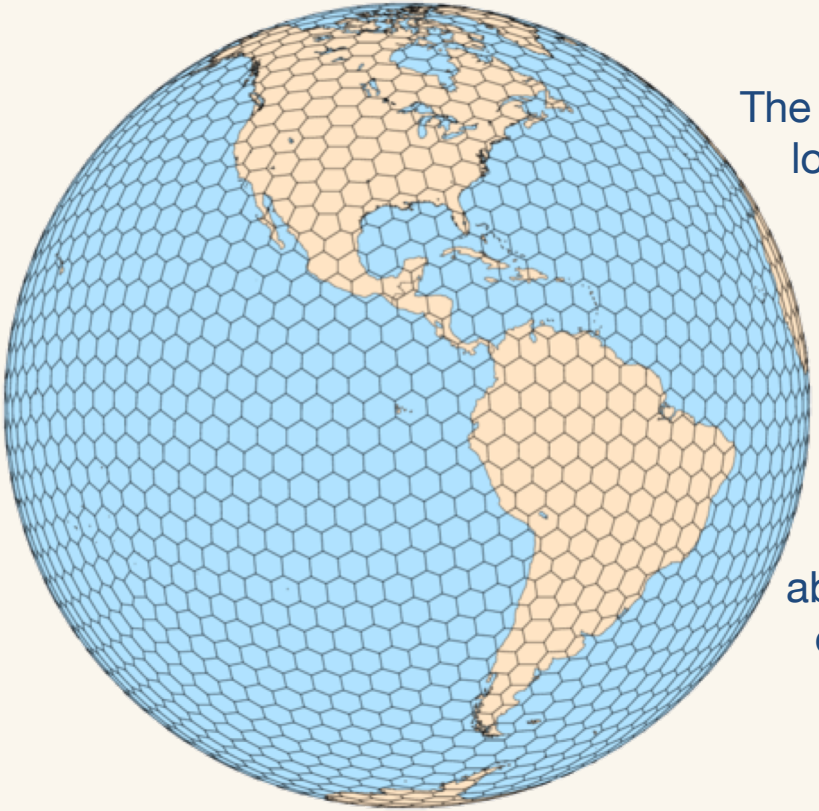
MPAS
Model for Prediction Across Scales

Generating Meshes for MPAS

MPAS
Model for Prediction Across Scales

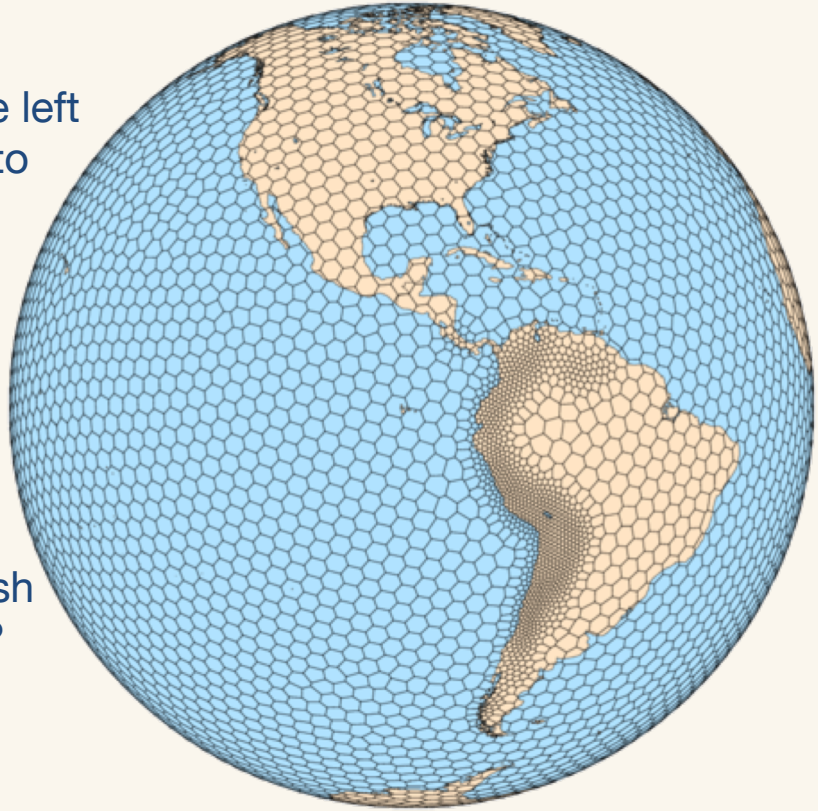
Motivation

How do we generate the meshes that are a central feature of MPAS?



The mesh on the left looks simple to generate...

but what about the mesh on the right?



