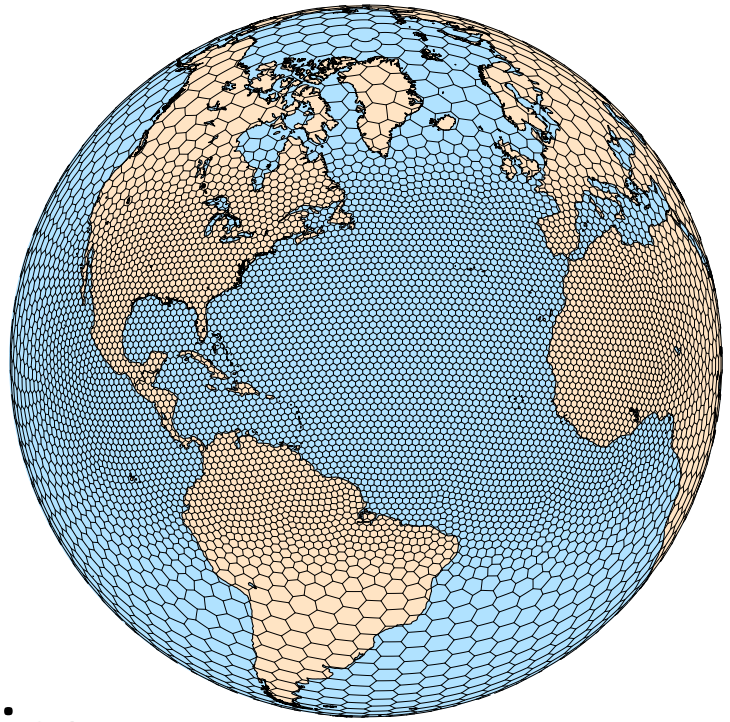
MPAS

Model for Prediction Across Scales



- Overview
- *Meshes*
- Atmospheric solver, physics
- Compiling and running MPAS
- Summary
- Practical session

Los Alamos NATIONAL LABORATORY EST. 1943

U.S. DEPARTMENT OF ENERGY | Office of Science

NSF

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

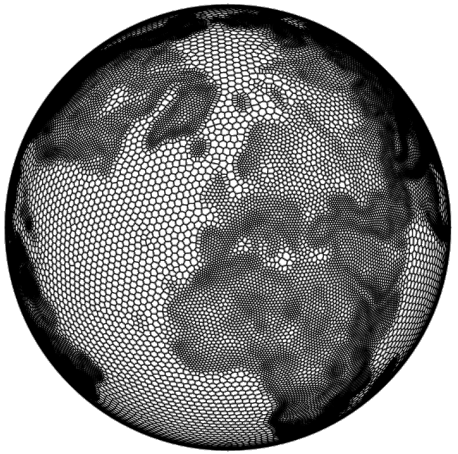
OUTLINE

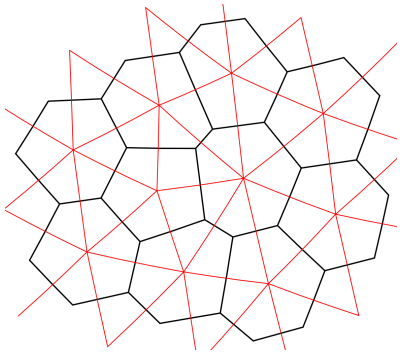
1. Introduction to MPAS meshes
   - What are centroidal Voronoi Tessellations, and how do we represent them in MPAS?

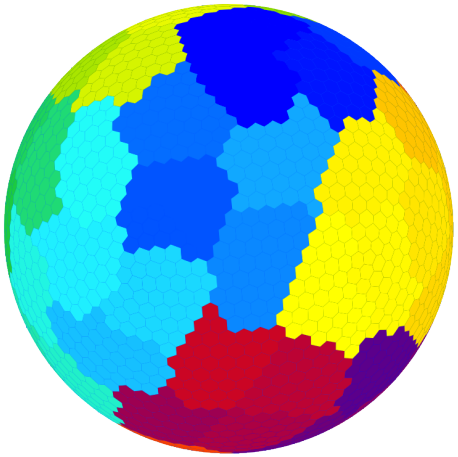
2. Considerations for handling unstructured meshes
   - In an unstructured mesh, how do we define the “positive” flow direction?

3. Mesh partitioning
   - How are meshes divided among processors?

4. Mesh generation
   - How do we achieve variable-resolution in an MPAS mesh?







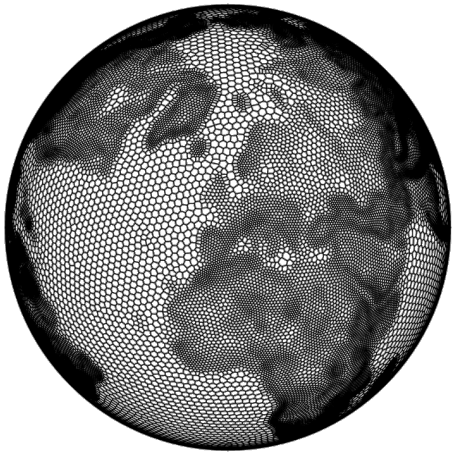
OUTLINE

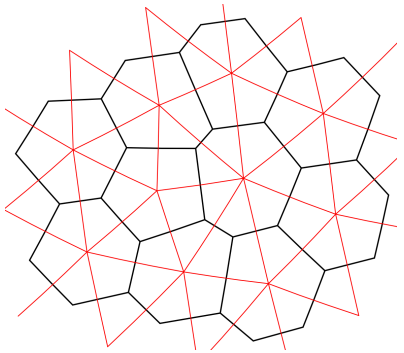
1. Introduction to MPAS meshes
   - What are centroidal Voronoi Tessellations, and how do we represent them in MPAS?

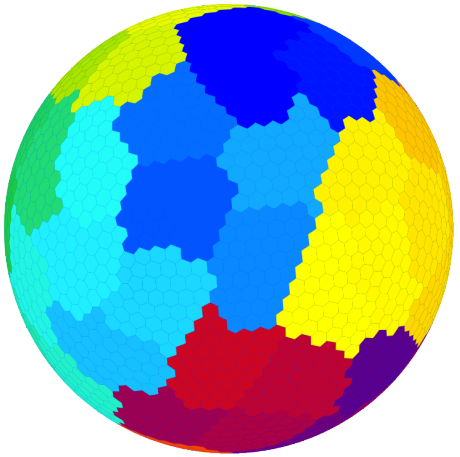
2. Considerations for handling unstructured meshes
   - In an unstructured mesh, how do we define the “positive” flow direction?

3. Mesh partitioning
   - How are meshes divided among processors?

4. Mesh generation
   - How do we achieve variable-resolution in an MPAS mesh?







A defining feature of MPAS models is their use of **centroidal Voronoi tessellations** (CVTs) with a C-grid staggering

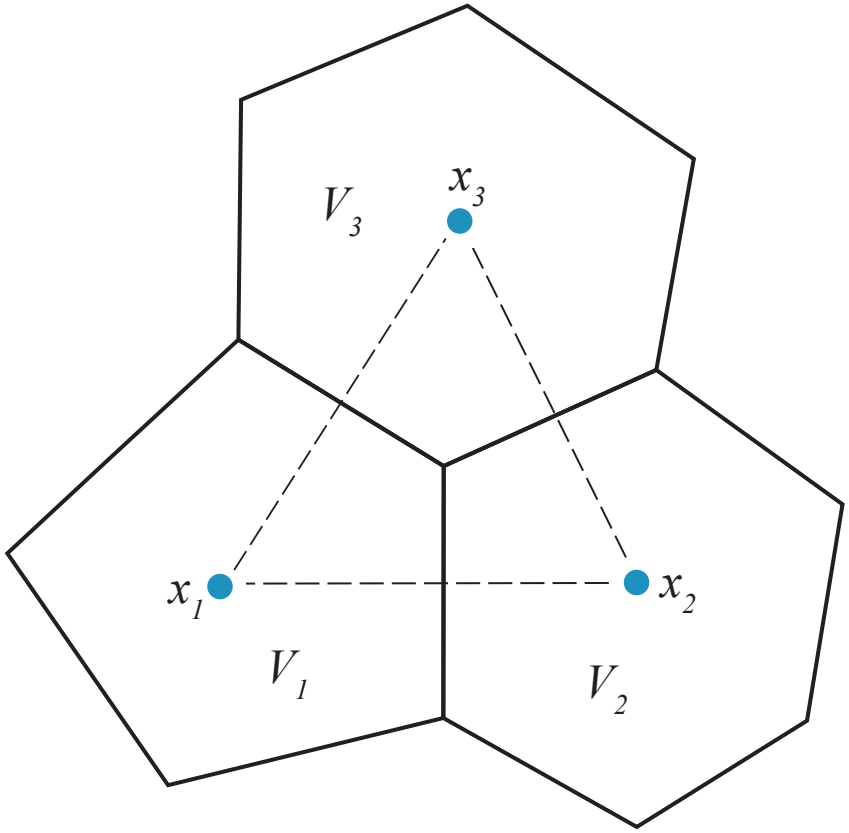
– When constrained to lie on the surface of a sphere, we often call them spherical centroidal Voronoi tessellations (SCVTs)

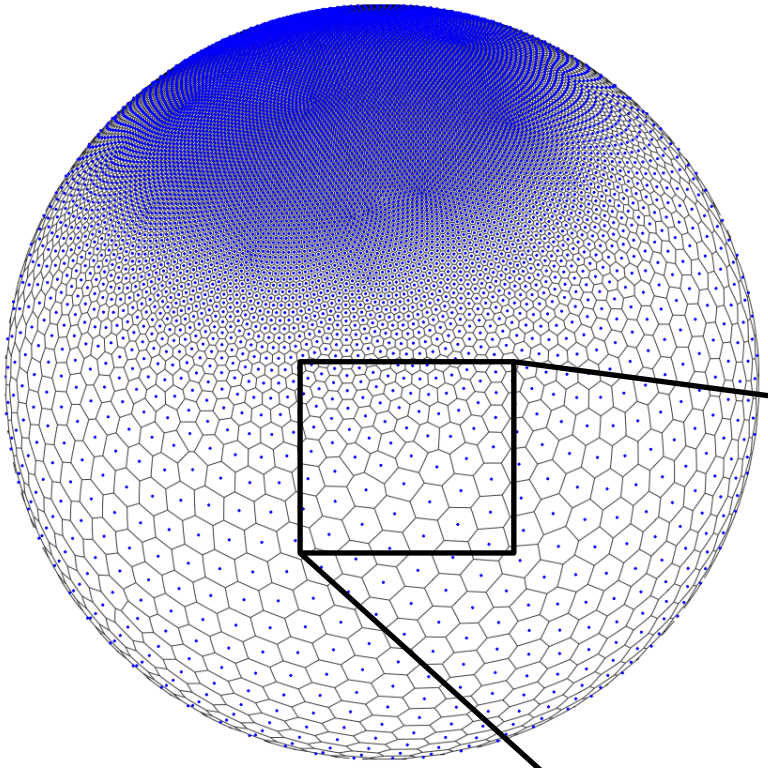
Voronoi = each grid volume (cell) V_i is uniquely associated with a *generating point* $\boldsymbol{x}_i$ such that all points within V_i are closer to $\boldsymbol{x}_i$ than to any other $\boldsymbol{x}_j$

- Lines joining generating points of adjacent cells are

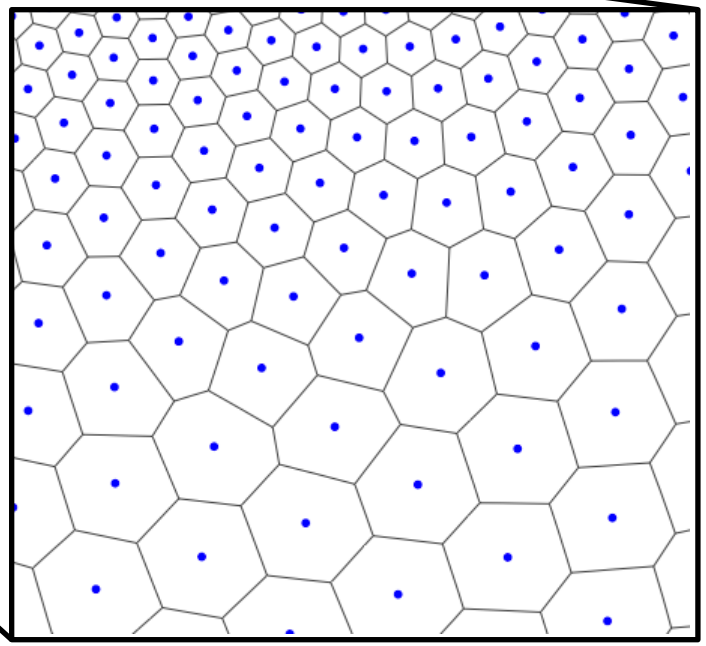
1. bisected by the shared cell face; and
2. intersect the shared cell face at a right angle.

