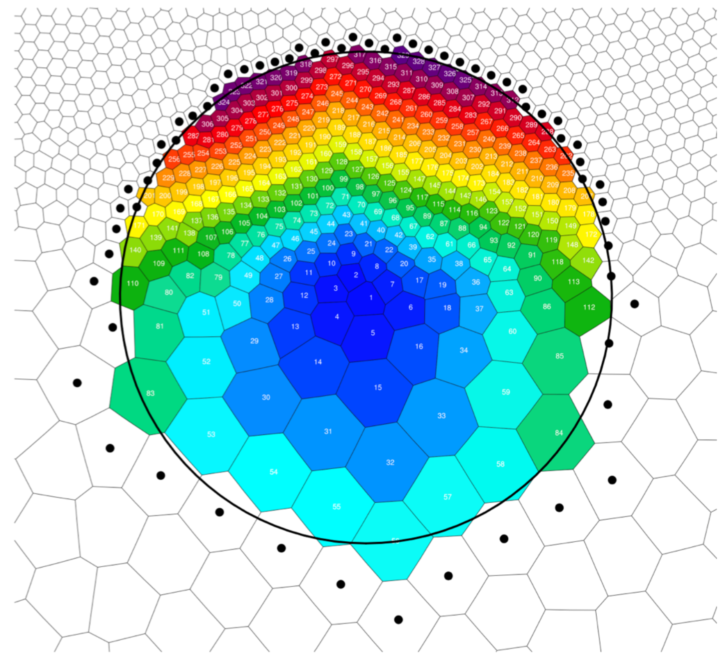
*At least, intuitively...

Making use of the MPAS mesh representation to more efficiently work with MPAS output

Similar to the problem of nearest grid cell, to scan all cells within a specified radius of a given (lat,lon) location, we could check all cells in the mesh...

... or we could make use of the connectivity arrays.

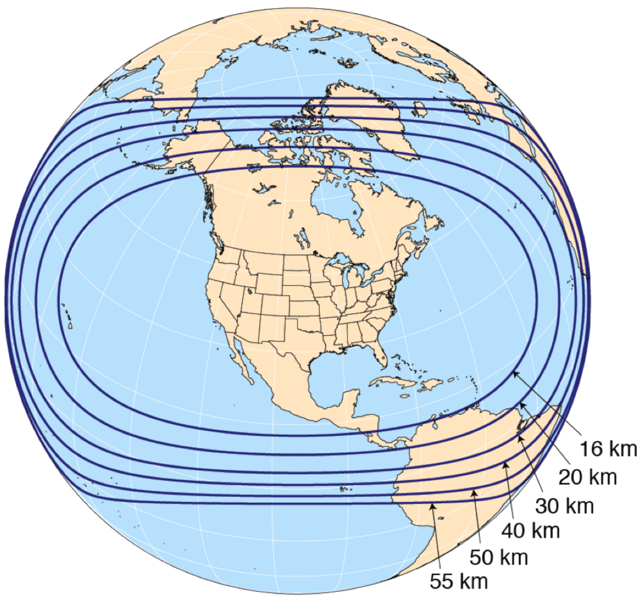
```
C = origin of the search
mark C as visited
insert C into the queue
do while (queue not empty)
    C = next cell from the queue

    C within search radius, so process C

    for i = 1 to nEdgesOnCell(C)
        k = cellsOnCell(i,C)
        if ( k not visited )
            mark k as visited
            if (k within search radius)
                insert k into the queue
```