Centroidal Voronoi tessellations defined

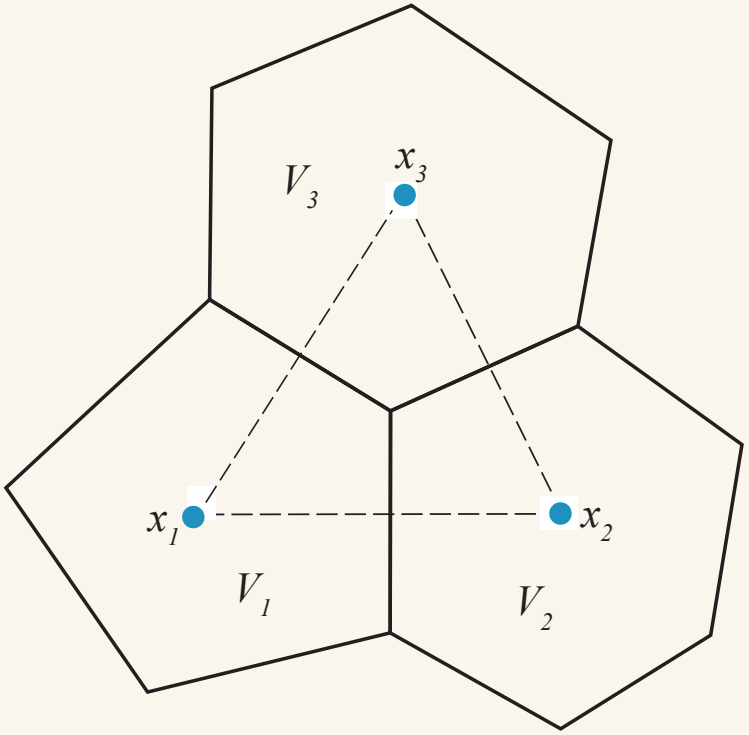
Both meshes in the preceding slide are (spherical) centroidal Voronoi tessellations:

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**)



Three Voronoi cells, V_i, along with their associated generating points, x_i. Each Voronoi cell corner is uniquely associated with a Delaunay triangle (dashed line).

Centroidal Voronoi tessellations as minimizers

Du et al.[1] show that the minimizer of a particular problem is necessarily a CVT

- It is reasonable, then, to approach the generation of CVTs as an optimization problem
- Many algorithms could be brought to bear, but we employ *Lloyd's Method*

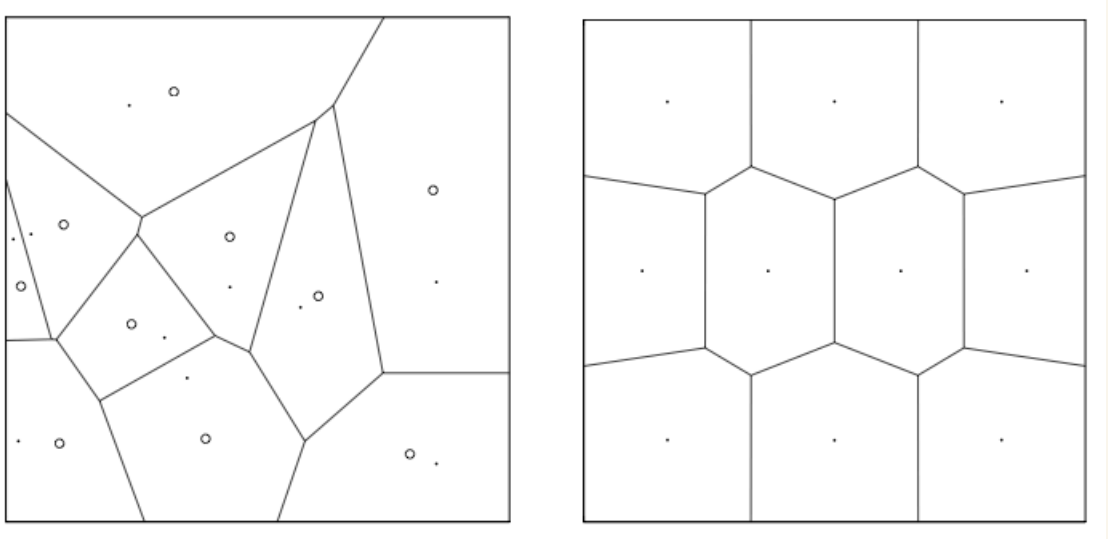
For constant density in the plane, hexagons provide the minimum energy configuration of Voronoi regions (Newman[2], also Gersho[3]).

Introduction to Lloyd's method

Originally developed by Stuart Lloyd at Bell Labs for optimal signal quantization

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



An example CVT (right) produced using Lloyd iteration, beginning from random points in the domain. From Du et al.[1]

MPAS
Model for Prediction Across Scales

Density function: the key to refinement

For a density function ρ(***x***) > 0, it is conjectured (Ju et al.[4]) 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.

By defining an appropriate density function, therefore, we can distribute cells essentially however we choose!

MPAS
Model for Prediction Across Scales

A few notes on density functions

Functions with continuous first derivatives seem to work well; most of our atmosphere meshes use a density function based on Ringler et al.[5] :

$$\rho(\mathbf{x}) = \frac{1-\gamma}{2}\left[\tanh\left(\frac{\beta - \|\mathbf{x}_c - \mathbf{x}\|}{\alpha}\right) + 1\right] + \gamma$$

Here, $\boldsymbol{x}_c$ is the center of the refinement region, β is the half-width of the high-resolution region, α is the width of the "transition zone", and γ is the minimum density. $\boldsymbol{x}_c$ and $\boldsymbol{x}$ are constrained to lie on the surface of the sphere.