Centroidal = the generating point for each Voronoi cell is also the mass centroid of that cell (**w.r.t. some density function**)



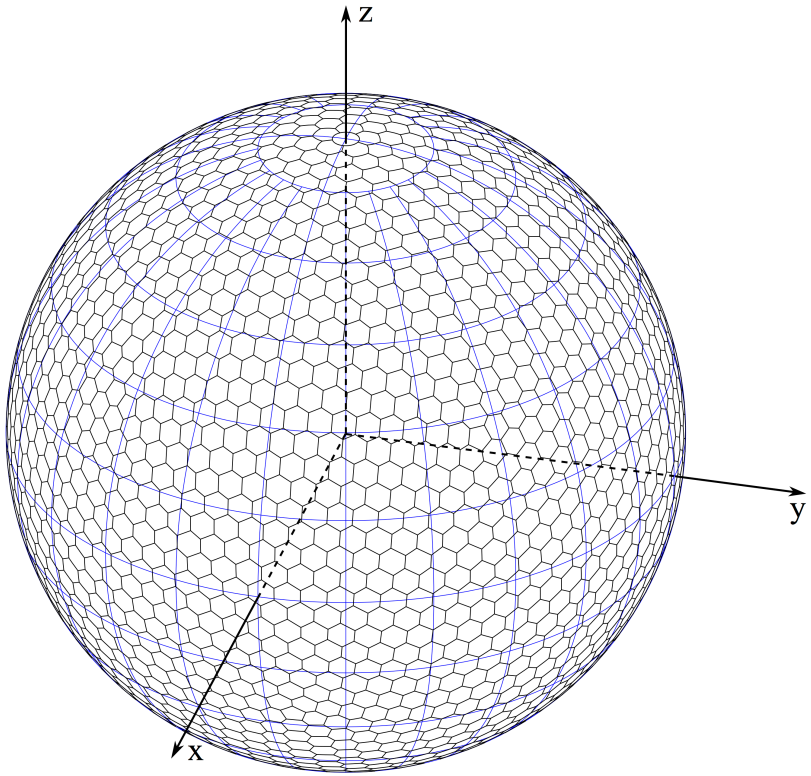


- Given a set of generating points, the primal mesh (finite volume mesh) in MPAS is defined by the Voronoi tessellation induced by this set
- The centroidal aspect of CVTs is used to produce meshes with smoothly-varying resolution



Observe that we have a fully unstructured horizontal mesh, not just a deformation of the icosahedral mesh! ***Cells may have 5, 6, 7, or more sides****.*

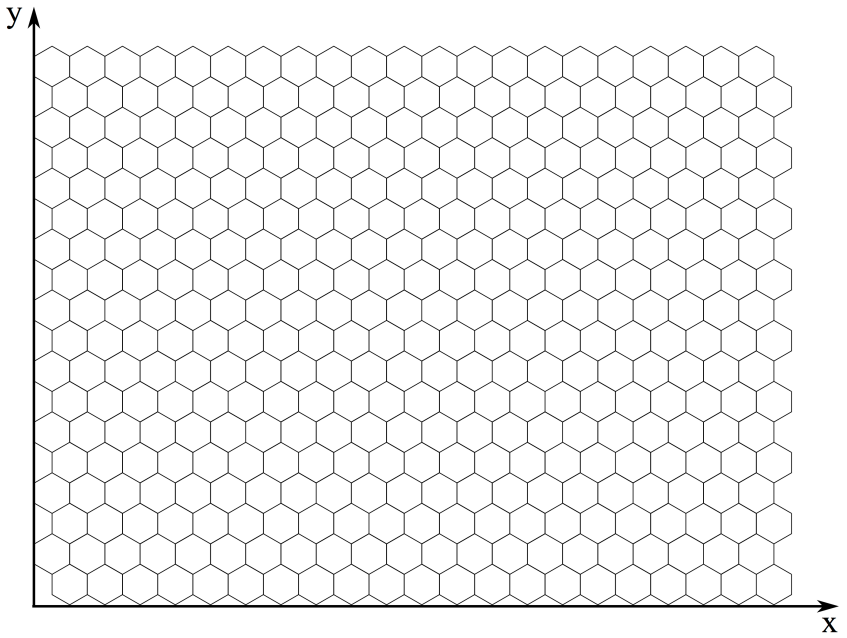
MPAS-Atmosphere supports meshes in two geometries:



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

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

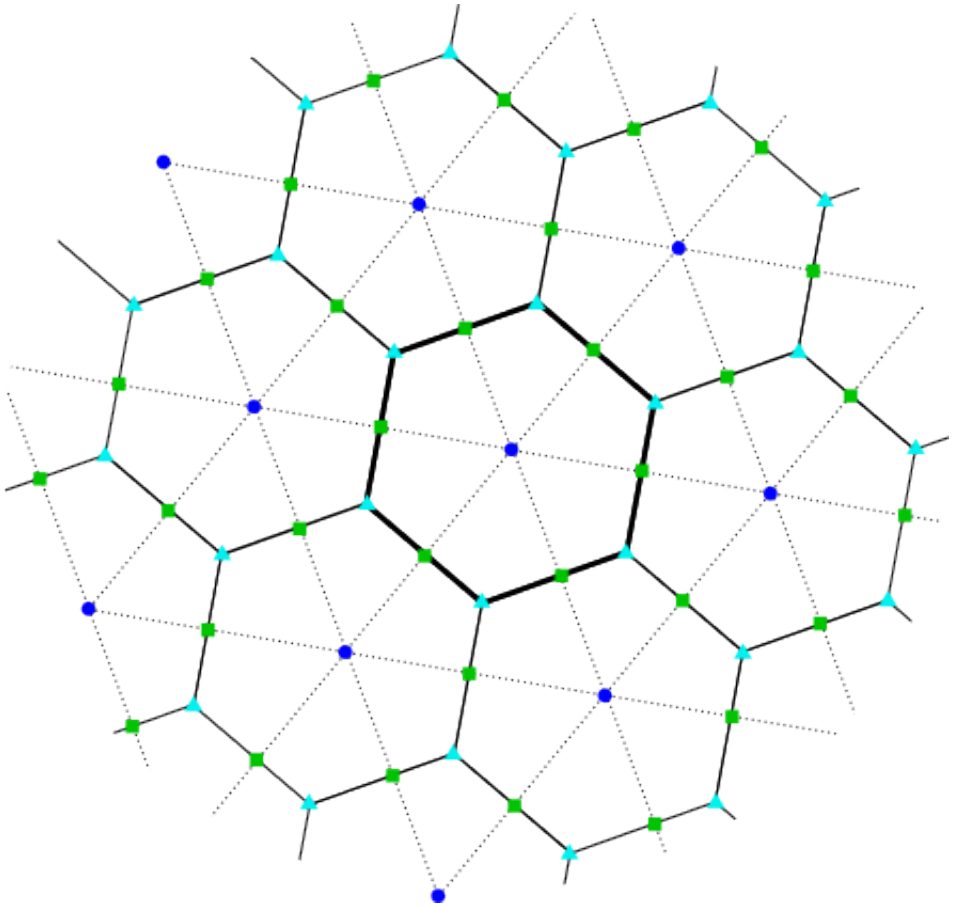
$$\phi = \arcsin\left(\frac{z}{r}\right)$$
$$\lambda = \arctan\left(\left|\frac{y}{x}\right|\right)$$



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

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

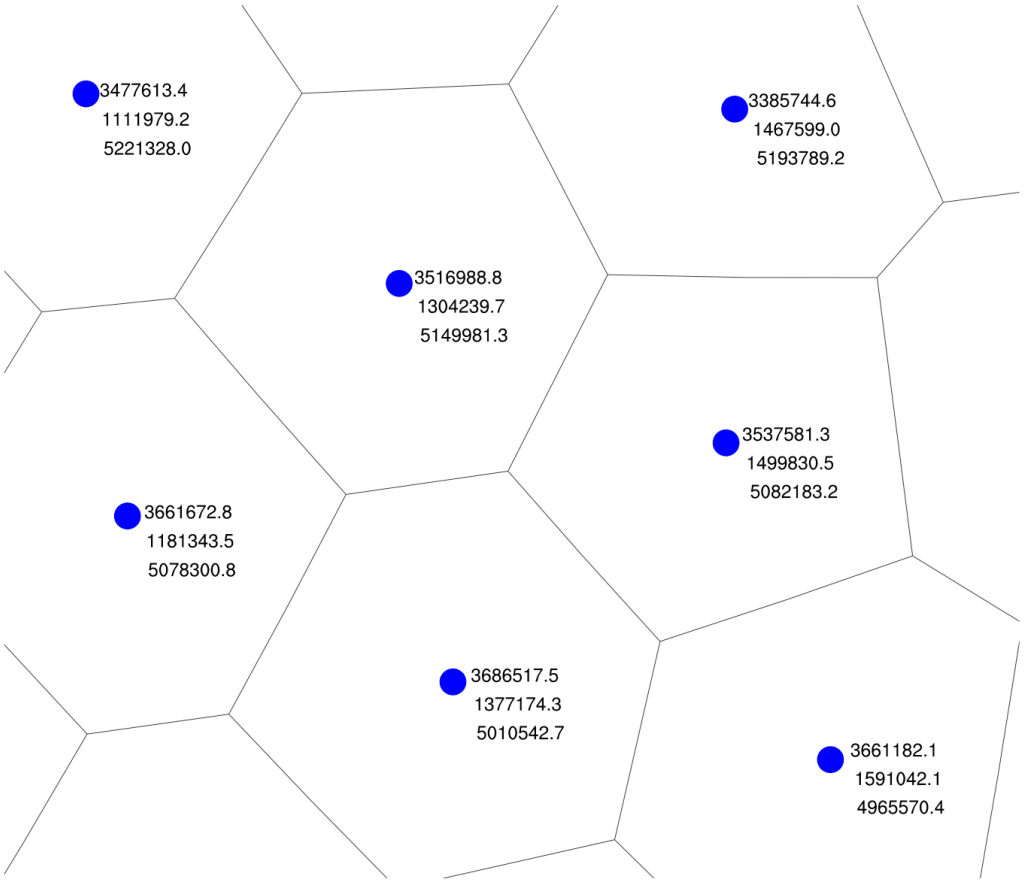
MPAS MESHES



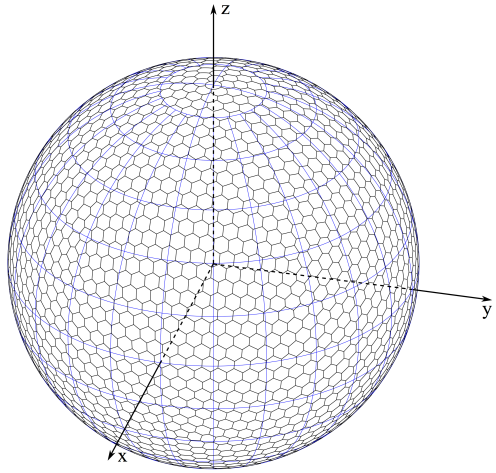
MPAS infrastructure recognizes fields defined at three locations:

- **Cells** (blue circles) - the generating points of the Voronoi mesh
- **Edges** (green squares) - the points where the dual mesh edges intersect the primal mesh edges
- **Vertices** (cyan triangles) - the corners of primal mesh cells

In MPAS-A, these locations are used to implement a C-staggered grid based on the Voronoi tessellation: prognosed normal velocities are located at *edges*, and other prognosed quantities are nominally located at *cells*.