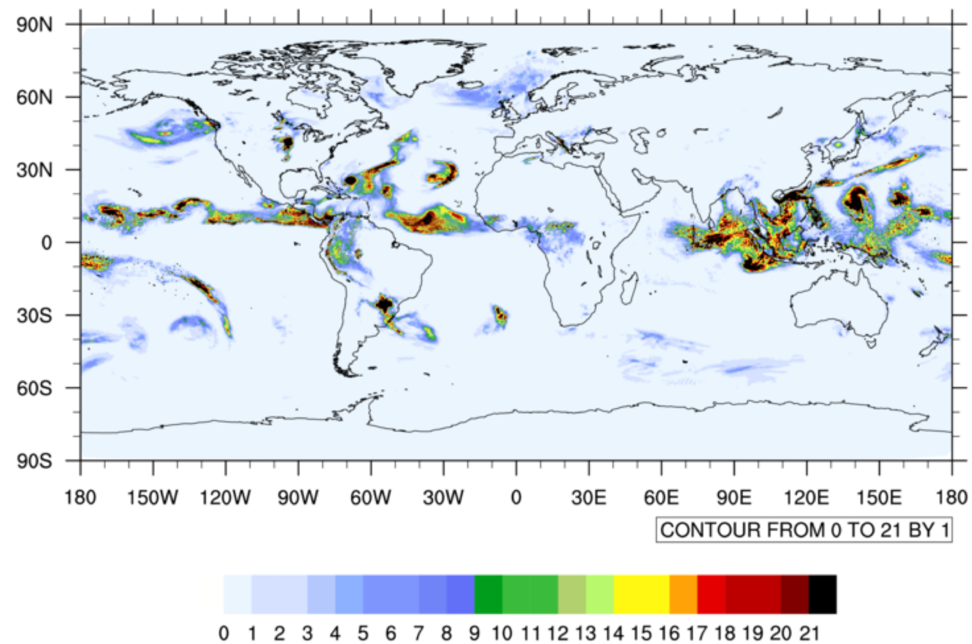
Above: Cells shaded according to the order in which they were visited by a 750-km radius search; dots indicate cells that were considered but found to be at a radius >750 km.

Important considerations for post-processing on variable-resolution meshes

Consider the computation of the daily precipitation rate on a variable-resolution MPAS mesh:



Above: An MPAS 60-15 km variable-resolution mesh with refinement over North America



Above: The accumulated total precipitation between 2016-10-14 00 UTC and 2016-10-15 00 UTC on from MPAS with the 'mesoscale_reference' physics suite.

How much can the way in which we compute the daily precipitation rate affect our results?

An obvious conclusion, but one that's important...

Taking a simple average of the precipitation rate in all cells gives **3.43 mm/day**

```
f1 = addfile("diag.2016-10-14_00.00.00.nc","r")
f2 = addfile("diag.2016-10-15_00.00.00.nc","r")
fld = (f2->rainc(0,:) + f2->rainnc(0,:)) –
      (f1->rainc(0,:) + f1->rainnc(0,:))
fg = addfile("init.nc","r")
print(sum(fld * fg->areaCell(:)) / sum(fg->areaCell(:)))
```

Weighting the precipitation rate by cell area gives **2.93 mm/day**

In a "typical" WRF simulation with map scale factors between 0.9 and 1.1, the cell area ratio between the largest cell and the smallest is about **1.49**.

In an MPAS simulation with a variable-resolution mesh with a refinement factor of four (e.g., 60-15 km grid distance), the cell area ratio between the largest and smallest cells in the mesh is **16**!

Tutorial Outline

Introduction

The MPAS mesh representation

Obtaining, compiling, and running MPAS

Physics and dynamics configuration

Post-processing and visualization

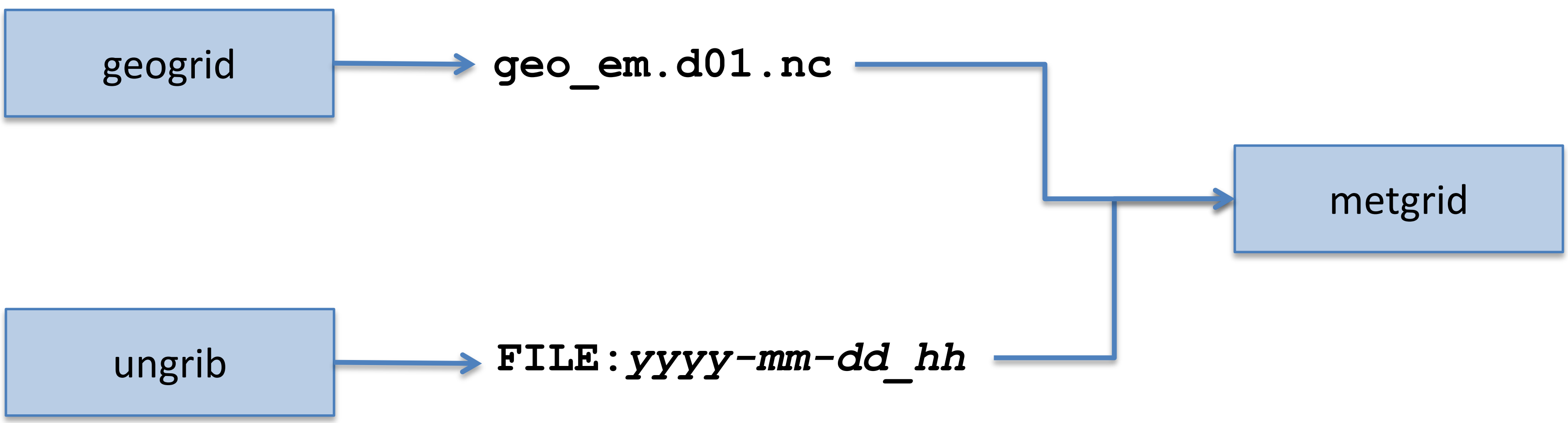
Driving WRF from MPAS output

How does one use MPAS output as ICs and LBCs for WRF?