How small can our "transition" zone be if we want the average cell diameter to vary by at most a factor of *r* (with $r > 1$) from any cell to its neigbors? At least

$$\frac{\Delta x_{coarse}}{r-1} - \frac{\Delta x_{fine}}{r-1}$$

MPAS
Model for Prediction Across Scales

How many generating points are needed?

Density function controls the relative resolution in mesh, while the total number of generating points set the absolute mesh resolutions

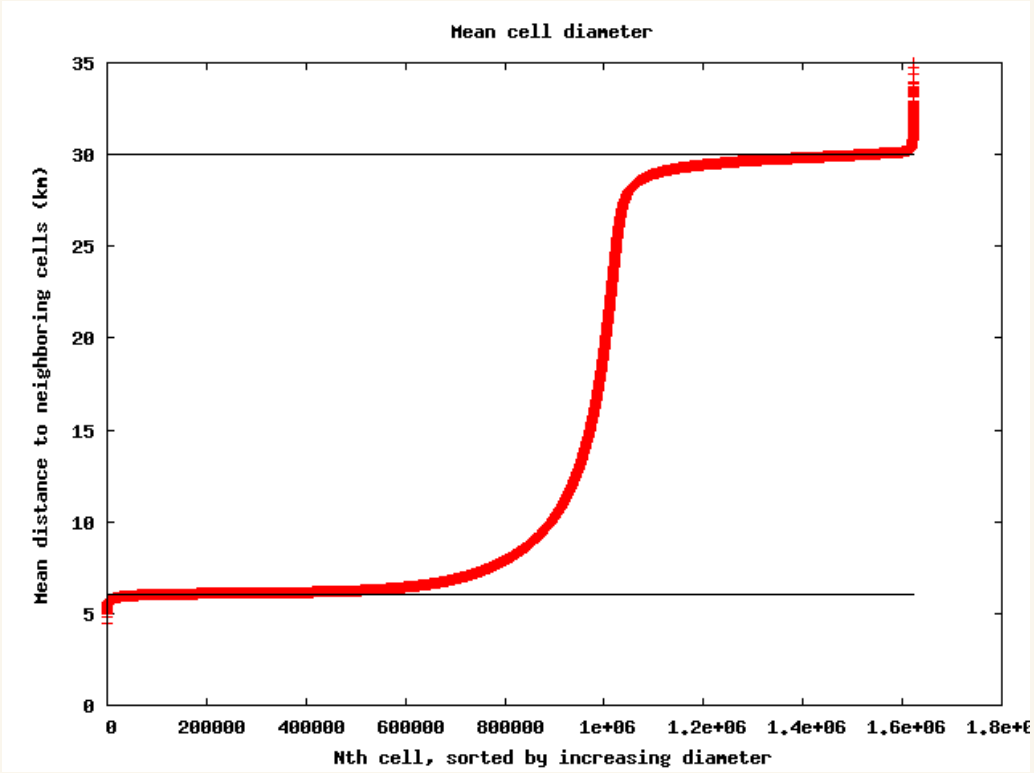
- We use a simple Monte-Carlo method to estimate the required number of generating points

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

N_s = number of samples

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

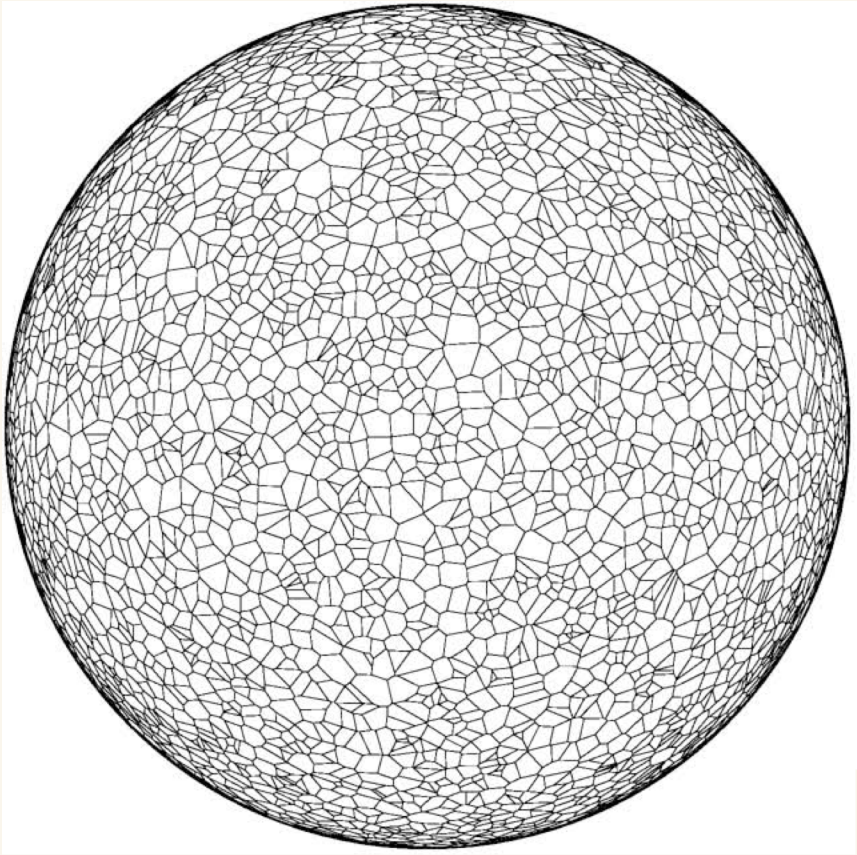
for 1 to N_s
  p = random point on sphere
  r = density(p)
  $ds = ds_{min} / r^{0.25}$
  $nCells = nCells + A_s /$ hexagon_area(ds)
end for