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



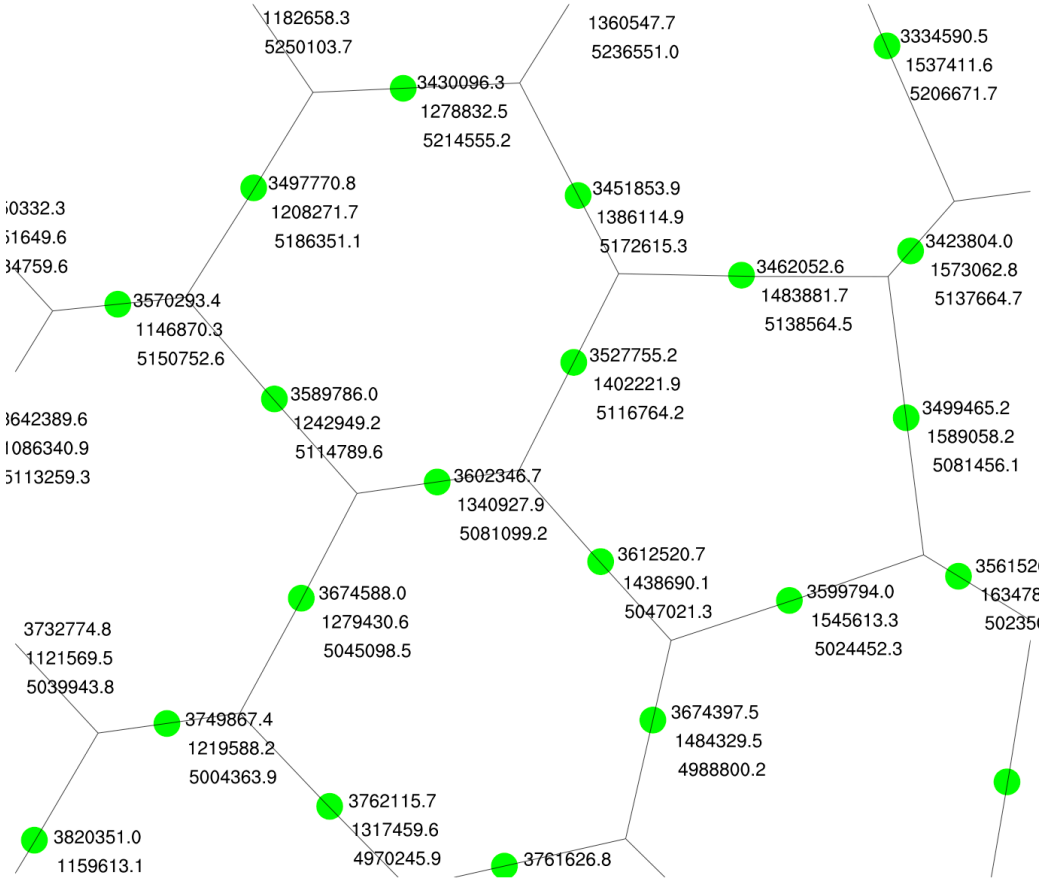
Global Cartesian coordinates are computed for each element

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

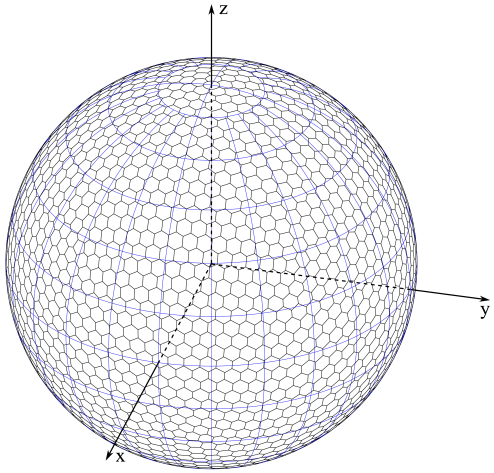
For cells: **xCell**, **yCell**, **zCell**

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



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



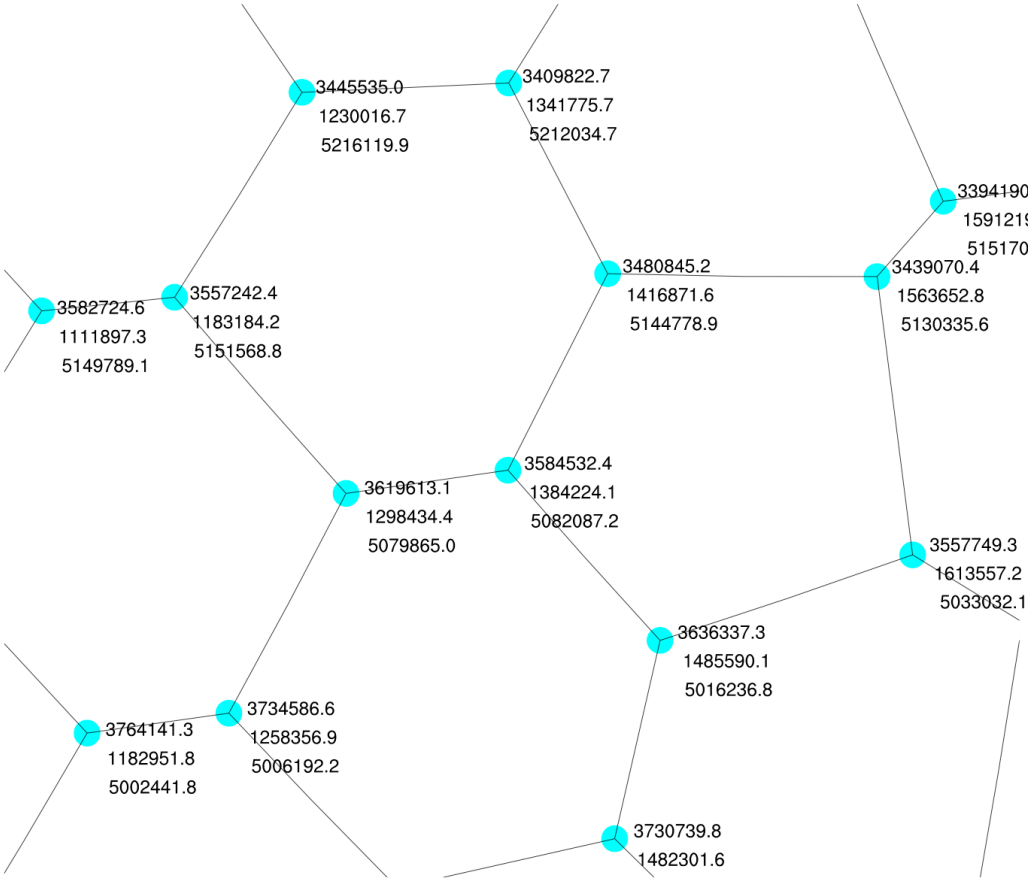
Global Cartesian coordinates are computed for each element

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

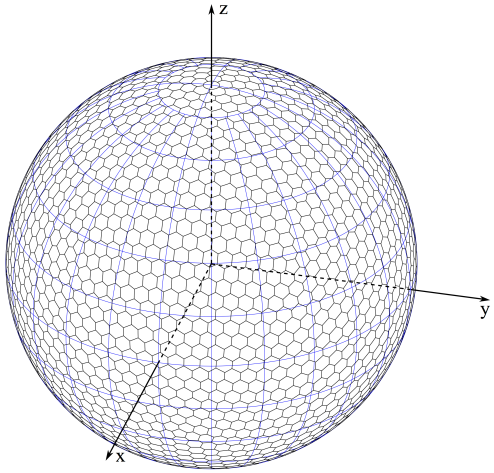
For cells: **xEdge**, **yEdge**, **zEdge**

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



Above: Cartesian coordiantes 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

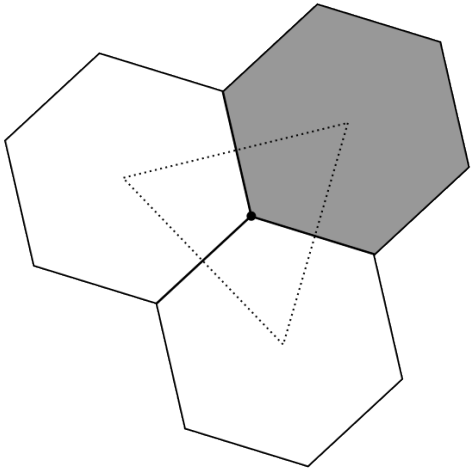
- For planar meshes, coordinates lie in the plane z=0
- For spherical meshes, coordinates lie on the surface of the sphere

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

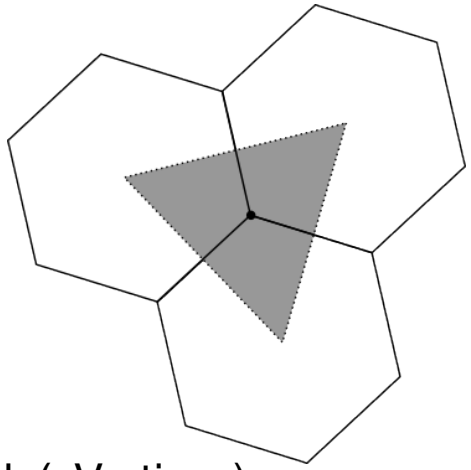
Latitudes and longitudes are computed from Cartesian coordinates as described earlier

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

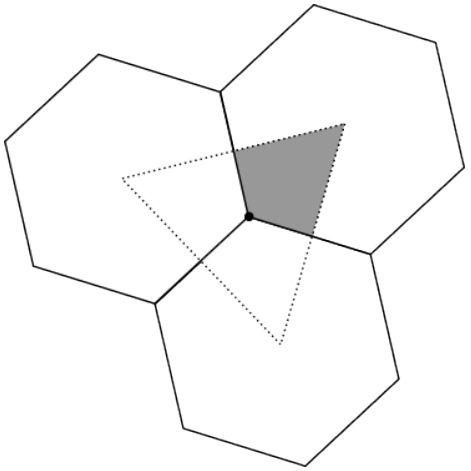
Given cell, edge, and vertex locations, we can compute some useful fields:



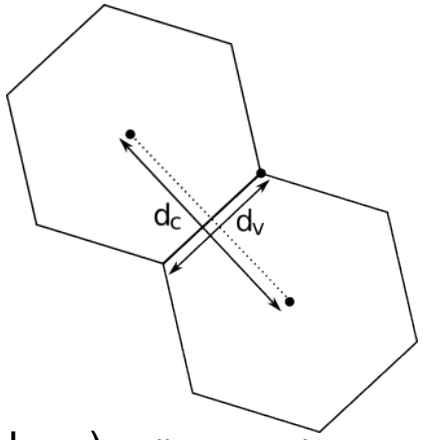
areaCell(nCells) – area of each cell



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



kiteAreasOnVertex(3,nVertices) – area of intersection between dual- and primal-mesh cells



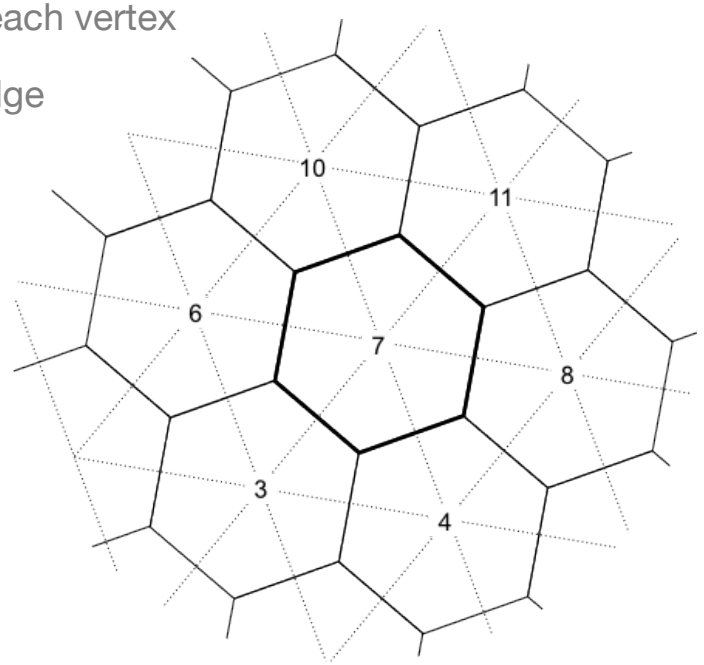
dcEdge(nEdges) – distances between cell centers

dvEdge(nEdges) – length of each edge

Meshes are explicitly described in MPAS by a set of connectivity arrays:

- 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(3,nVertices) – the indices of edges incident with each vertex
- verticesOnEdge(2,nEdges) – the indices of endpoint vertices for each edge
- cellsOnVertex(3,nVertices) – the indices of cells meeting at each vertex
- cellsOnEdge(2,nEdges) – the indices of cells split 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 cell, edge, and vertex indices are re-numbered to a contiguous range.

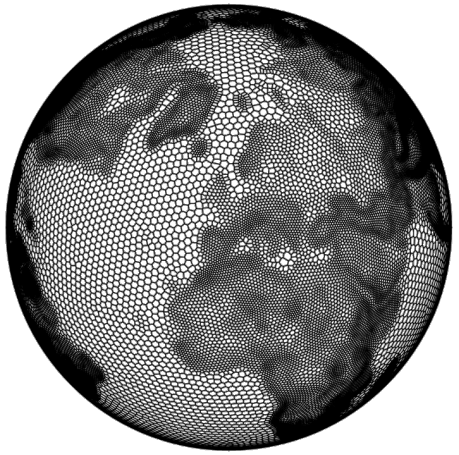
OUTLINE

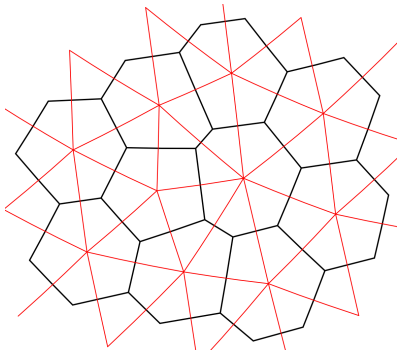
1. Introduction to MPAS meshes
   - What are centroidal Voronoi Tessellations, and how do we represent them in MPAS?

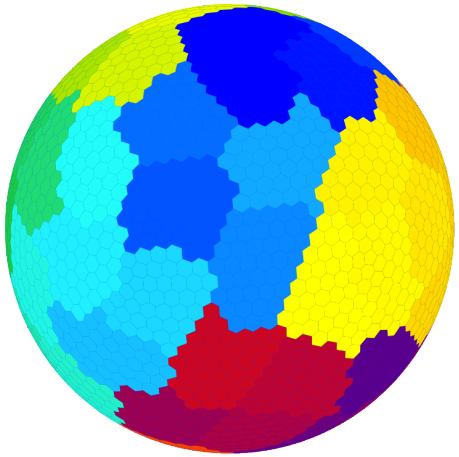
2. Considerations for handling unstructured meshes
   - In an unstructured mesh, how do we define the “positive” flow direction?

3. Mesh partitioning
   - How are meshes divided among processors?

4. Mesh generation
   - How do we achieve variable-resolution in an MPAS mesh?

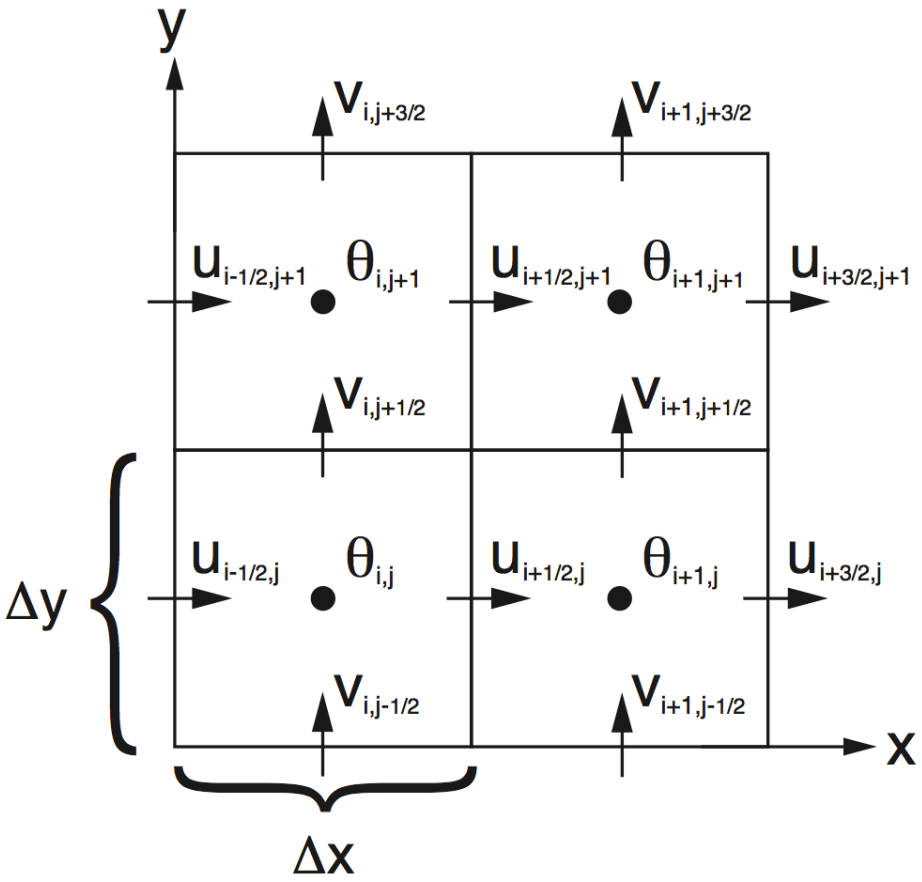


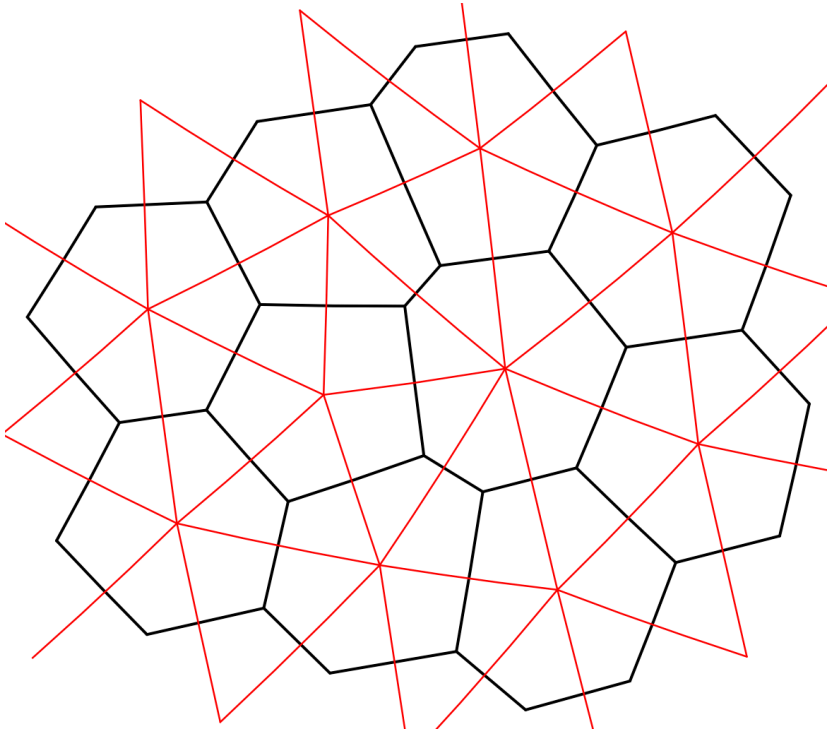




HORIZONTAL WIND VECTORS

In a rectangular C-grid, which directions represent positive U and positive V?



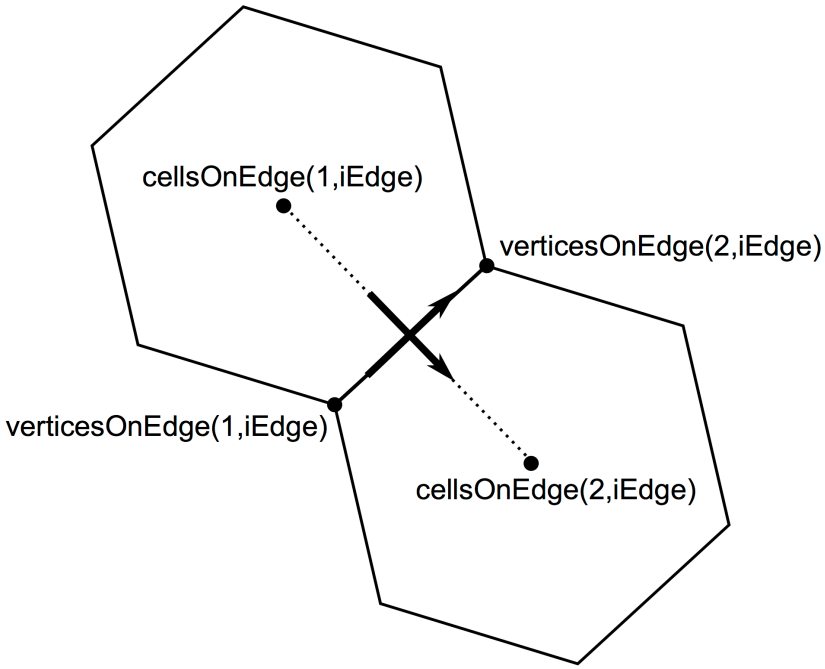


On a rectangular grid, one might say that positive U flows from left to right, and positive V flows from bottom to top when looking down on the xy-plane.