Basic requirements:

1. WPS v3.9
2. MPAS v5.x
   - Some older MPAS versions may also work, but are untested

What is normally done to initialize WRF with GRIB data:

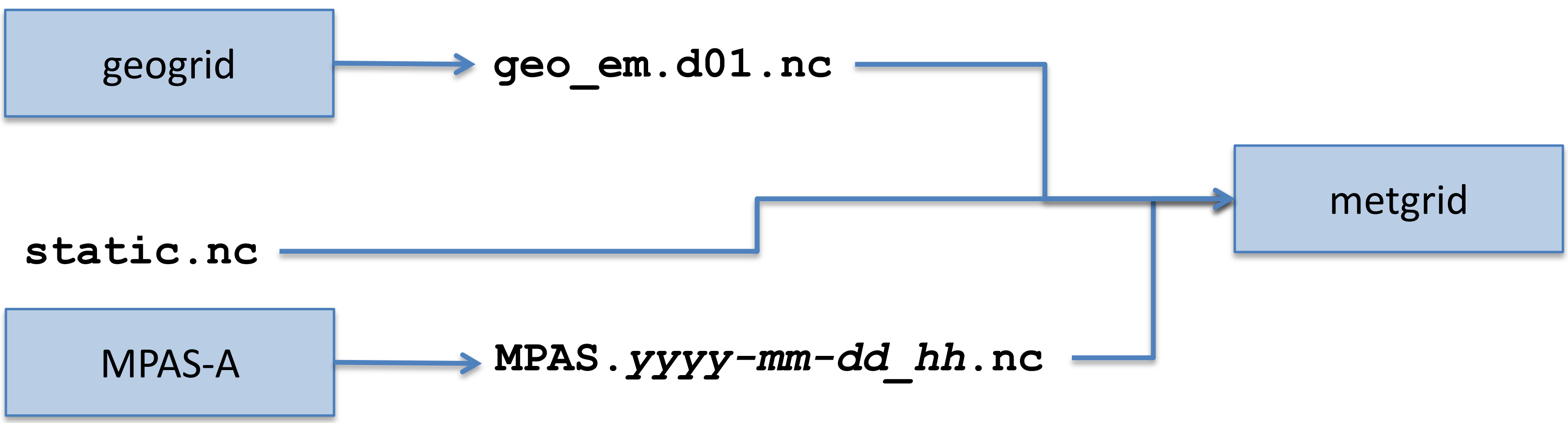


How does one use MPAS output as ICs and LBCs for WRF?

Basic requirements:

1. WPS v3.9
2. MPAS v5.x
   - Some older MPAS versions may also work, but are untested

What is done to use MPAS output to initialize WRF:



The WPS v3.9 metgrid program can directly read and interpolate native MPAS netCDF files!

When running MPAS, write a new output stream to be used by the WPS metgrid program:

```
<stream name="wrf_ic_bc"
        type="output"
        filename_template="MPAS.$Y-$M-$D_$h.nc"
        output_interval="3:00:00" >

        <var name="xtime"/>
        <var_array name="scalars"/>
        <var name="pressure"/>
        <var name="zgrid"/>
        <var name="theta"/>
        <var name="uReconstructZonal"/>
        <var name="uReconstructMeridional"/>
        <var name="u10"/>
        <var name="v10"/>
        <var name="q2"/>
        <var name="t2m"/>
        <var name="skintemp"/>
        <var name="surface_pressure"/>
        <var name="mslp"/>
        <var name="tslb"/>
        <var name="smois"/>

</stream>
```