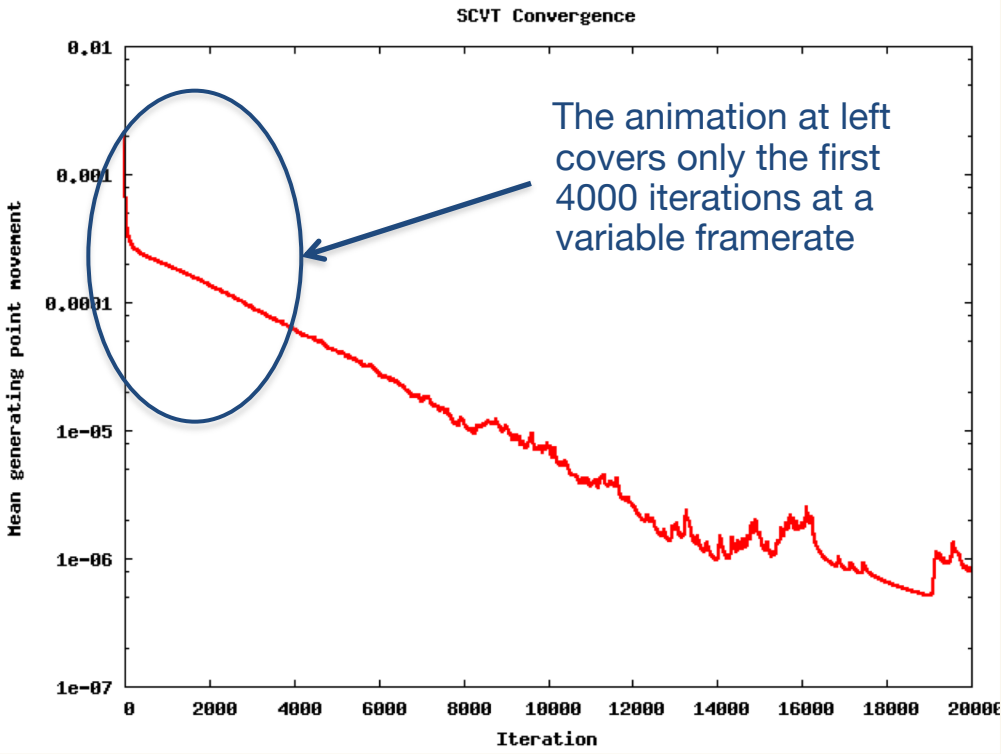


Resulting mean cell diameters from a 30-6 km mesh with the number of generating points estimated using the Monte-Carlo method

A first attempt at mesh generation

The obvious approach suggests simply defining density function, generating the required number of generating points randomly, and iterating to convergence





MPAS
Model for Prediction Across Scales

Problems with the obvious approach

With uniformly-distributed random initial generating points, there is apparently quite a lot of time spent in allowing points to migrate around the sphere