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

HORIZONTAL WIND VECTORS (2)

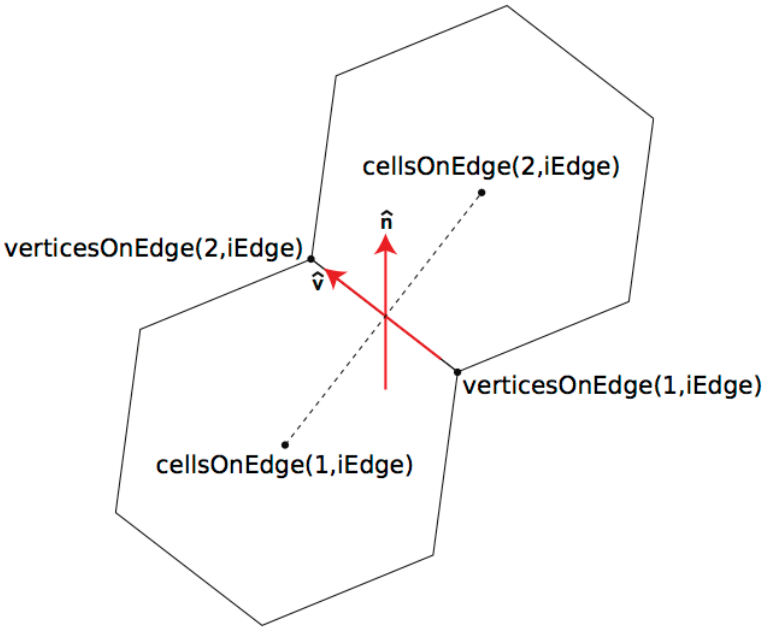
Without grid cells arranged in rows and columns, we need some simple rules to unambiguously define the positive directions for velocities



- Positive u (normal) velocity is always defined as flow from **cellsOnEdge(1,iEdge)** to **cellsOnEdge(2,iEdge)** for an edge iEdge
- Positive v (tangential) velocity is always defined as flow from **verticesOnEdge(1,iEdge)** to **verticesOnEdge(2,iEdge)** for an edge iEdge
- The cross product of the positive *u* and *v* vectors always points upward (out of the plane)

HORIZONTAL WIND VECTORS (3)

The absolute geographic orientation of each edge is specified with the field *angleEdge*, which gives the angle between local north and the positive tangential direction for an edge



angleEdge(nEdges) – angle of edge w.r.t. true north

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

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.

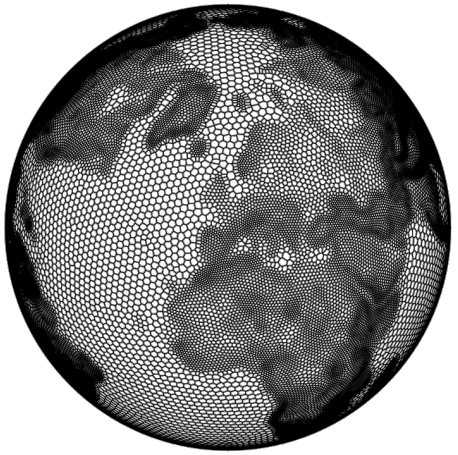
OUTLINE

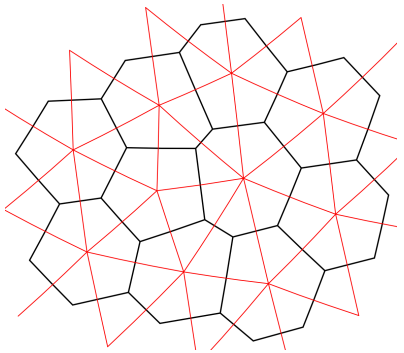
1. Introduction to MPAS meshes
   - What are centroidal Voronoi Tessellations, and how do we represent them in MPAS?

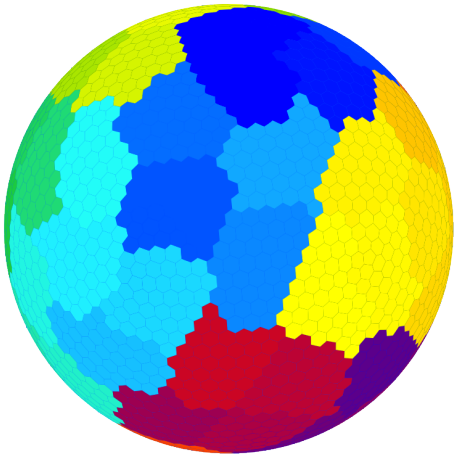
2. Considerations for handling unstructured meshes
   - In an unstructured mesh, how do we define the “positive” flow direction?

3. Mesh partitioning
   - How are meshes divided among processors?

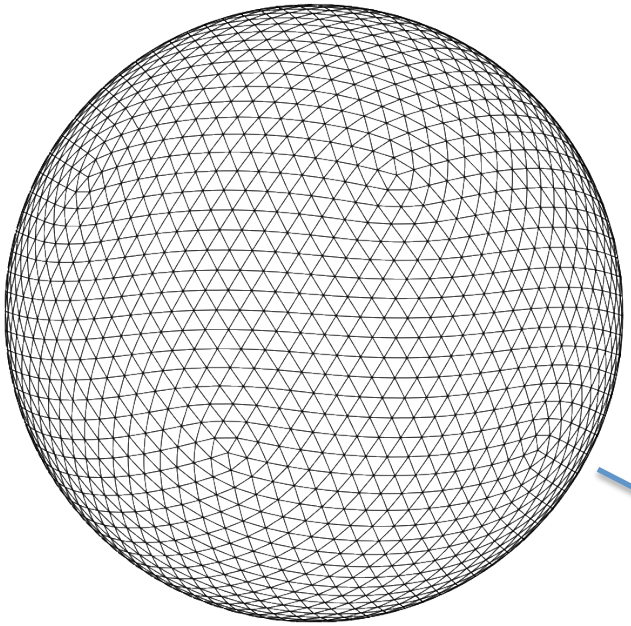
4. Mesh generation
   - How do we achieve variable-resolution in an MPAS mesh?







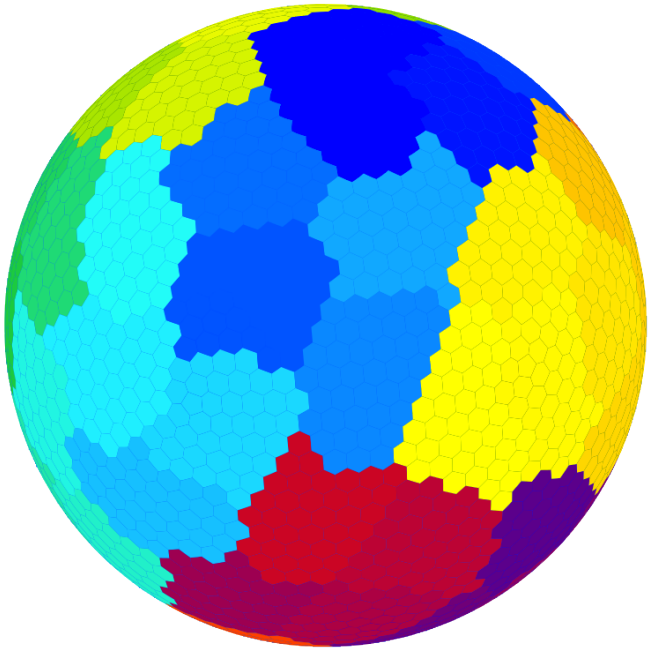
PARALLEL DECOMPOSITION



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

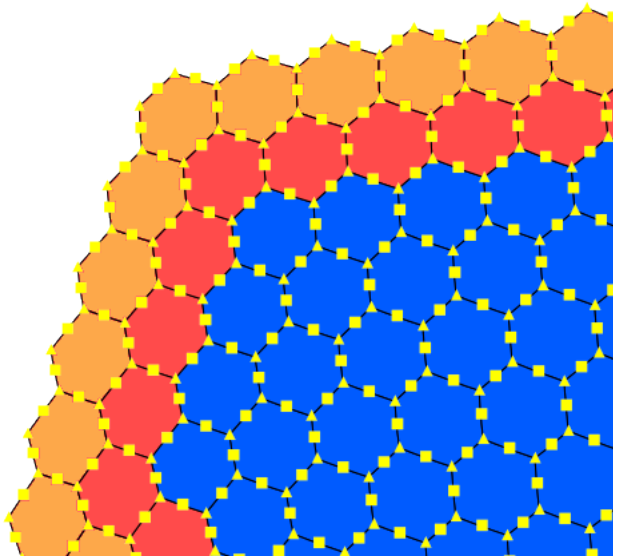
- Currently done as a pre-processing step, but could be done “on-line”

Metis also handles weighted graph partitioning

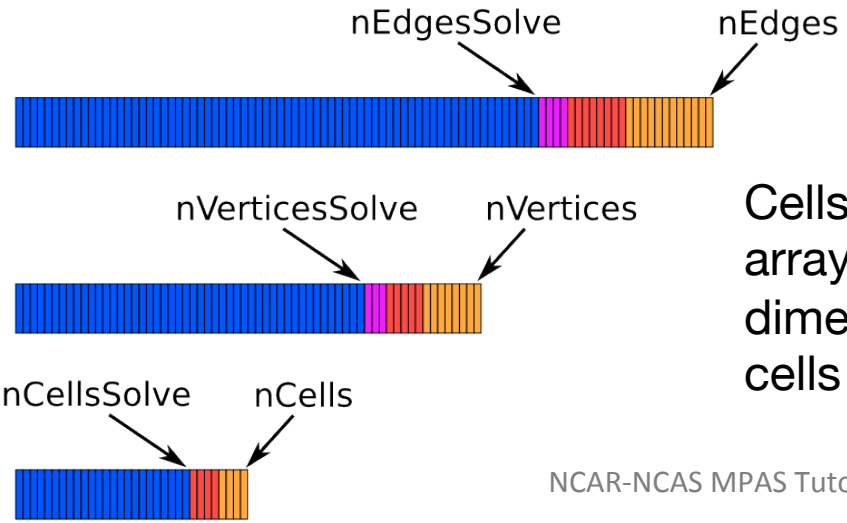
- Given *a priori* estimates for the computational costs of each grid cell, we can better balance the load among processes

PARALLEL DECOMPOSITION (2)

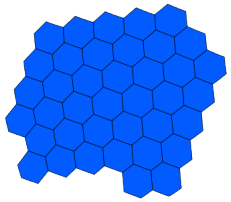
Given an assignment of cells to a process, any number of layers of halo (ghost) cells may be added



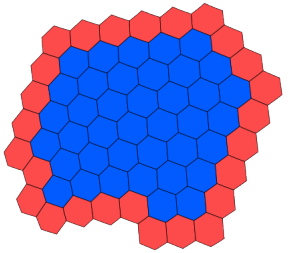
With a complete list of cells stored in a block, adjacent edge and vertex locations can be found; we apply a simple rule to determine ownership of edges and vertices adjacent to real cells in different blocks



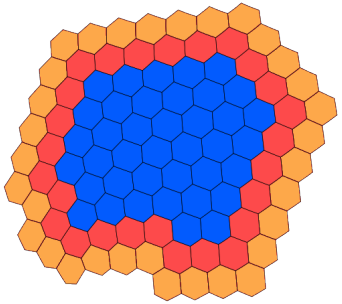
Cells are stored in a 1d array (2d with vertical dimension, etc.), with halo cells at the end of the array