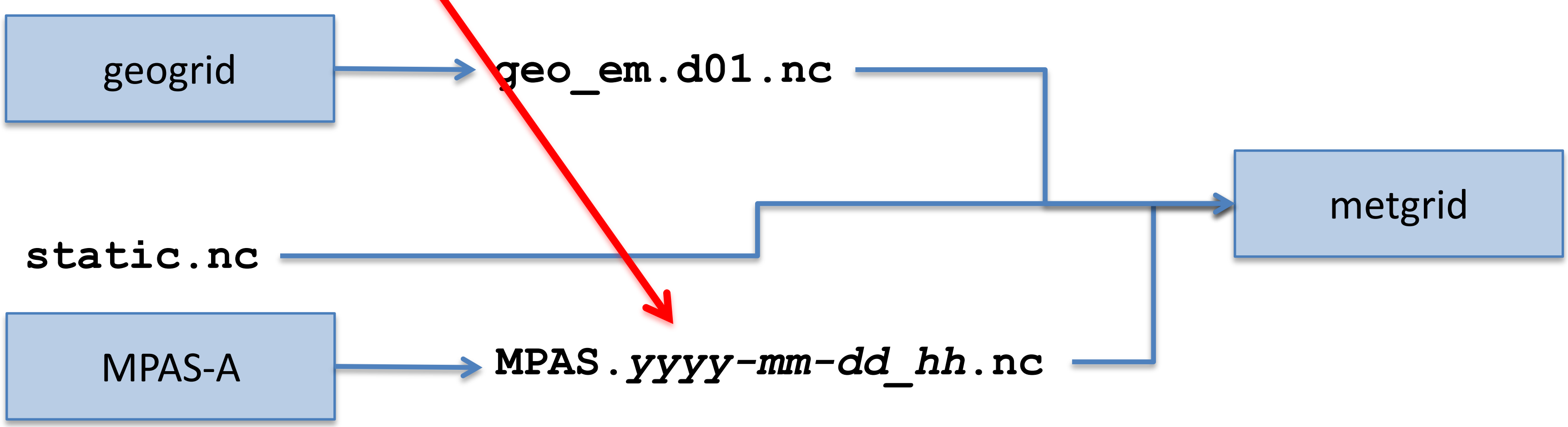
The filename template must end with “`$Y-$M-$D_$h.nc`”

The output interval of this stream is typically the LBC update interval to be used in WRF

How does one use MPAS output as ICs and LBCs for WRF?

Running MPAS after adding the “wrf_ic_bc” stream to the `streams.atmosphere` results in files like:

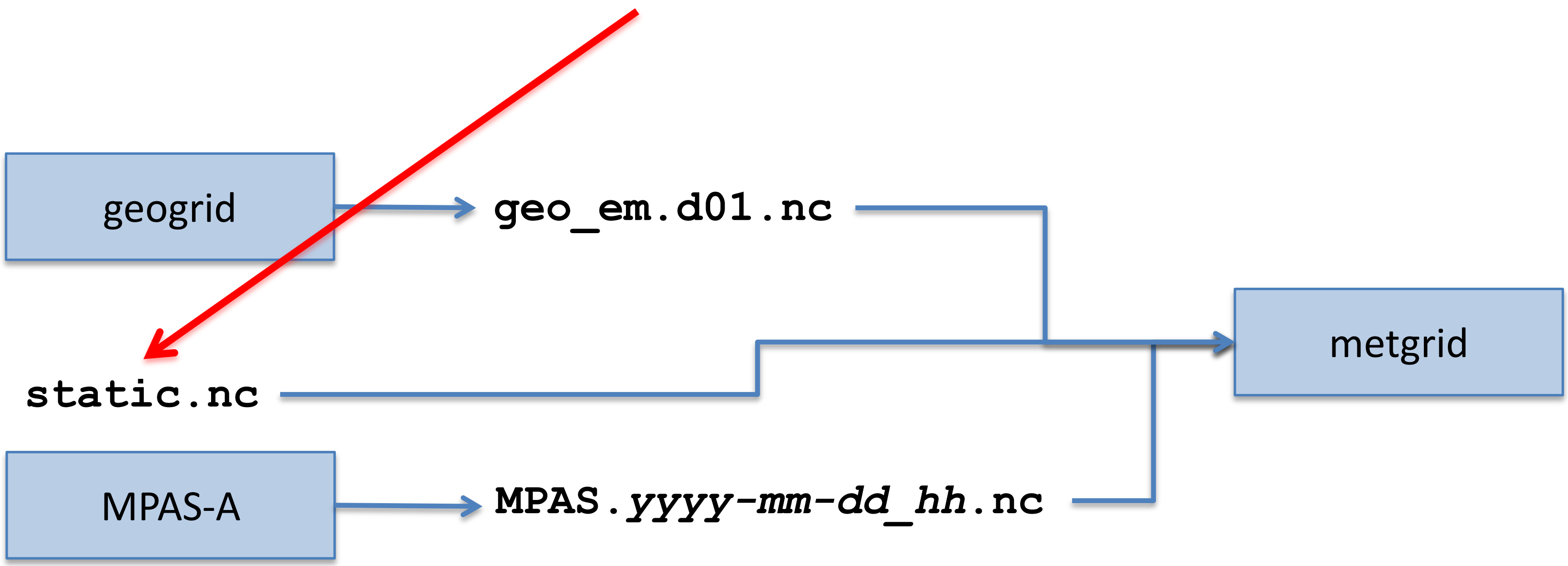
```
MPAS.2017-06-16_00.nc
MPAS.2017-06-16_03.nc
MPAS.2017-06-16_06.nc
MPAS.2017-06-16_09.nc
...
```