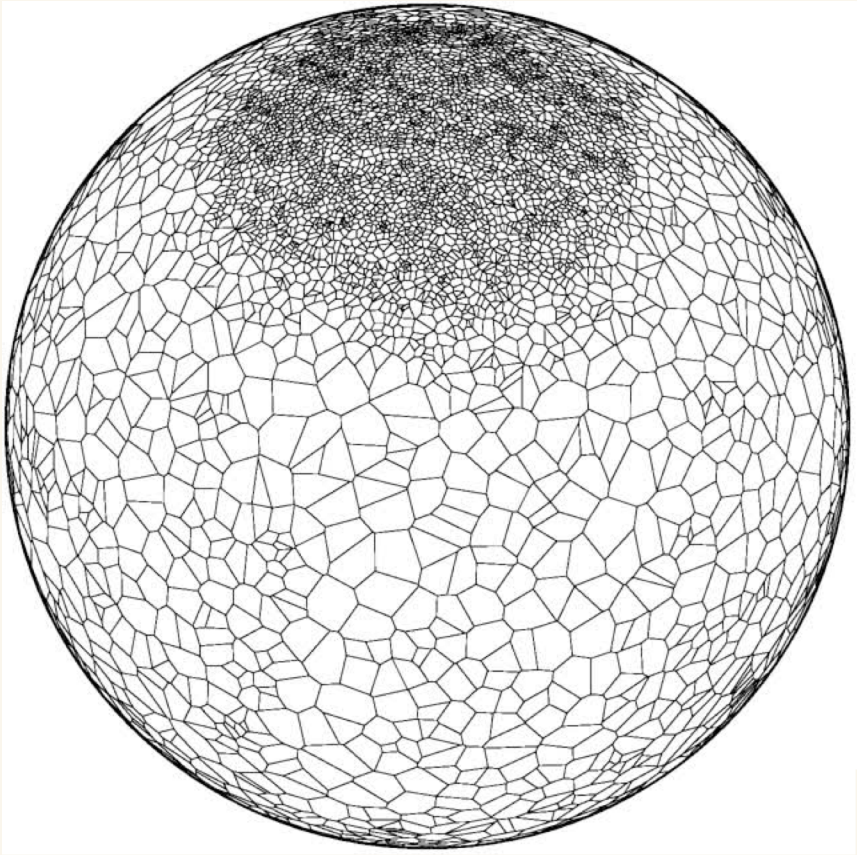
- This suggests we try to get cells into nearly their final position as "cheaply" as possible

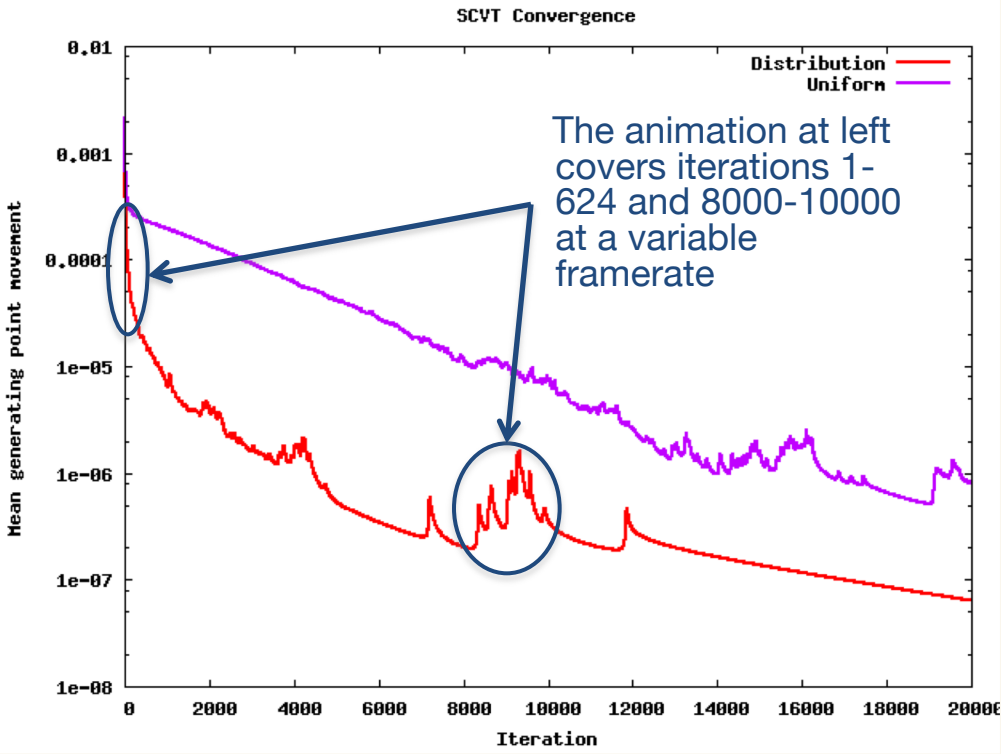
Rather than choosing uniformly-distributed points, one might instead choose points based on the density function

- As a second attempt, draw initial points from a probability density function defined as $\rho(\boldsymbol{x})^{0.5}$

A second attempt...

Choosing initial generating points according to the PDF given by $\rho(\boldsymbol{x})^{0.5}$ does significantly reduce the generating point movement overall