Block of cells owned by a process



Block plus one layer of halo/ghost cells

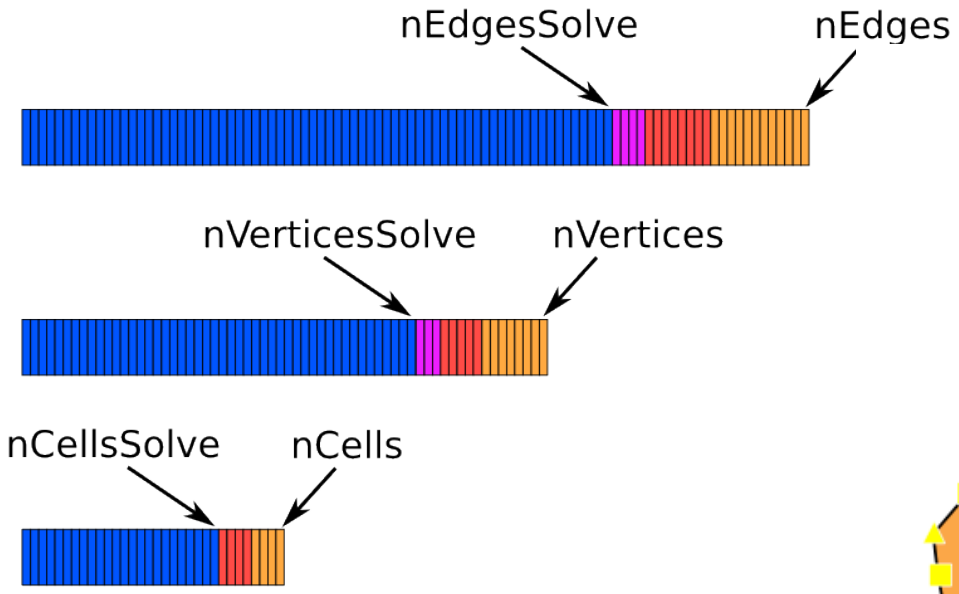


Block plus two layers of halo/ghost cells

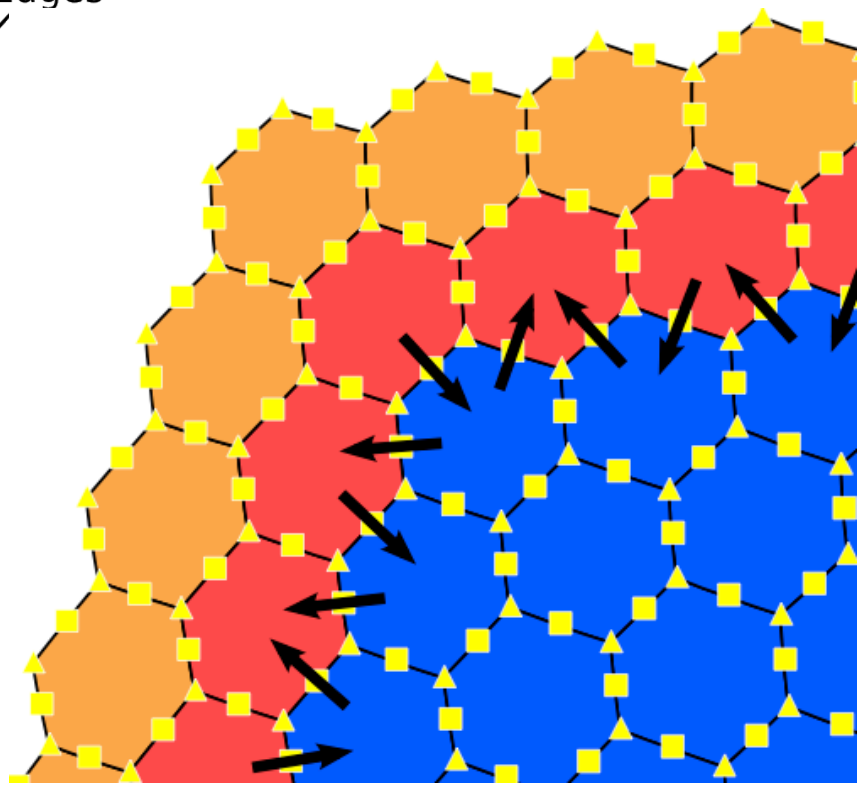
PARALLEL DECOMPOSITION (3)

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

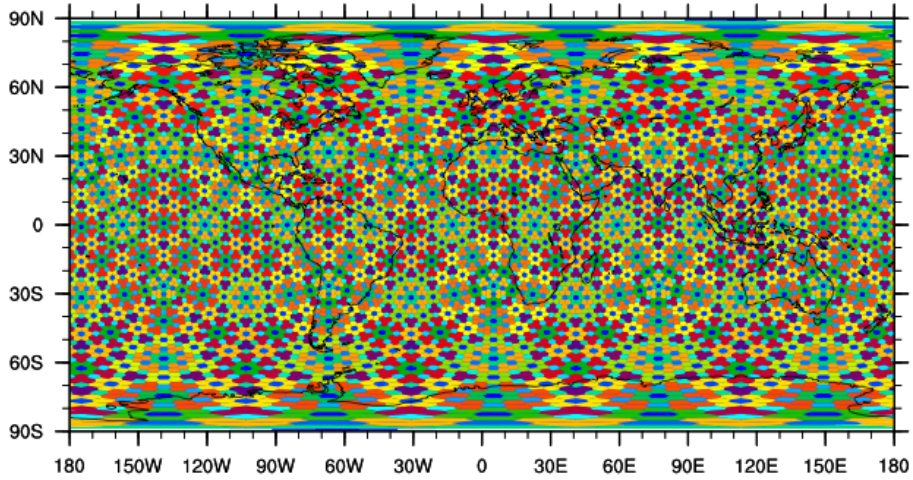


For *n* layers of ghost cells, we have *n+1* layers of ghost edges and ghost vertices.

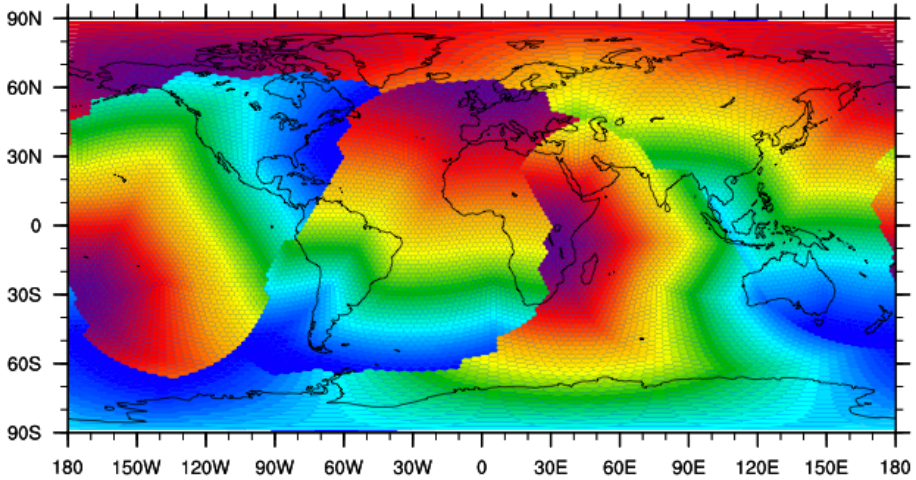


MISCELLANEOUS NOTES ON GRID INDEXING

- Vertical dimension inner-most (fastest varying in Fortran)
  - E.g., **`theta(nVertLevels,nCells)`**
- We do not perform any sort of local cell re-ordering, e.g., in an attempt to improve cache reuse



Cells of a 10242 mesh colored according to their global index



Cells of a 10242 mesh decomposed across 4 MPI tasks and colored according to their re-ordered local index

- Indices exist that allow a solver to process just the “owned”, any individual halo layers, or all cells/edges/vertices
  - Allows us to reduce redundant computation as much as possible
  - Provides fine granularity for trading computation and communication

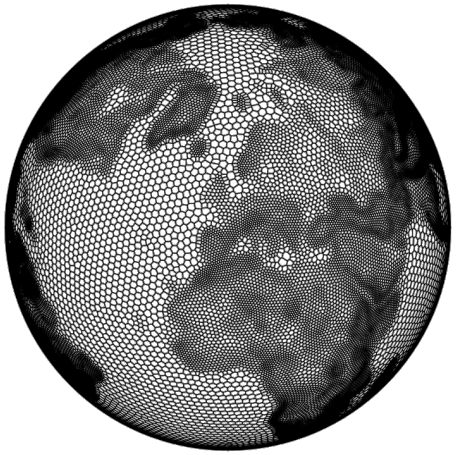
OUTLINE

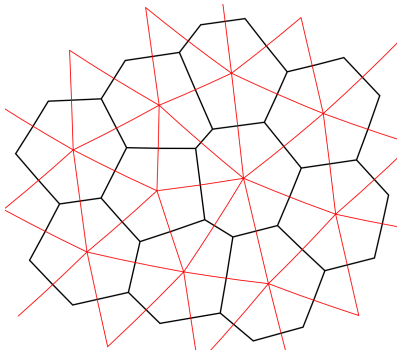
1. Introduction to MPAS meshes
   - What are centroidal Voronoi Tessellations, and how do we represent them in MPAS?

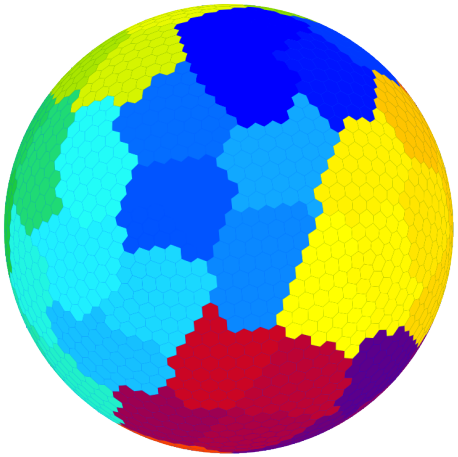
2. Considerations for handling unstructured meshes
   - In an unstructured mesh, how do we define the “positive” flow direction?