How does one use MPAS output as ICs and LBCs for WRF?

The `MPAS.yyyy-mm-dd_hh.nc` files generated by the MPAS simulation have atmosphere and land-surface fields, but they don't contain any information about the MPAS horizontal mesh, terrain height, or land-sea mask

Metgrid must also be provided with the matching MPAS "static" file



How does one use MPAS output as ICs and LBCs for WRF?

When running metgrid, MPAS netCDF files are handled in much the same way in the `&metgrid` namelist variables `constants_name` and `fg_name`

- The key difference is that MPAS netCDF filenames are prefixed with '`mpas:`'

```
&metgrid
  constants_name = 'mpas:static.nc'
  fg_name = 'mpas:MPAS'
/
```

This is the prefix of the MPAS.yyyy-mm-dd_hh.nc output files; any prefix can be chosen when running MPAS...

This is the full name of the MPAS mesh file; as of WPS v3.9, other MPAS files (e.g., the `init.nc` file) should not be used...

How are MPAS field names matched up to names in the “met_em” files?

A new keyword, mpas_name, appears in the entries of the METGRID.TBL.ARW file:

```
========================================
name=UU
mpas_name=uReconstructZonal
        mandatory=yes
        interp_option=sixteen_pt+four_pt+average_4pt
        is_u_field=yes
        output_stagger=U
        fill_missing=0.
        fill_lev=200100:const(-1.E30)
========================================
```

The interpolation options specified for a field in the METGRID.TBL.ARW file are ignored when interpolating from MPAS output

Optimizing for storage space:

- Because the ‘zgrid’ field from MPAS doesn’t change in time, we can place this field in its own netCDF file and use it metgrid with the `constants_name` namelist option

```
&metgrid
  constants_name = ‘mpas:static.nc’, ‘mpas:zgrid.nc’
  fg_name = ‘mpas:MPAS’,
/
```

A few lines of python can extract the zgrid field from an MPAS initial conditions file for this purpose:

```
from netCDF4 import Dataset

fin = Dataset('init.nc')
fout = Dataset('zgrid.nc','w',format='NETCDF3_64BIT')

nCells = fin.dimensions['nCells'].size
nVertLevelsP1 = fin.dimensions['nVertLevelsP1'].size

fout.createDimension(dimname='Time',size=None)
fout.createDimension(dimname='nCells',size=nCells)
fout.createDimension(dimname='nVertLevelsP1',size=nVertLevels
P1)
fout.createVariable(varname='zgrid',datatype='f',dimensions=(
'nCells','nVertLevelsP1'))
fout.variables['zgrid'][:] = fin.variables['zgrid'][:]
fout.close()
fin.close()
```

Other points worth noting:

- Processing multiple domains with the WPS works the same with MPAS output as with GRIB/intermediate data
- Metgrid can combine fields from MPAS with fields from intermediate files
    - Though this is probably only useful for obtaining SST, sea-ice, or land-surface fields from another GRIB source, since 3-d atmospheric fields would need to have the same vertical levels in both the MPAS and intermediate files

```
&metgrid
  constants_name = ‘mpas:static.nc’
  fg_name = ‘mpas:MPAS’, ‘ERAI_SOIL’
/
```

How does one use MPAS output as ICs and LBCs for WRF?

Other points worth noting (continued):

- When starting a WRF simulation from the beginning of an MPAS simulation, the previously-outlined approach may not be optimal
  - ICs are first interpolated to MPAS grid, then a second time from MPAS grid to WRF grid
  - *Using da_update_bc.exe may help?*