MPAS
Model for Prediction Across Scales

Incremental grid generation

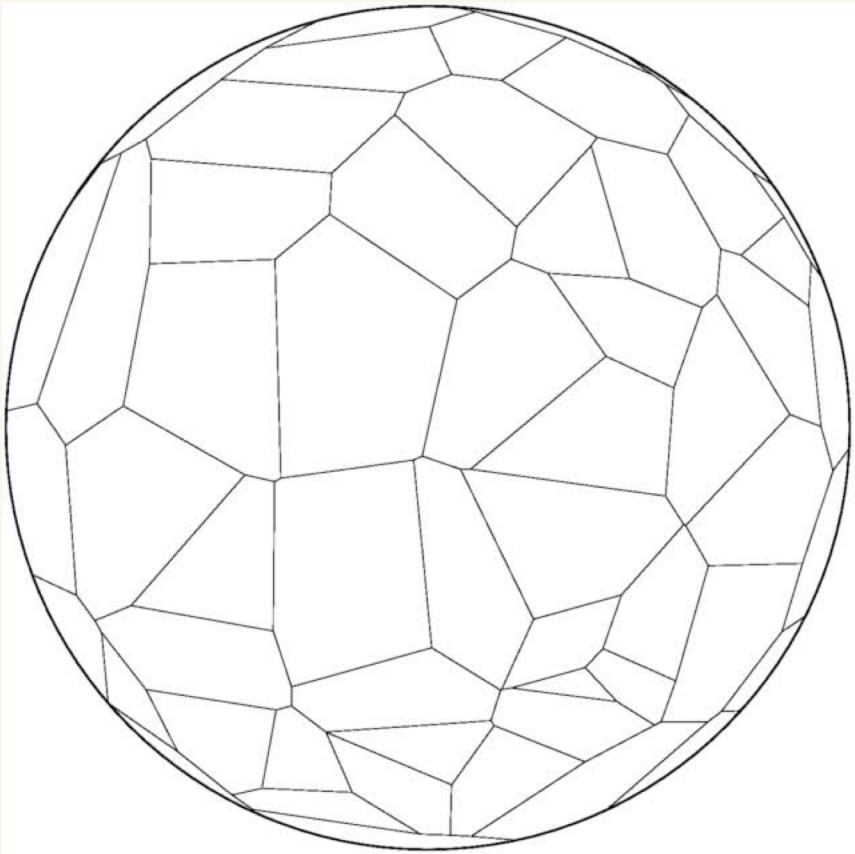
The preceding approaches may be fine for coarse meshes, but convergence can still take quite some time for high-resolution meshes.

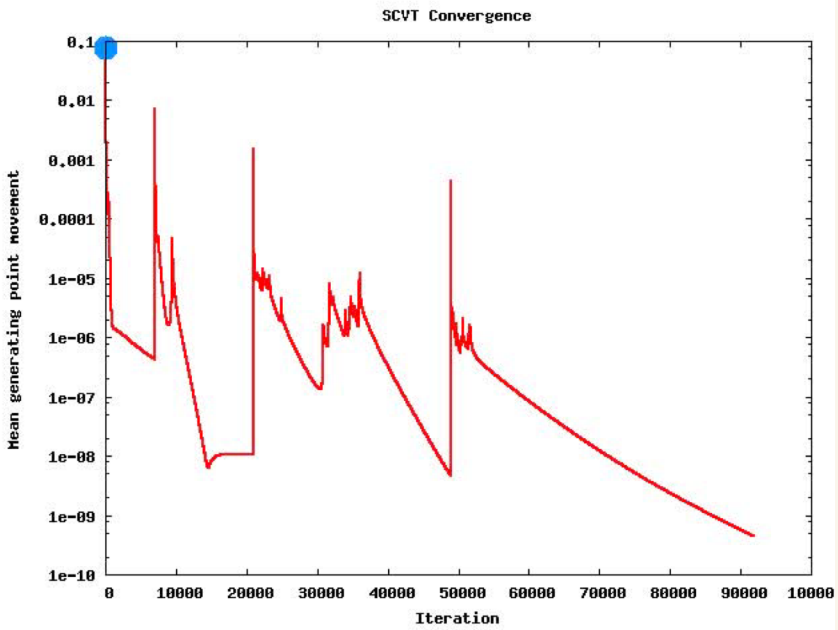
Todd Ringler (LANL) has suggested an approach that leads to meshes in significantly less time, and is what we now employ almost exclusively:

1. Begin with a random set $X_0 = \{\boldsymbol{x}_i\}$, with $|X_0| = O(100)$, of initial generating points suitable for a mesh that is 2^n times coarser in resolution than the target mesh
2. For $k = 0 \ldots n$
    1. Perform Lloyd iterations on X_k to convergence to produce a mesh M_k
    2. If $k < n$, let X_{k+1} be the set of points resulting from the bisection of M_k
3. The mesh M_n is the a CVT with the desired resolution

Incremental mesh generation

Experience indicates that the incremental approach to grid generation does lead to a higher-quality mesh in significantly less wall-clock time





Simple parallel implementation

The incremental generation approach appears to be robust enough, but generating a mesh *is still very slow*

- Can take several months on a single multi-core desktop for high-resolution meshes
- Lloyd iteration has two main computational phases, one of which is embarrassingly parallel

```
while (not converged)
    let {x_i} = {x_i'}
    compute Voronoi regions, {V_i}, of {x_i}
    compute mass centers, {x_i'}, of {V_i}
end
```

Currently computed in serial *in the code we use for MPAS-Atmosphere meshes*

Trivially parallelized, since computation of mass center for a Voronoi region is independent of other regions

MPAS
Model for Prediction Across Scales

Simple parallel implementation

Unless we're parallelizing a part of the Lloyd iteration that takes considerable time, we'll not gain very much

- How much time is spent in the two main phases of *our implementation of* Lloyd's method?

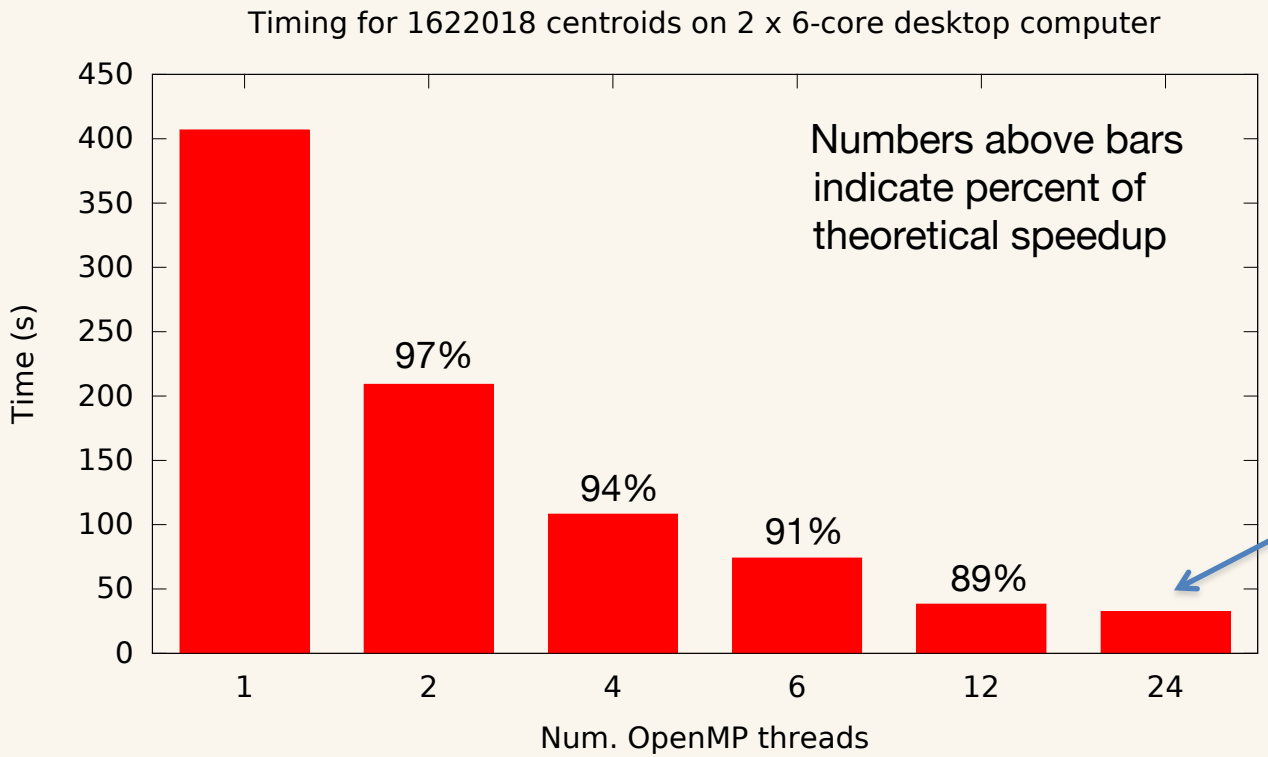
Number of generating points	Centroid computation (seconds)	Voronoi tessellation (seconds)	Ratio of centroid time to Voronoi time
100	0.02688	0.000138	194.8
1000	0.24740	0.001441	171.7
10000	2.462	0.01958	125.7
101378	25.37	0.1478	171.7
405506	101.9	0.8024	127.0
1622018	406.9	3.753	108.4
6488066	1629	23.86	68.29

"Toy" meshes with random initial generating points

Refinement stages from a 15-3 km mesh

Simple parallel implementation

With the addition of a single OpenMP parallel loop, we're able to gain a very significant improvement in performance



Using SMT reduces runtime by a further 15% compared with 12 OpenMP threads

From Amdahl's Law, we achieve an overall speedup of 11.3 on a 2x6-core Xeon desktop!

MPAS
Model for Prediction Across Scales

Distributed-memory parallel algorithm

Parallelization of centroid computation with OpenMP is nearly trivial, but limits scaling to the number of cores in a shared-memory node

- Even with 24 threads on the desktop in my office, generating a 15-3 km mesh with ~6.5 M cells still took *months*!

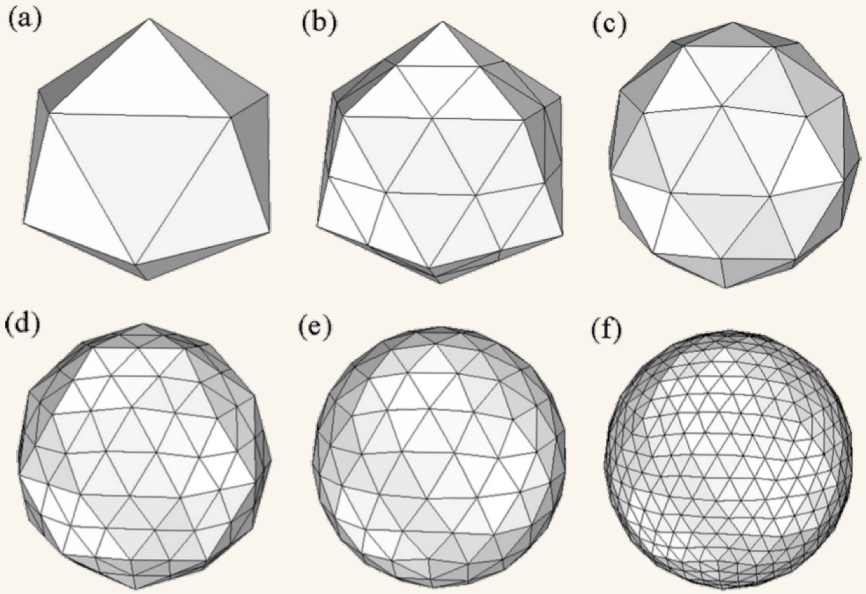
Jacobsen et al.[6] describe a more scalable approach with both phases of Lloyd's method parallelized with MPI

- Parallelize Voronoi tessellation using stereographic projections
- This approach seems better for larger meshes in particular

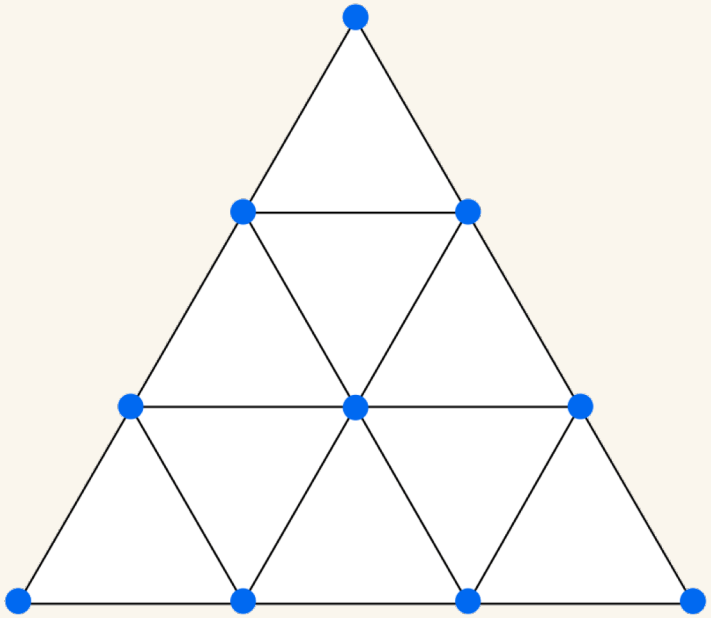
Quasi-uniform meshes

In contrast to variable-resolution meshes, quasi-uniform meshes are comparatively simple!

- Grid distance in original icosahedron = 7530 km (for earth radius 6371 km)
- Divide triangles by any integral factor, and let vertices of triangles be generating points after projection to the sphere



Bisecting an icosahedral mesh and projecting vertices to the sphere. Lipscomb and Ringler (2005)



For quasi-uniform MPAS meshes, we may divide by any positive integer factor.

MPAS
Model for Prediction Across Scales

Summary and future work

Grid generation at present is a slow process that often requires human judgment in, e.g., knowing when a stage of mesh generation is converged "enough", working around stable but undesirable mesh features, etc.

- Before this is supportable to community, we need to make mesh generation more user-friendly
  - Integrated grid metrics are needed to tell us:
    - When to stop iterating
    - Whether we can expect good numerical properties from resulting mesh
- Improvements to performance are still needed
  - Can techniques like over-relaxation be applied to improve convergence rate?
  - Implement code optimizations, exploit GPUs for embarrassingly parallel work, etc. to improve iteration rate

References

[1] Du, Q., V. Faber, and M. Gunzburger, 1999. "Centroidal Voronoi Tessellations: Applications and Algorithms". *SIAM Review,* **41**, 637 – 676.

[2] Newman, D. J., 1982. "The Hexagon Theorem". *IEEE Transactions on Information Theory*, **28**, 137 – 139.

[3] Gersho, A., 1979. "Asymptotically Optimal Block Quantization". *IEEE Transactions on Information Theory*, **25**, 373 – 380.

[4] "Voronoi Diagrams and their Application in Climate and Global Modeling", Chapter in *Numerical Techniques for Global Atmospheric Models*. Springer-Verlag Berlin Heidelberg, 2011.

[5] Ringler, T. D., D. Jacobsen, M. Gunzburger, L. Ju, M. Duda, and W. Skamarock, 2011. "Exploring a Multiresolution Modeling Approach within the Shallow-Water Equations". *Mon. Wea. Rev.,* **139**, 3348 – 3368.

[6] Jacobsen, D.W., M. Gunzburger, T. Ringler, J. Burkardt, and J. Peterson, 2013. "Parallel algorithms for planar and spherical Delaunay construction with an application to centroidal Voronoi tessellations". *Geosci. Model Dev.,* **6**, 1353 – 1365.