3. Mesh partitioning
   - How are meshes divided among processors?

4. Mesh generation
   - How do we achieve variable-resolution in an MPAS mesh?

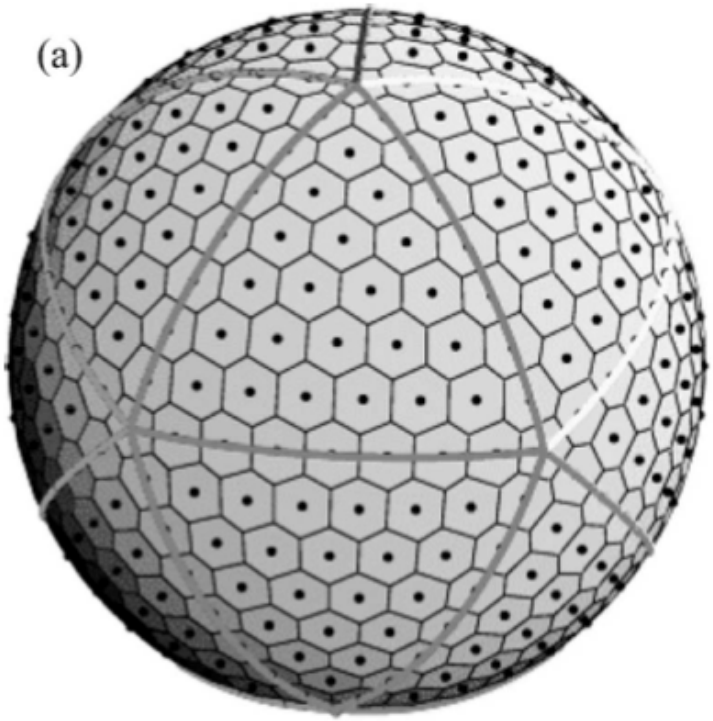


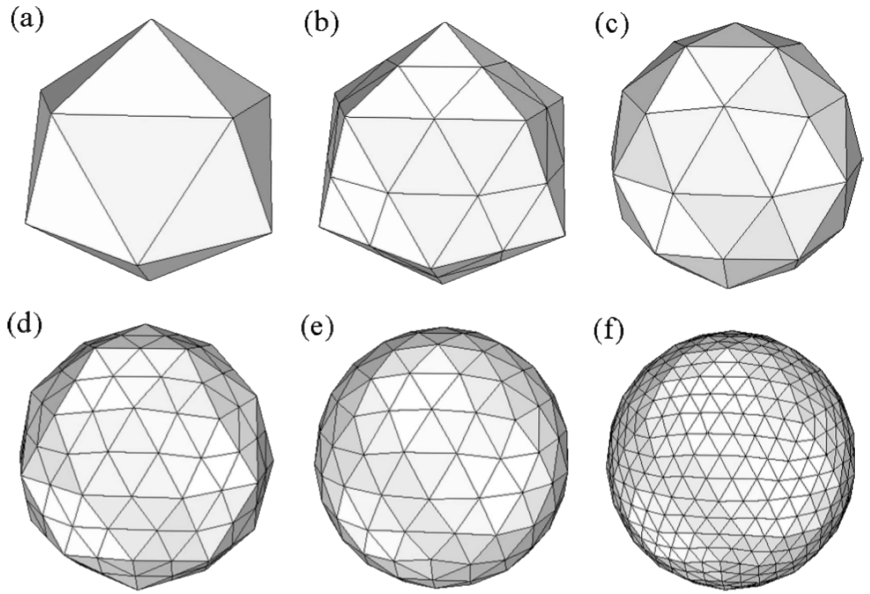




HOW DO WE CREATE MESHES FOR MPAS-ATMOSPHERE?

- For quasi-uniform Voronoi tessellations, we can employ successive subdivision of the icosahedron
  - The vertices of these triangular meshes can be used as the generating points for a spherical Voronoi tessellation
  - To create a spherical ***centroidal*** Voronoi tessellation, some iteration is required

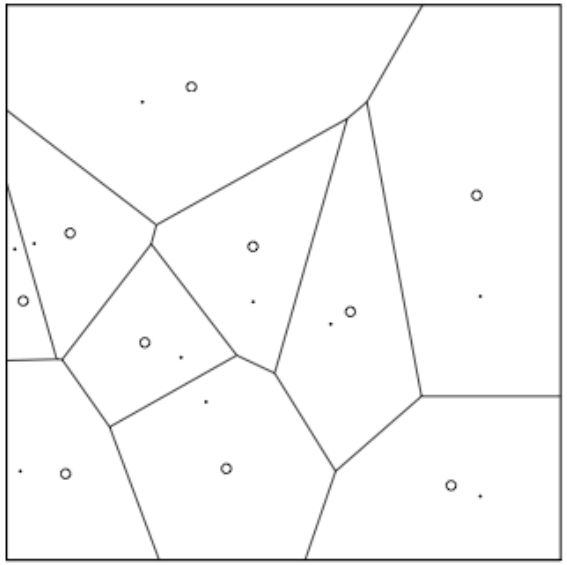


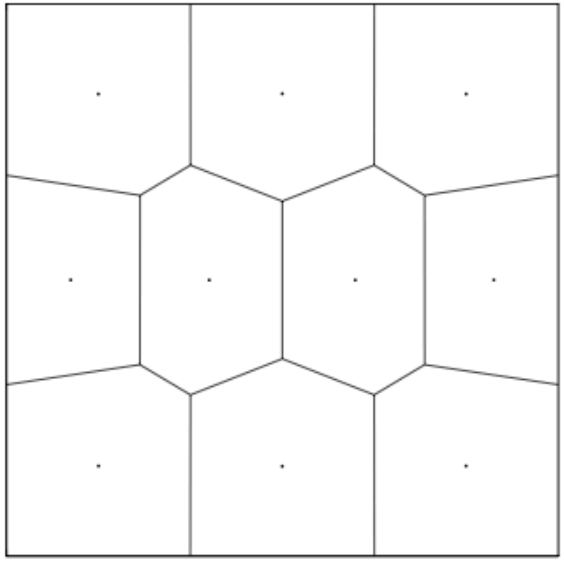


From Lipscomb and Ringler (2005)

Given an initial set of generating points, **Lloyd's method** may be used to arrive at a CVT:

1. Begin with any set of initial points ("generating points")
2. Construct a Voronoi diagram for the point set
3. Locate the mass centroid of each Voronoi cell
4. Move each generating point to the mass centroid of its Voronoi cell
5. Repeat 2-4 to convergence



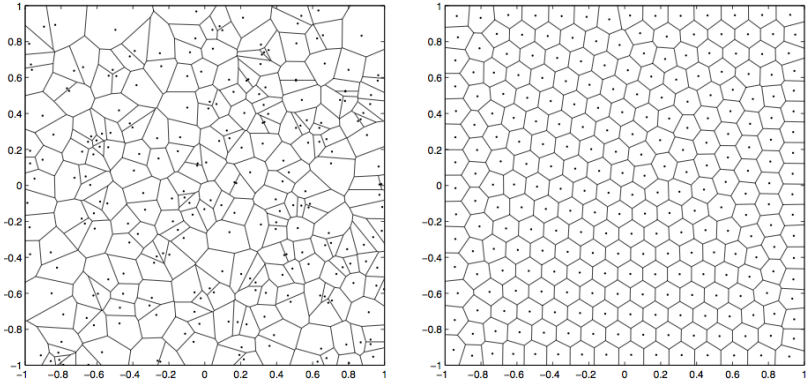


From Du et al. (1999)

MacQueen's method, an randomized alternative to Lloyd's method may also be used; no Voronoi diagrams need to be constructed, but convergence is generally much slower

Lloyd's method can be viewed as the minimization of an energy functional; in the plane, it can be shown that hexagonal Voronoi cells provide the minimum energy configuration *for constant density*

To create regions of grid refinement, we simply define a non-uniform density function over the domain, and use this when computing the mass centroids of Voronoi cells in Lloyd's method



From Du et al. (1999)

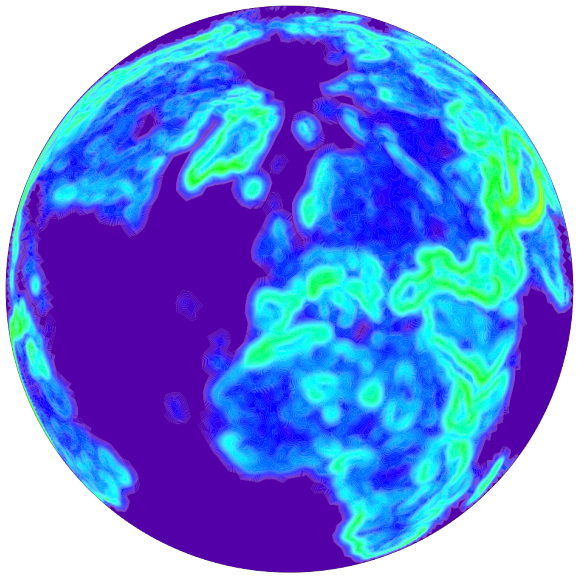
For a density function $\rho(\boldsymbol{x}) > 0$, it is conjectured (Ju et al. (2010)) that, as the number of Voronoi cells increases, the diameters, *h*, of Voronoi cells are related by

$$\frac{h_i}{h_j} \approx \left(\frac{\rho(\mathbf{x}_j)}{\rho(\mathbf{x}_i)} \right)^{1/(d'+2)}$$

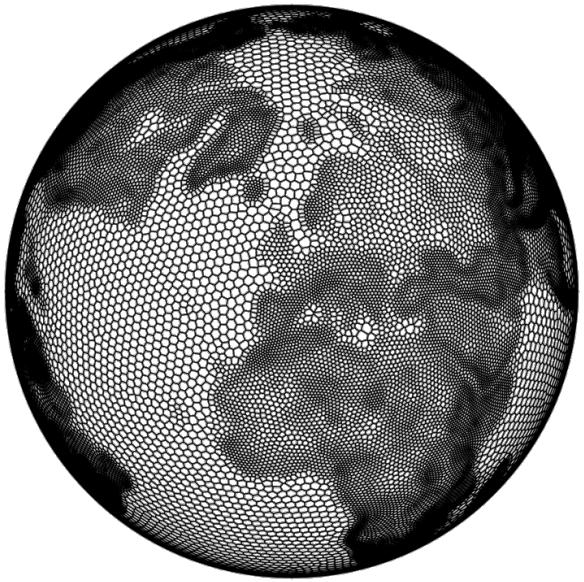
where *d'* is 2.

AN ABSURD EXAMPLE OF MESH REFINEMENT

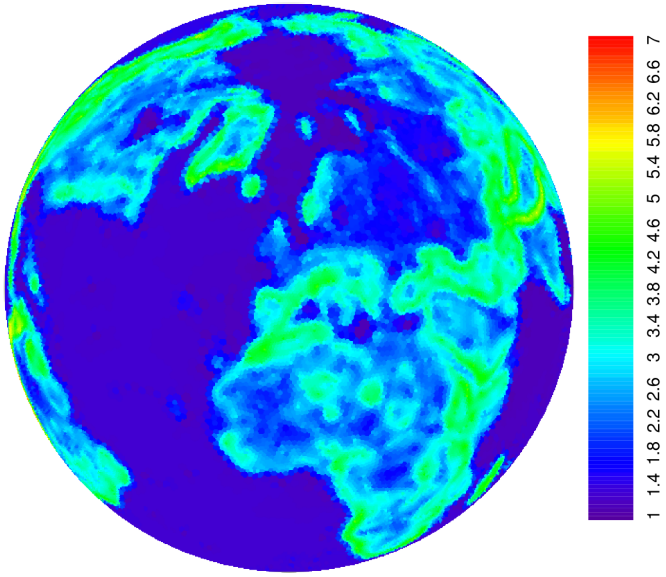
Define ρ(***x***) as a function of the magnitude of the topography gradient, for example:



Fourth root of density function



Resulting SCVT



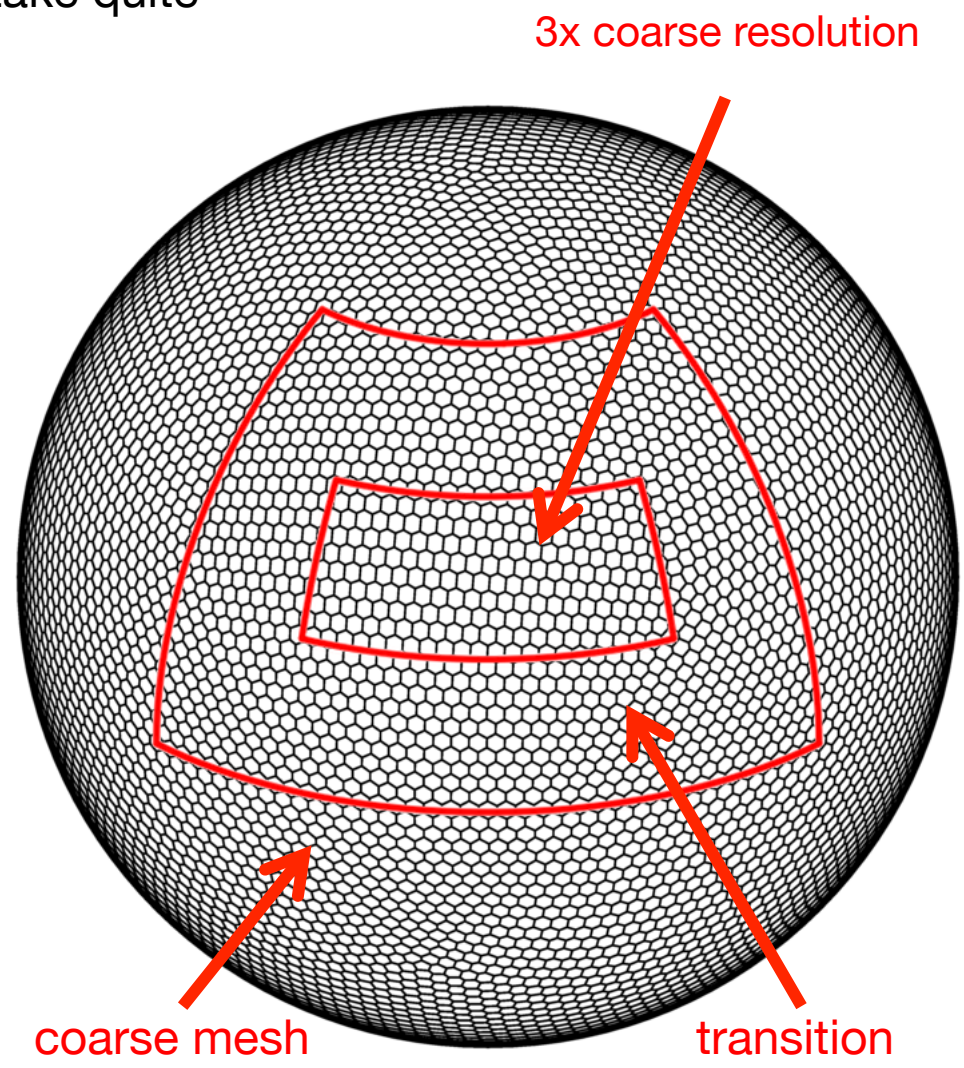
Normalized inverse mean distance of cell centroid from each of its neighboring centroids

Evidence from initial testing of the MPAS non-hydrostatic atmosphere code on multi-resolution meshes on the Cartesian plane suggests that smoother refinement (i.e., less abrupt transitions) produces better results

Meshes with more than $O(10^5)$ cells take quite a while to generate!

Typical procedure:

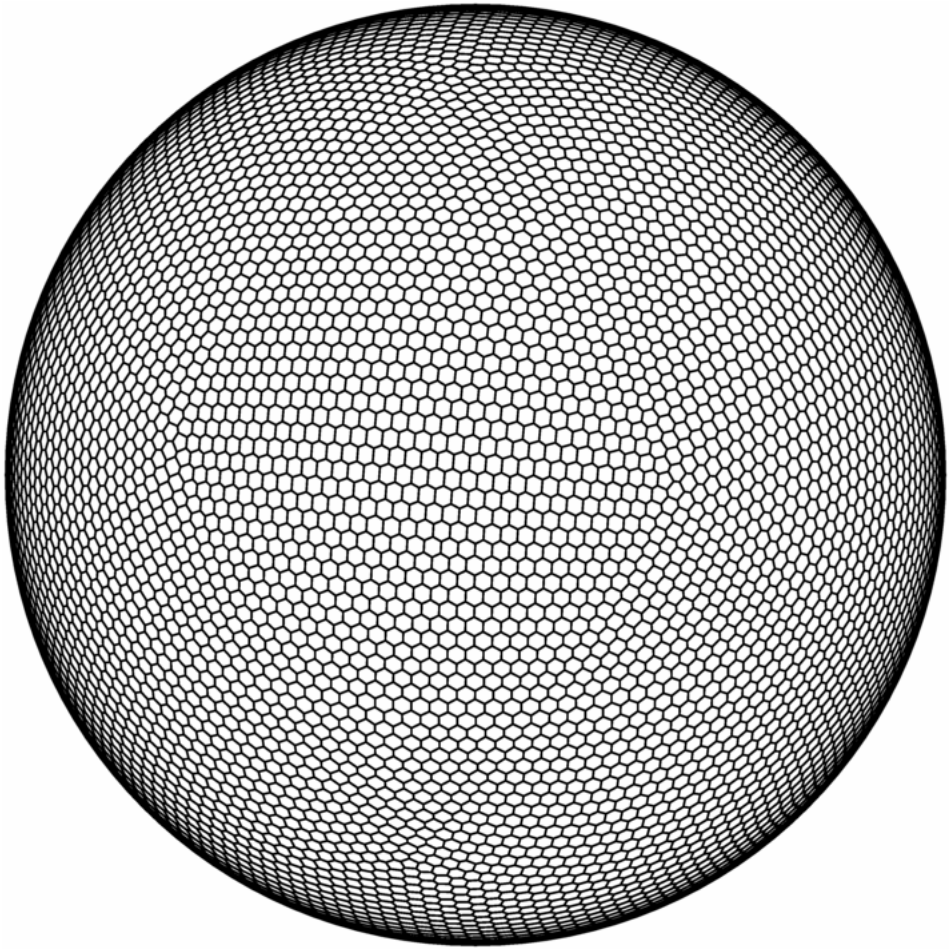
1. Decide on size/shape of refinement region, mesh resolution
2. Estimate the total number of cells needed in the mesh
3. Work backwards from final number of cells to $O(100)$ cells
   - Coarsening a mesh by a factor of two gives $(n+6)/4$ cells
4. Iterate to convergence, bisect the mesh, repeat



Meshes with more than $O(10^5)$ cells take quite a while to generate!

Typical procedure:

1. Decide on size/shape of refinement region, mesh resolution
2. Estimate the total number of cells needed in the mesh
3. Work backwards from final number of cells to $O(100)$ cells
   - Coarsening a mesh by a factor of two gives $(n+6)/4$ cells
4. Iterate to convergence, bisect the mesh, repeat



An alternative approach is to add refinement cells in the refinement region, and hold the rest of the mesh fixed

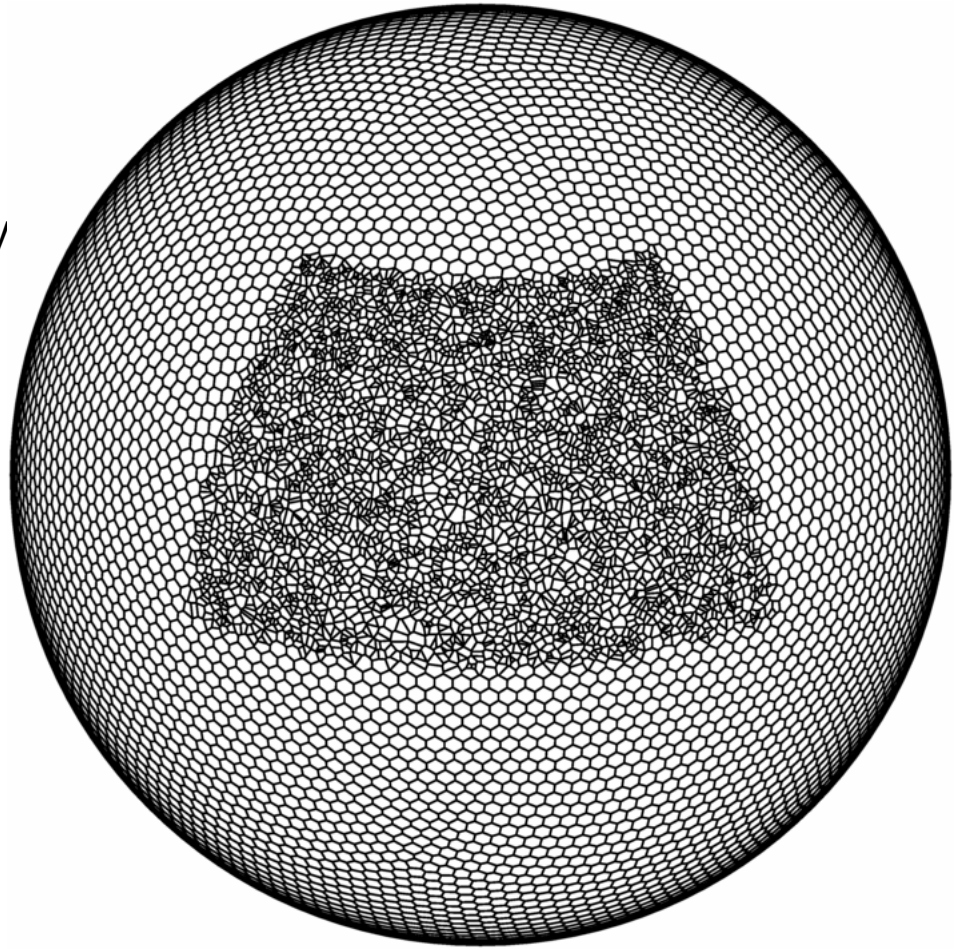
Simple Monte-Carlo method to estimate how many points (nCells) are needed to achieve a specified absolute resolution, given a density function:

ds_{min} = min. desired grid distance in the mesh

N_s = desired # samples

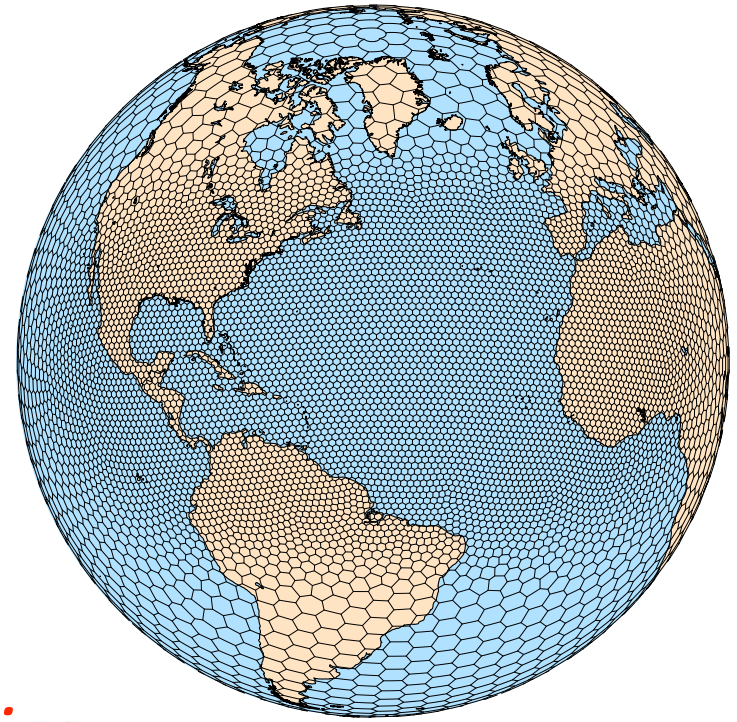
$A_s = 4\pi R^2 / N_s$

for 1 to N_s
    p = random point on sphere
    r = density(p)
    $ds = ds_{min} / r^{0.25}$
    nCells = nCells + A_s / hexagon_area(ds)
end for



MPAS

Model for Prediction Across Scales



- Overview
- Meshes
- *Atmospheric solver, physics*
- Compiling and running MPAS
- Summary
- Practical session

Los Alamos NATIONAL LABORATORY EST. 1943

U.S. DEPARTMENT OF ENERGY | Office of Science

NSF

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH