Mesh Decomposition for Parallel Unstructured Mesh Generation

David Marcum

Outline

- AFLR motivation and basic algorithm.
- Need for parallel operation.
- Sub-domain decomposition.
- Mesh generation process with sub-domains.
- Initial results & future directions.

International Chair

- What type of chair is it?

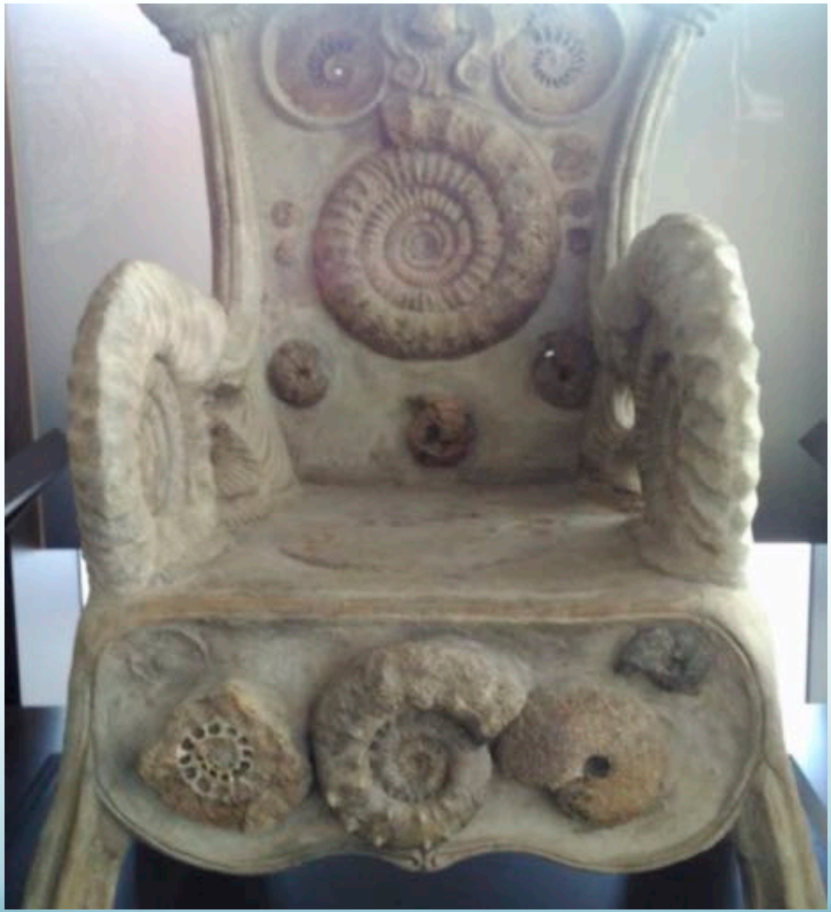
International Chair

- What type of chair is it?

gamma
INVENTEURS DE LA BONNE MAILLE

International Chair

- What type of chair is it?



International Chair

- What type of chair is it?

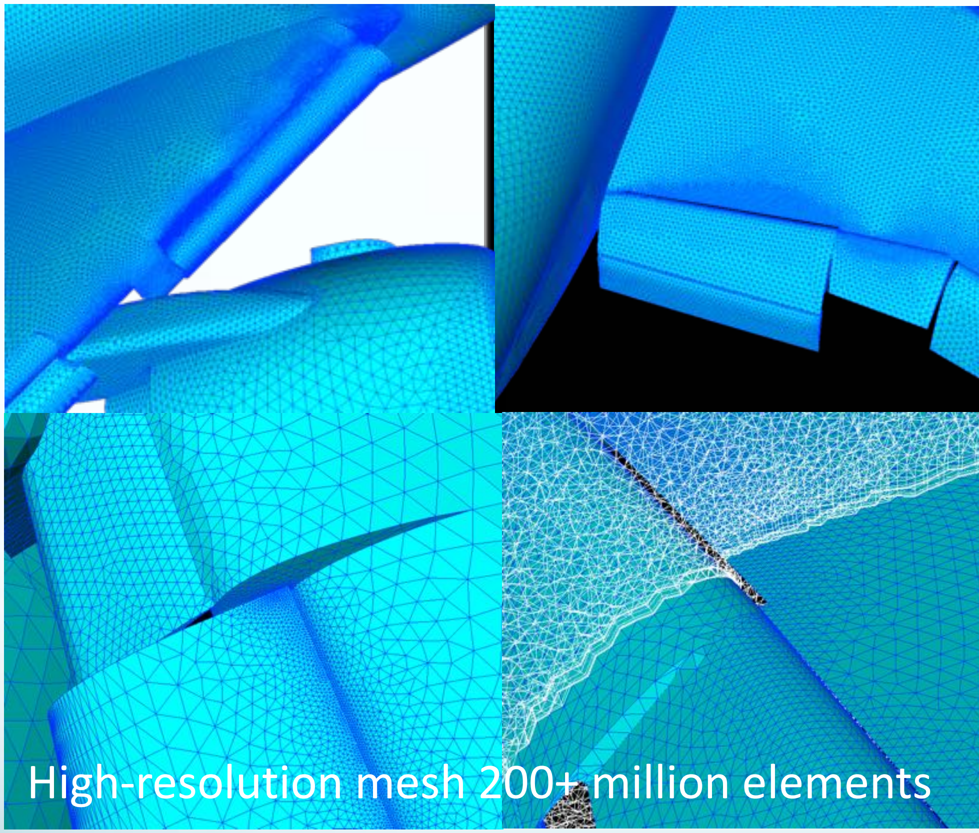
International Chair

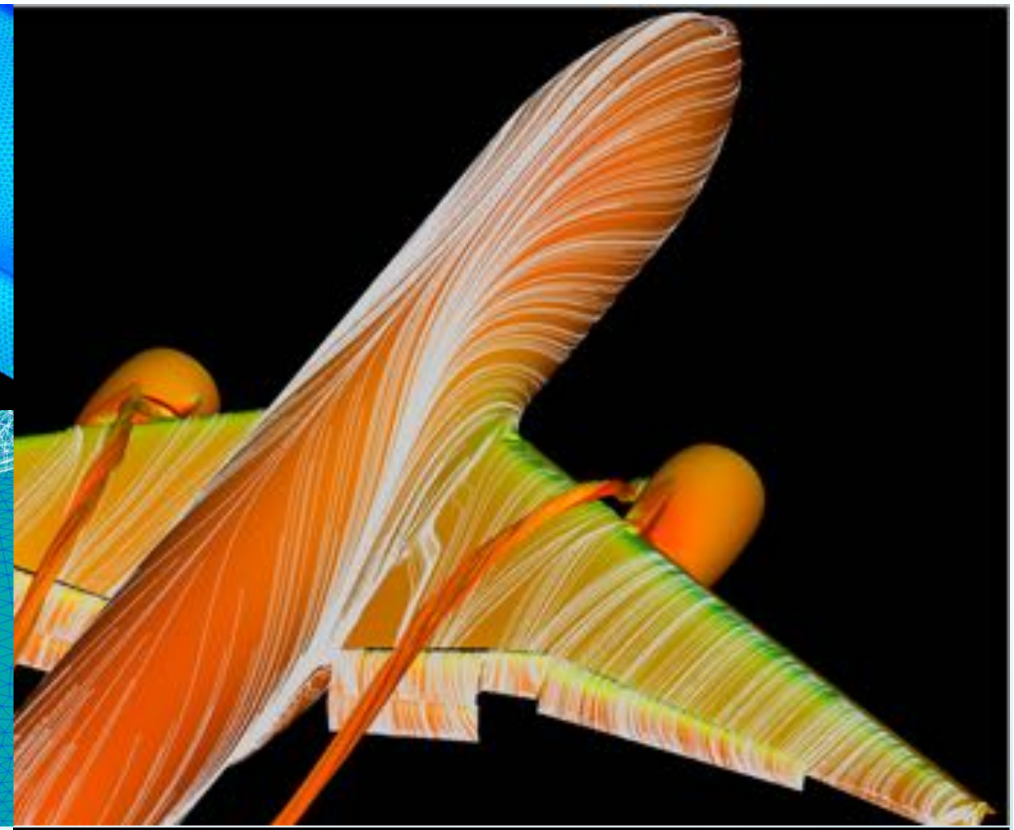
- What type of chair is it?

International Chair

- What type of chair is it?

Motivation: "High-End" CFD Applications (Commercial Aircraft)

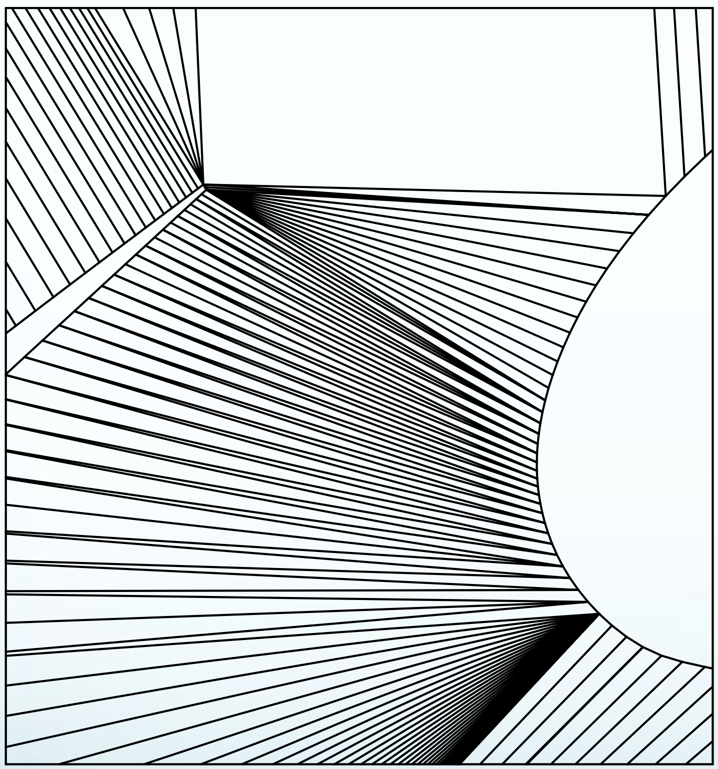




Some characteristics:
- Large variation in length scales
- Thick BL regions
- Precise BL growth
- Small gaps
- Anisotropic surface meshes
- Small details often crucial to predicting key flow physics

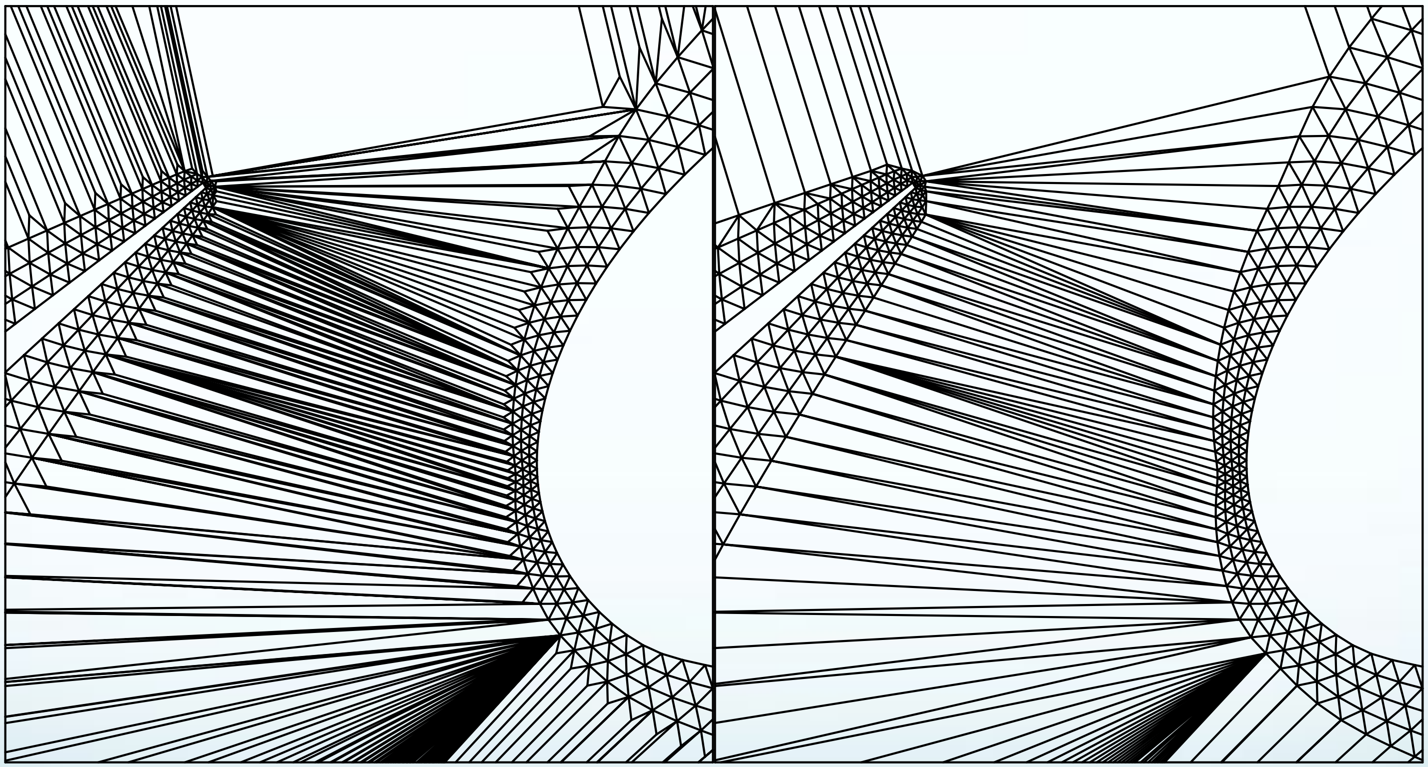
Simulation Courtesy of *The Boeing Company*
CFD Solver: Metacomp Technologies *CFD++*
Mesh Generation Framework: Boeing *MADCAP*
Mesh Generation: MSU *AFLR* surface and volume mesh generation

Initial Triangulation



- Outer region uses AFLR based mesh generation.
- Starting point is an initial triangulation of the boundary.
- Process here is shown in 2D with purely isotropic elements.

Point Insertion & Connectivity Optimization



Need for Parallel Mesh Generation

- Realistic configurations require an ever increasing level of resolution that scales with computational resources available. Hundreds of millions of elements is production level work in some cases.
- Far from ultimate level of resolution in many applications (turbulent flow, etc.)
- Mesh generation in serial becomes the bottleneck in the process in terms of time required.
- Inclusion of mesh generation within the solver process for solution adaptation or geometry that changes require that the mesh processing be done as a scalable process also.
- Need a truly scalable mesh generation process for current and future use.

Scalable Parallelization

- A single strategy for scalable and optimal parallel and vector operations is not possible. The local operations produce global changes and the mesh itself is dynamically evolving data.

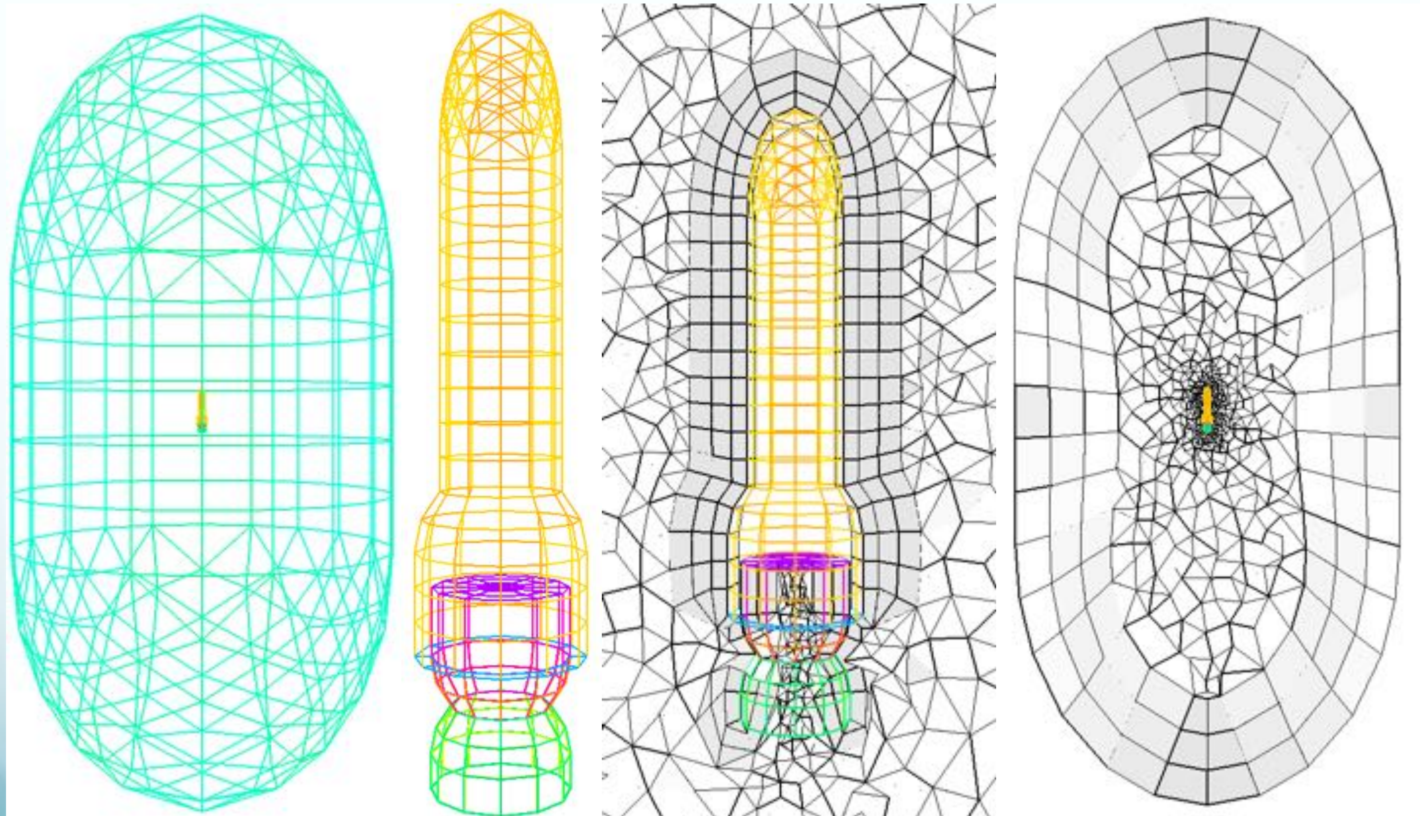
- Multiple approaches and strategies are being considered in this work to eventually produce a truly scalable methodology.
  - Uniform core mesh for large regions of similar sized elements. This is very fast (1G elements in a couple minutes of serial for total mesh). It is also readily parallelized and scalable. Limited to regions of similar sized elements and very specific applications.
  - Decomposition into subdomains for independent meshing for macro scale parallelization.
    - Unstructured coarse mesh decomposition into subdomains with true internal boundaries.
    - Iterative partitioning of subdomains throughout the process.
    - **Oct-tree like decomposition with *virtual* or *pseudo-constrained* sub-domain boundaries.**
    - Goal is a decomposition that produces no artifacts and requires no direct inter-processor communication. Each sub-domain is intended to be processed independently.
  - Fine scale parallelization of individual local processes within each subdomain.

Scalable Parallelization: BL Issues

- The strategy proposed for basic mesh generation is suitable for both isotropic and anisotropic metric driven mesh generation of the outer mesh.
- BL portion is currently generated by a differing process. It also scales with the boundary surface mesh which is a known entity at the time or of processing. Decomposition based on the surface mesh seems more appropriate in this case. Basic multi-level parallel approach still applies.
- Current work with anisotropic aligned metric driven mesh generation shows promise for a more universal and unified approach.

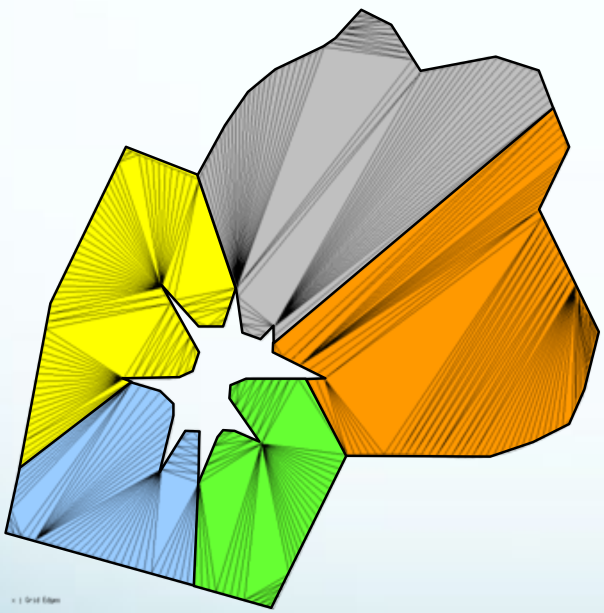
Domain Decomposition Strategies (1)

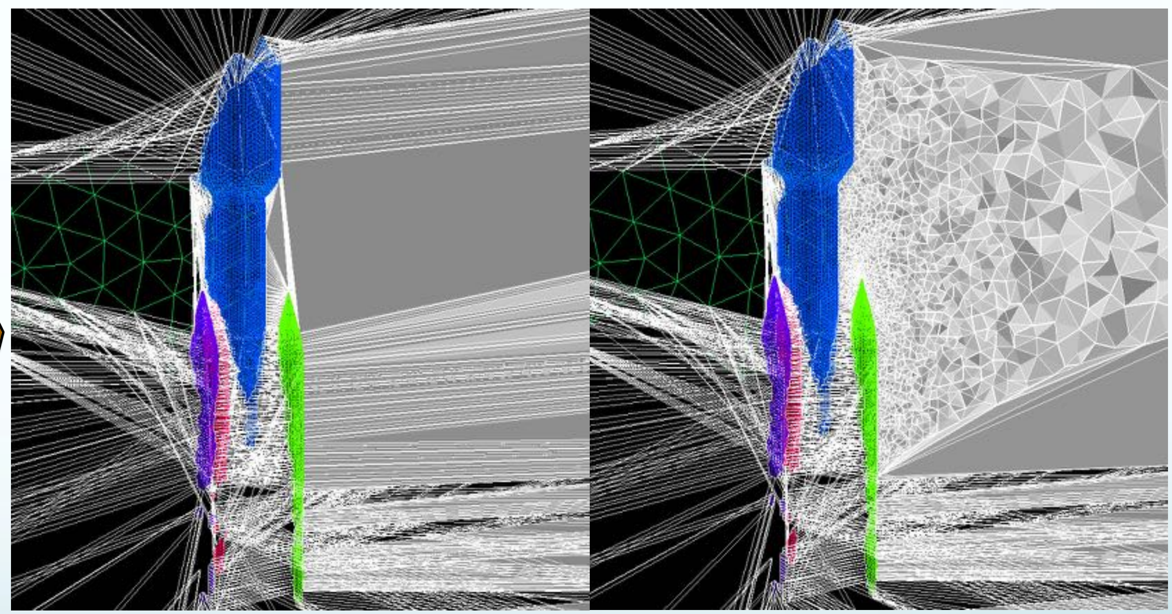
- **Coarse Mesh Decomposition:** Issues include problems in arriving at a suitable coarse mesh, internal surface mesh generation (overhead), and limited scalability. ***REJECTED.***



Domain Decomposition Strategies (2)

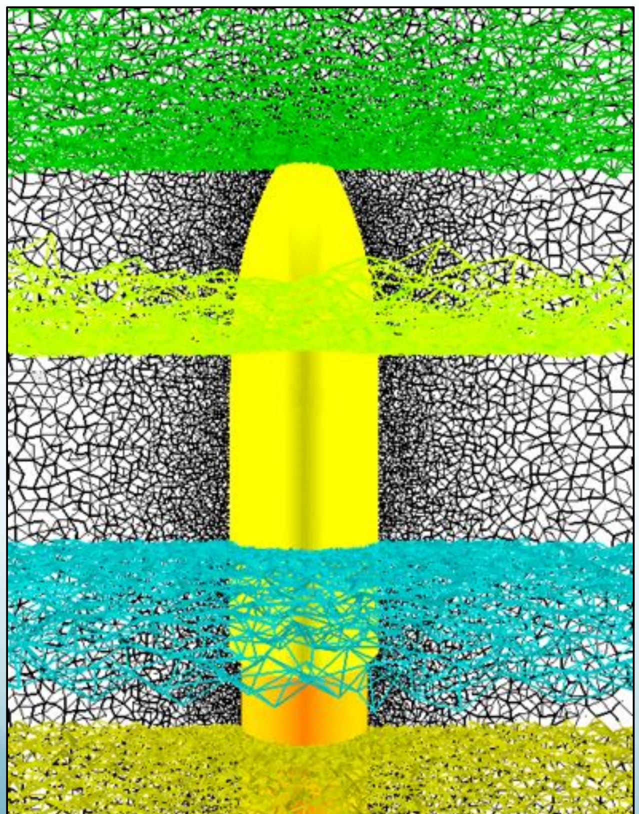
- **Iterative Partitioning:** Issues include requirement of a full-domain initial mesh and significant communication when repartitioning. ***REJECTED.***

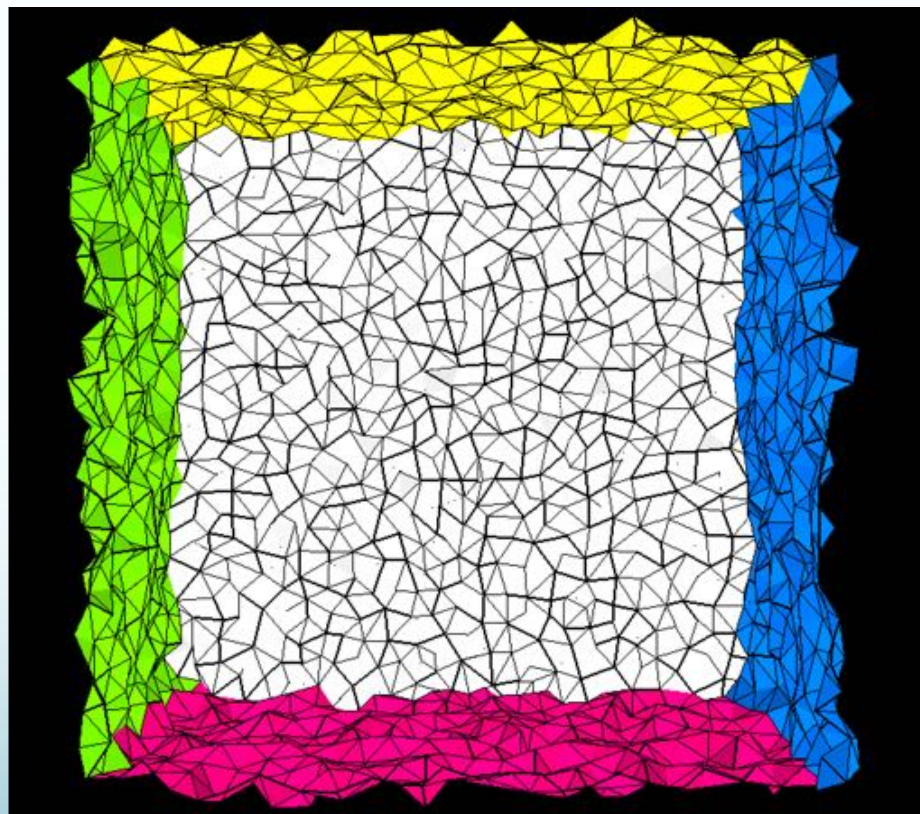




Domain Decomposition Strategies (3)

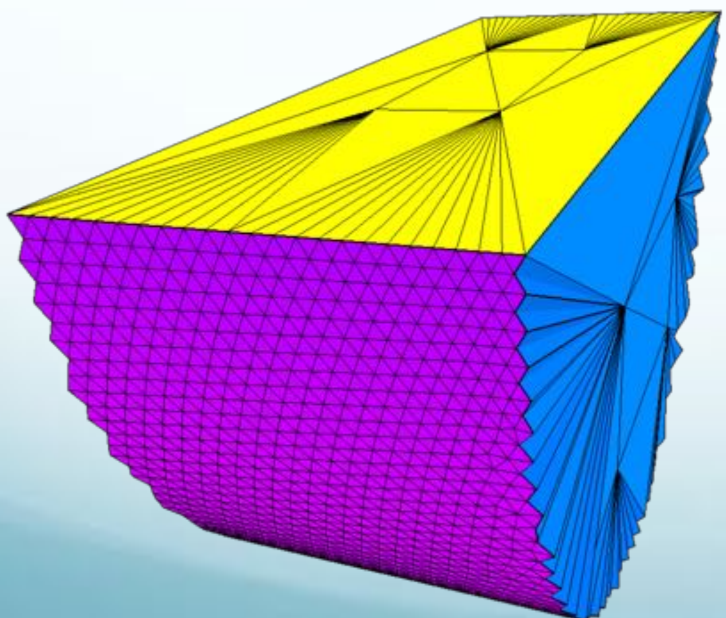
- **Virtual Domain Decomposition:** Use simple Cartesian/oct-tree like virtual decomposition. Virtual boundaries must be extracted, transferred, and combined with others. ***UNDER DEVELOPMENT.***

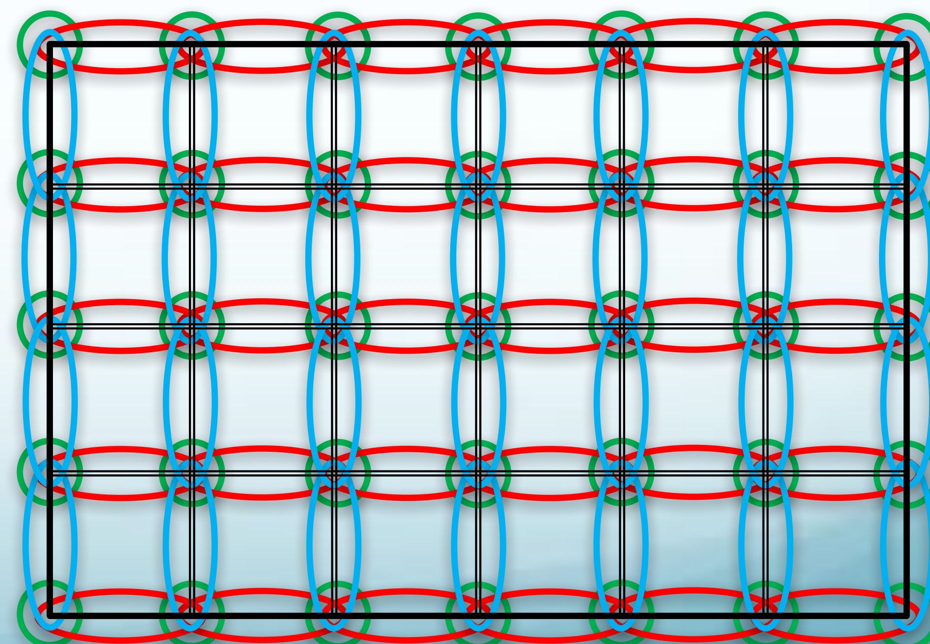




Domain Decomposition Strategies (4)

- **Virtual Domain Decomposition with Pseudo-Constrained Sub-Domains:** Use simple Cartesian/oct-tree like virtual decomposition. No coloring needed. Sub-domains temporarily constrained with corner points and triangulated surfaces. Issues with virtual boundaries are significantly minimized (single surface extraction). No coloring needed. ***UNDER DEVELOPMENT.***





Automated Domain Decomposition

- A simple Cartesian decomposition was considered for the overall decomposition. In this approach the decomposition itself is trivial.
- The domain is over decomposed for load-balancing and the minimum number of sub-domains required can be determined for a given number of cores.
- A scheduler is required to load-balance and launch new processes as old ones complete (collaborative with ODU).
- AFLR mesh generation operates independently on each sub-domain of the decomposition.
- Modifications were required within AFLR to implement this approach.
  - Boundary-conditions and boundary surface recovery were modified to allow *virtual* boundaries (those between interior sub-domains).
  - Boundary-conditions also modified for *pseudo-constrained* sub-domain boundaries.
  - Sub-domain seeding was added to allow generation of volume elements for domains without true boundary surface portions. With advancing method there must be something to advance from.

Individual Sub-Domain Processing

- Each individual sub-domain is comprised of *virtual* or *pseudo-constrained* sub-domain boundaries along with true boundary surface portions in some sub-domains.
- Since our mesh generation algorithm relies on frontal advancement we need to add more fronts.
- Seeding with element clusters was chosen to provide the additional fronts.
  - Seed clusters do not produce artifacts like a surface would.
  - Sizing based on actual local length scales (or metric if used) from the background mesh.
  - Essential if there are no true boundary surfaces in sub-domain.
  - Used in all sub-domain cases as portion of true boundary may be very small.
- Result is a process that behaves as in standard mode.

Sub-Domain Coloring

- Only used with *virtual* boundary method.
- A coloring process is used to sort the sub-domains.
- A simple coloring is used to illustrate the process.
- Consider a domain that is enclosed within a Cartesian sub-domain mesh with each cell or sub-domain of size Δx, Δy, and Δz. This can be represented topologically as an i, j, and k domain, where i represents variation in x only, j represents variation in y only, and k represents variation in z only. On the first pass every other sub-domain within a given i-row (constant j and k) is available for processing independently. The i-row above each is skipped as well as the entire adjacent k-plane (constant k). This pattern is continued until all the sub-domains are accounted for.

Sub-Domain Coloring (continued)

- The numbers shown correspond to each pass. There are a total of 8 passes required with this form of coloring, the first four for the odd k-planes and the next four the even ones. Pass 1 sub-domains must pass their right and left boundaries to Pass 2 sub-domains, their top and bottom boundaries to Pass 3 sub-domains, and their front and back boundaries to Pass 5 sub-domains.

- Coloring on odd k-planes 1,3,5…

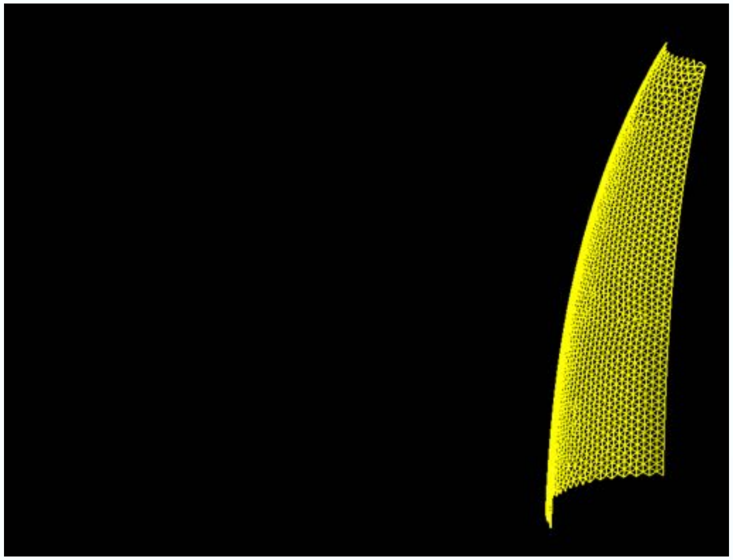
1	2	1	2	1	2	1	2	1
3	4	3	4	3	4	3	4	3
1	2	1	2	1	2	1	2	1
3	4	3	4	3	4	3	4	3
1	2	1	2	1	2	1	2	1
3	4	3	4	3	4	3	4	3
1	2	1	2	1	2	1	2	1
3	4	3	4	3	4	3	4	3
1	2	1	2	1	2	1	2	1

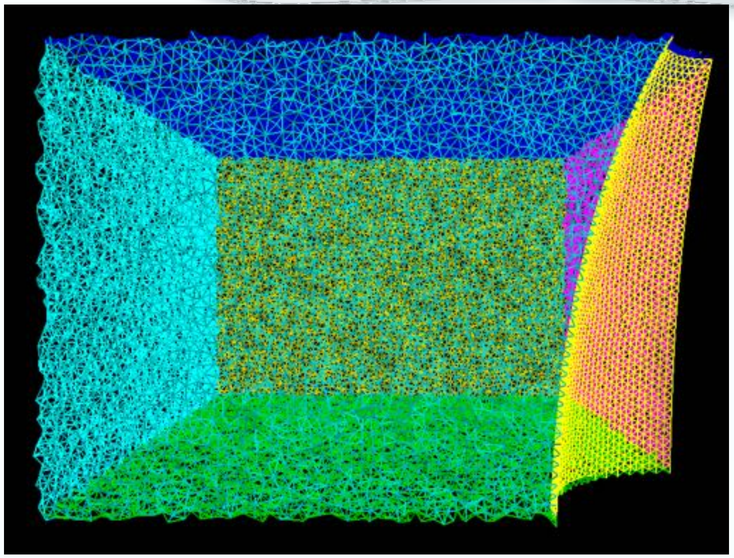
Sub-Domain Coloring (continued)

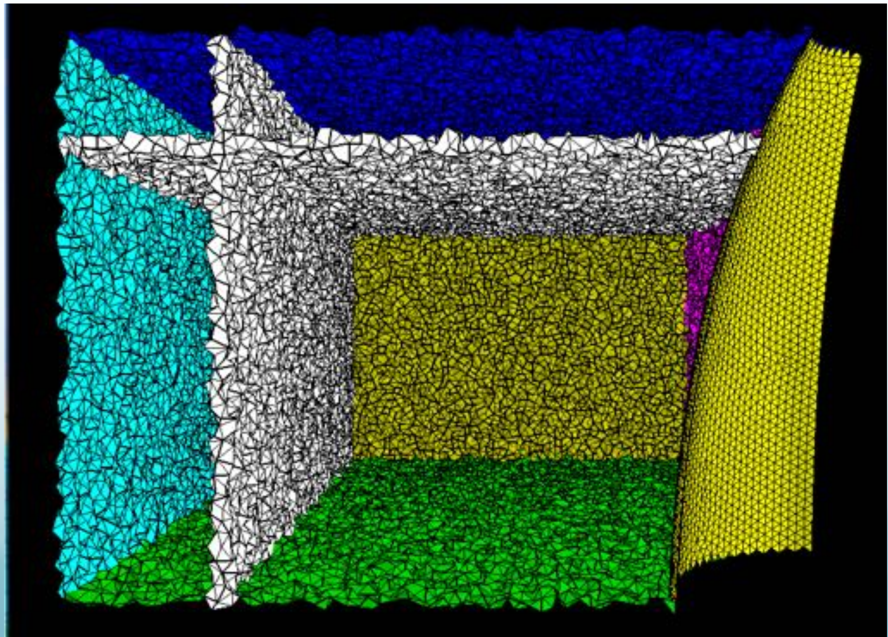
- Coloring on even k-planes 2,4,6…

5	6	5	6	5	6	5	6	5
7	8	7	8	7	8	7	8	7
5	6	5	6	5	2	5	6	5
7	8	7	8	7	8	7	8	7
5	6	5	6	5	6	5	6	5
7	8	7	8	7	8	7	8	7
5	6	5	6	5	6	5	6	5
7	8	7	8	7	8	7	8	7
5	6	5	6	5	6	5	6	5

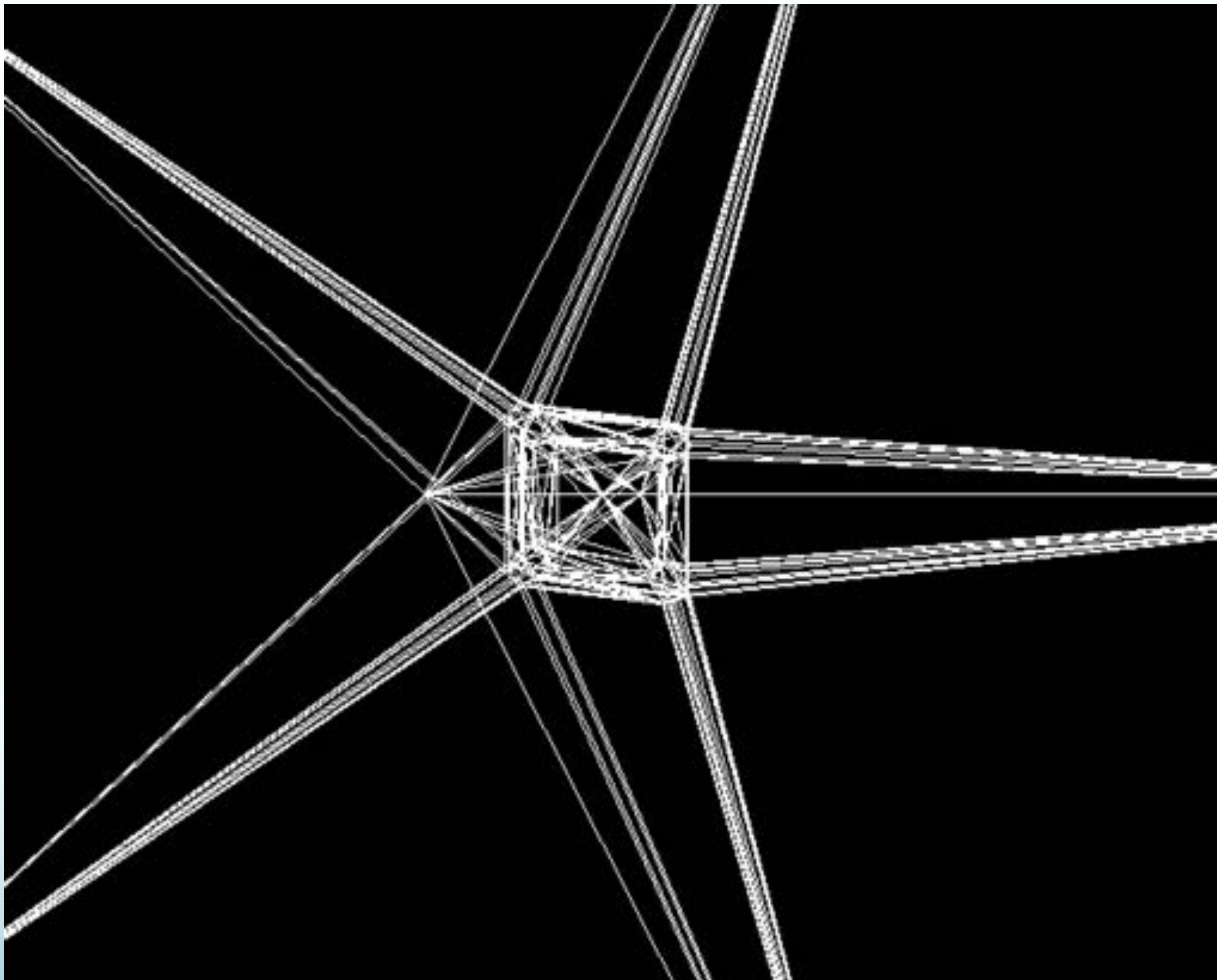
Individual Sub-Domains With a Boundary Portion



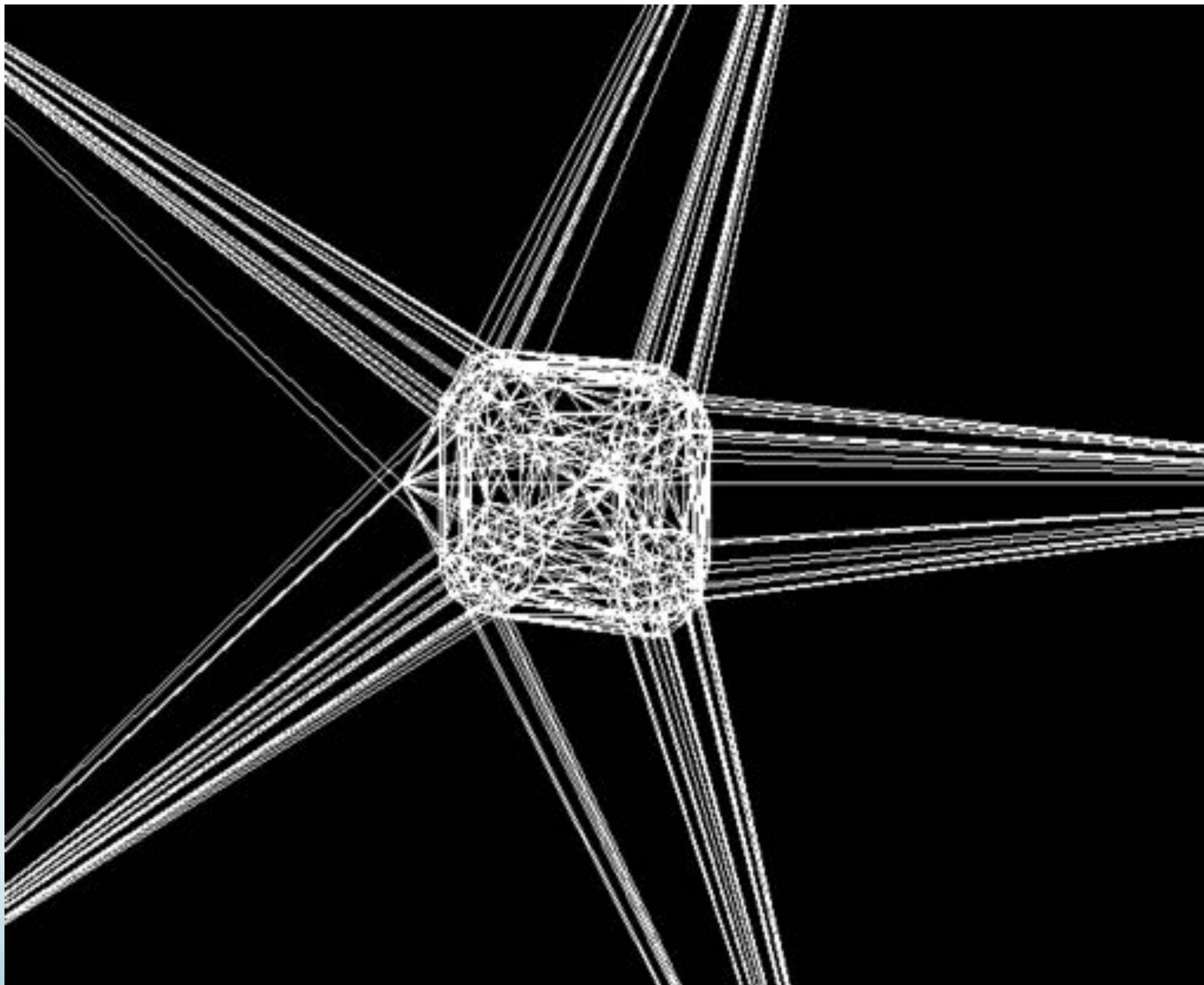




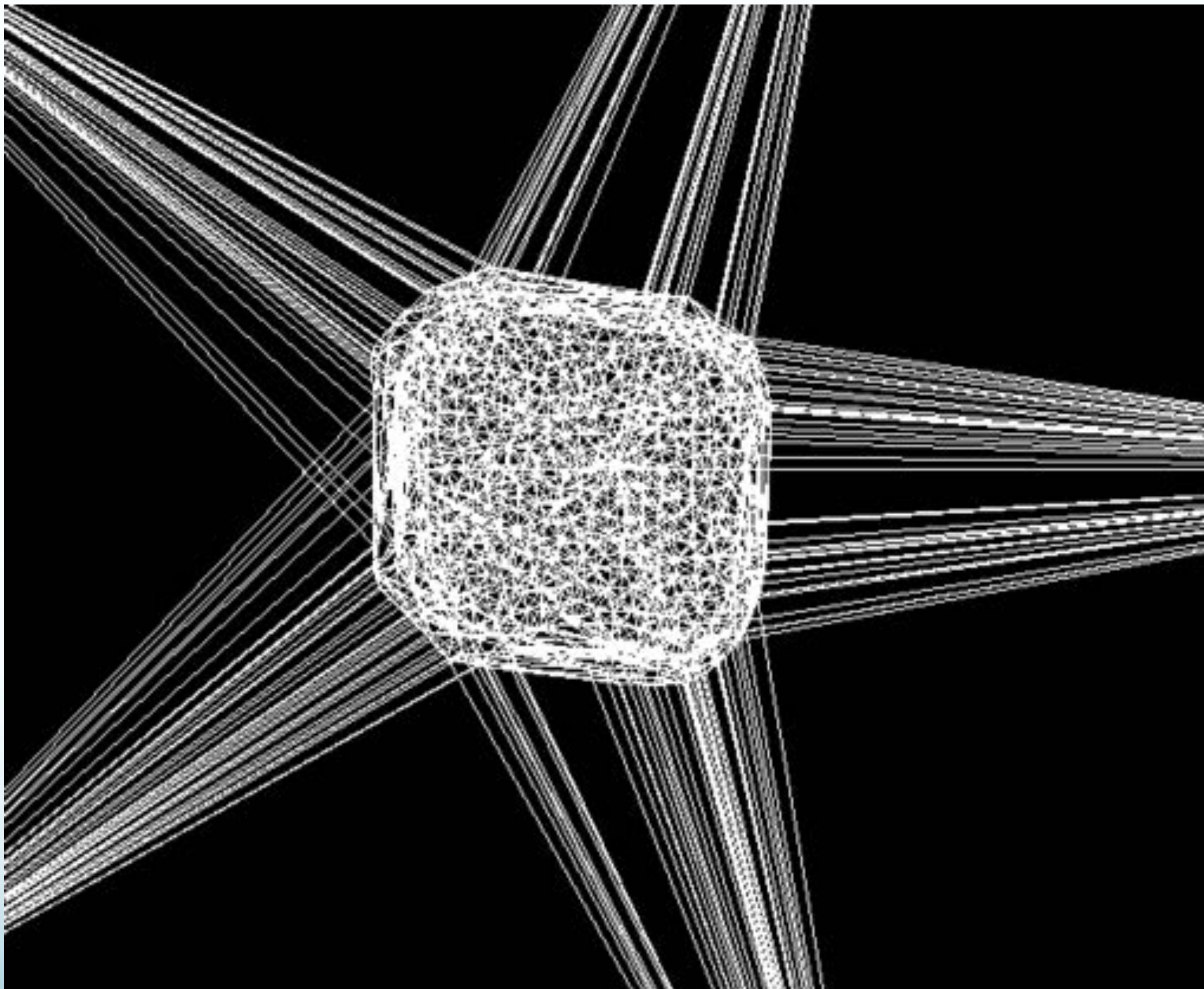
Seeding for Sub-Domains Without any True Boundaries (start)



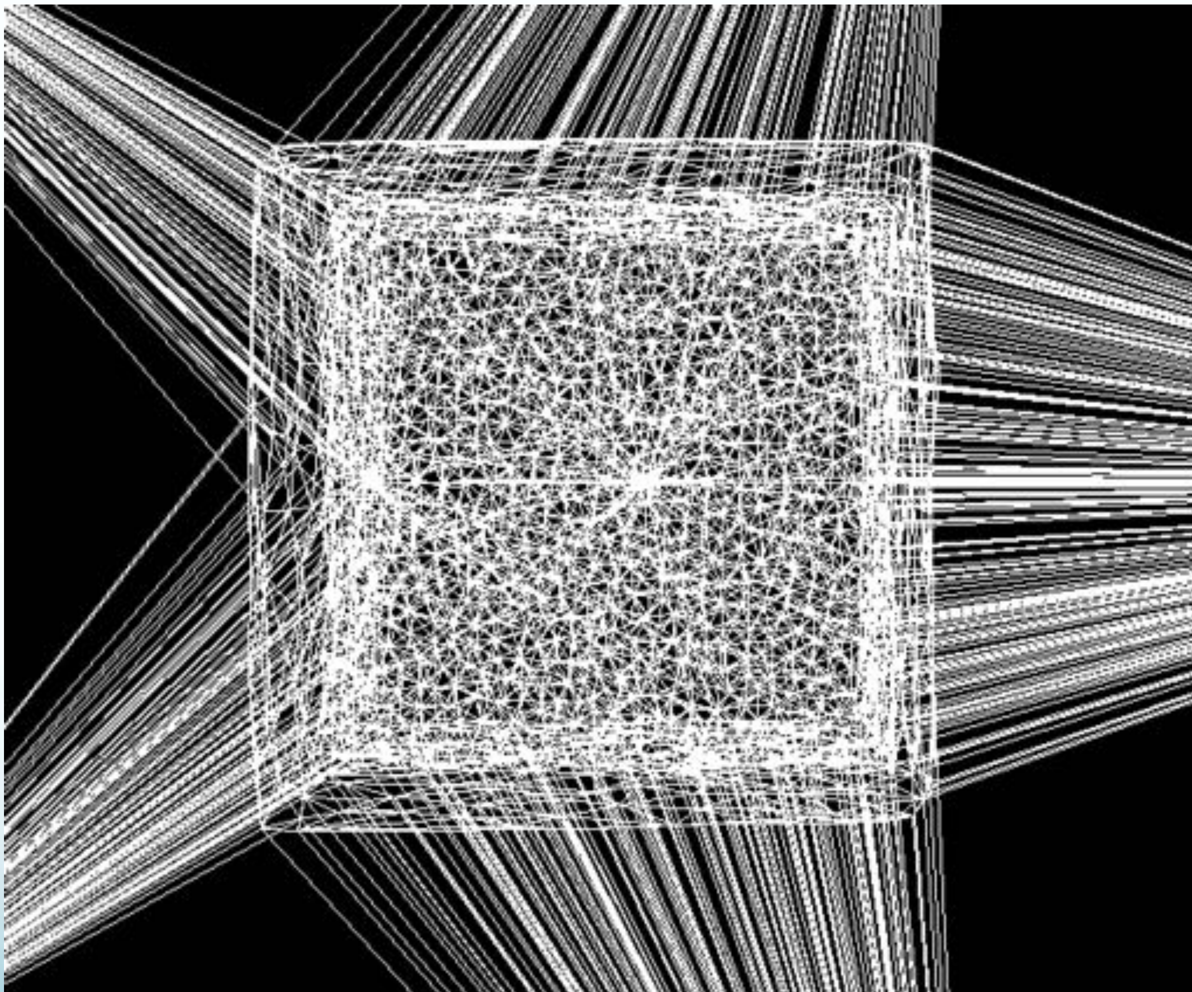
Advancement From Seeds



Advancement From Seeds (5 passes)



Advancement From Seeds (final)



Trimming and Classification of *Virtual* Sub-Domain Boundaries

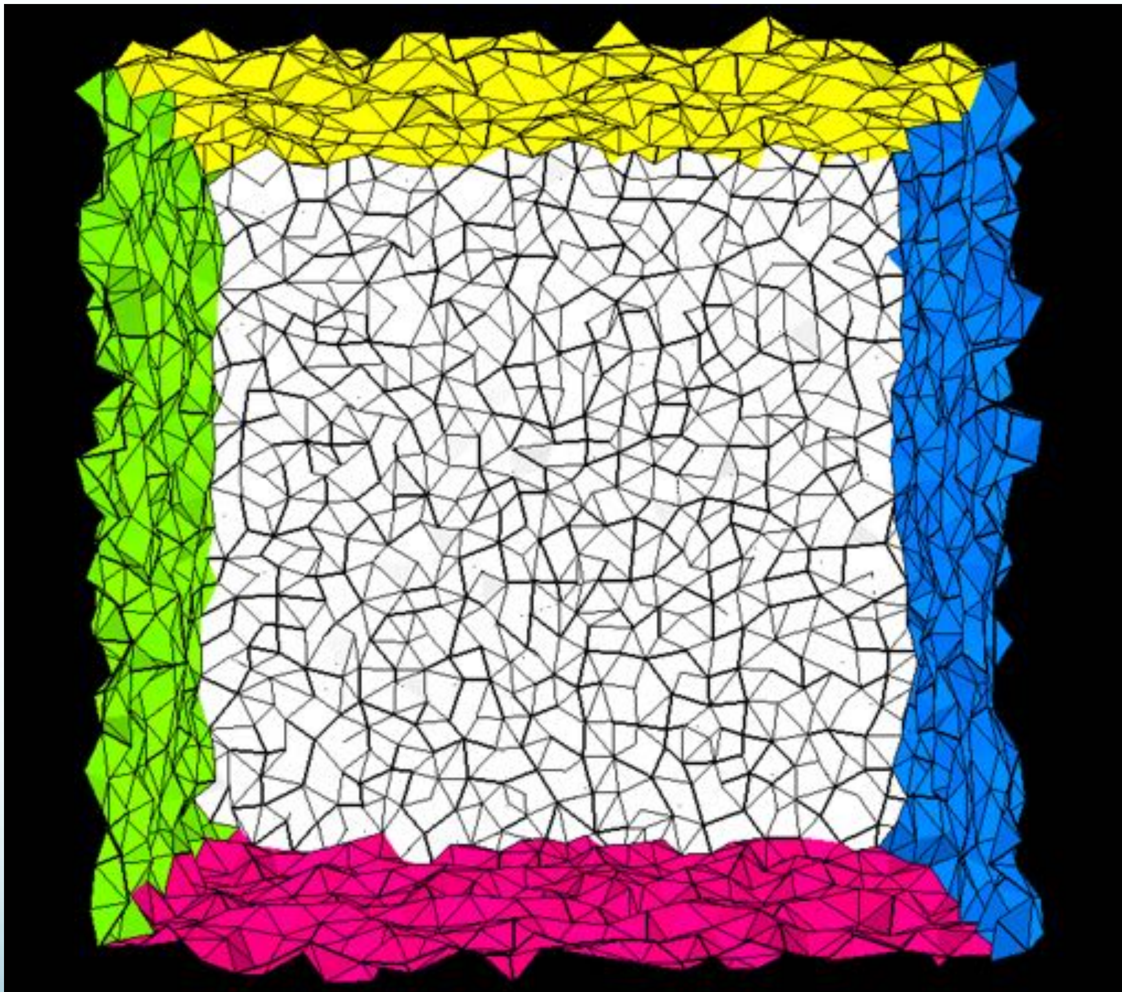
- Mesh generation is allowed to advance past the *virtual* sub-domain boundaries.
  - Distance is based on local length scale (typical is 2 *x* the individual element size).
  - Allows algorithm to perform optimally in terms of quality, which degrades if the mesh generation is not allowed some room for optimizing with smoothing and local-reconnection.

- Since the domain is not enclosed in true boundaries, the boundary recovery process can not simply delete external elements used to start the method.

- The process is modified by trimming based on the virtual sub-domain boundaries and subsequent classification of which boundary surface an exposed element belongs in.

- Classification is required as virtual boundaries result in surfaces of exposed elements that must be passed to neighboring sub-domains as constraints in subsequent mesh generation.

Final Sub-Domain Mesh After Trimming



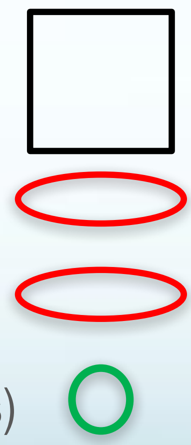
Pseudo-Constrained Approach

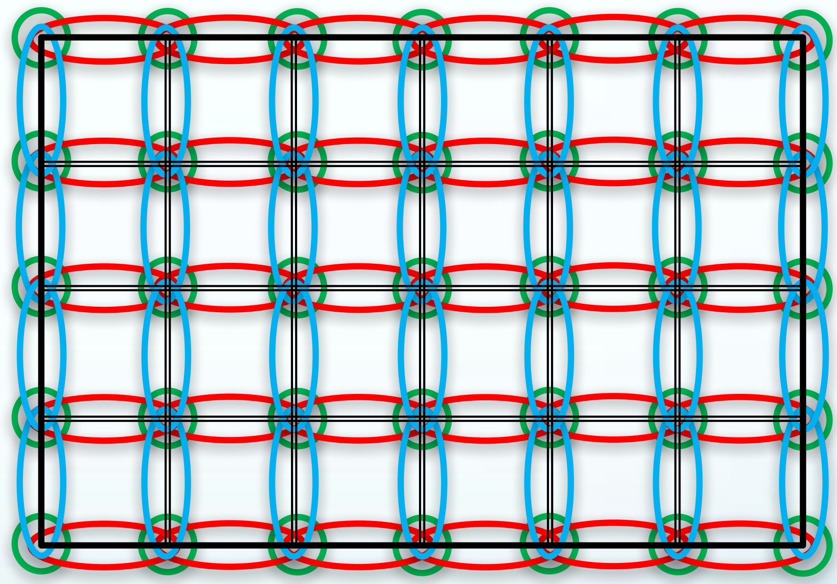
- Previous virtual-constrained method imposed no hard constraints only virtual boundaries and sub-domain interfaces were derived. This led to questions on robustness in forming guaranteed valid connections with neighbors. Pseudo-constrained algorithm addresses this issue. The temporary constraints are removed in subsequent processing on sub-domain interface regions.

- Constraint surfaces are triangulated with additional temporary points (speeds up subsequent mesh generation). The constraint surfaces only serve as a temporary interface that is guaranteed to connect to the neighbor sub-domain. Volume mesh generation within each sub-domain is fully independent and includes a buffer near the constraint surfaces to preserve interior mesh quality. Multiple passes are then added after each sub-domain is processed. Within those passes new sub-domains are created that contain the buffer region near constraint surfaces. Multiple passes allow individual sub-domains of similar numbers to the initial to be created and processed on each pass.

Pseduo-Constrained Sub-Domains

- Use simple Cartesian/oct-tree like virtual decomposition. No coloring needed. Sub-domains temporarily constrained with corner points and triangulated surfaces.
- No coloring needed.
- Minimum of 4 passes required.
  - Pass 1 (regions)
  - Pass 2 (surfaces)
  - Pass 3 (edges)
  - Pass 4 (corner points)



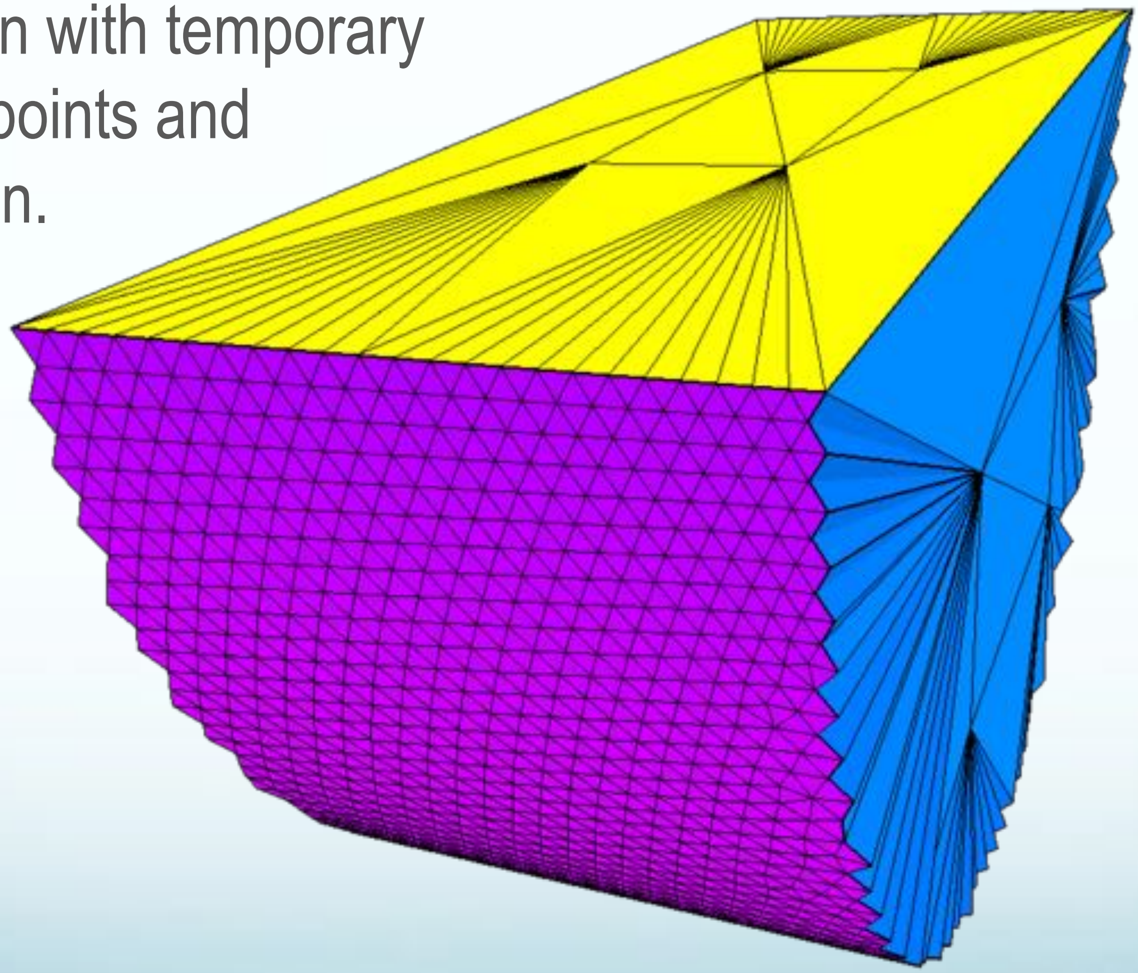


Pseduo-Constrained Sub-Domains Processing

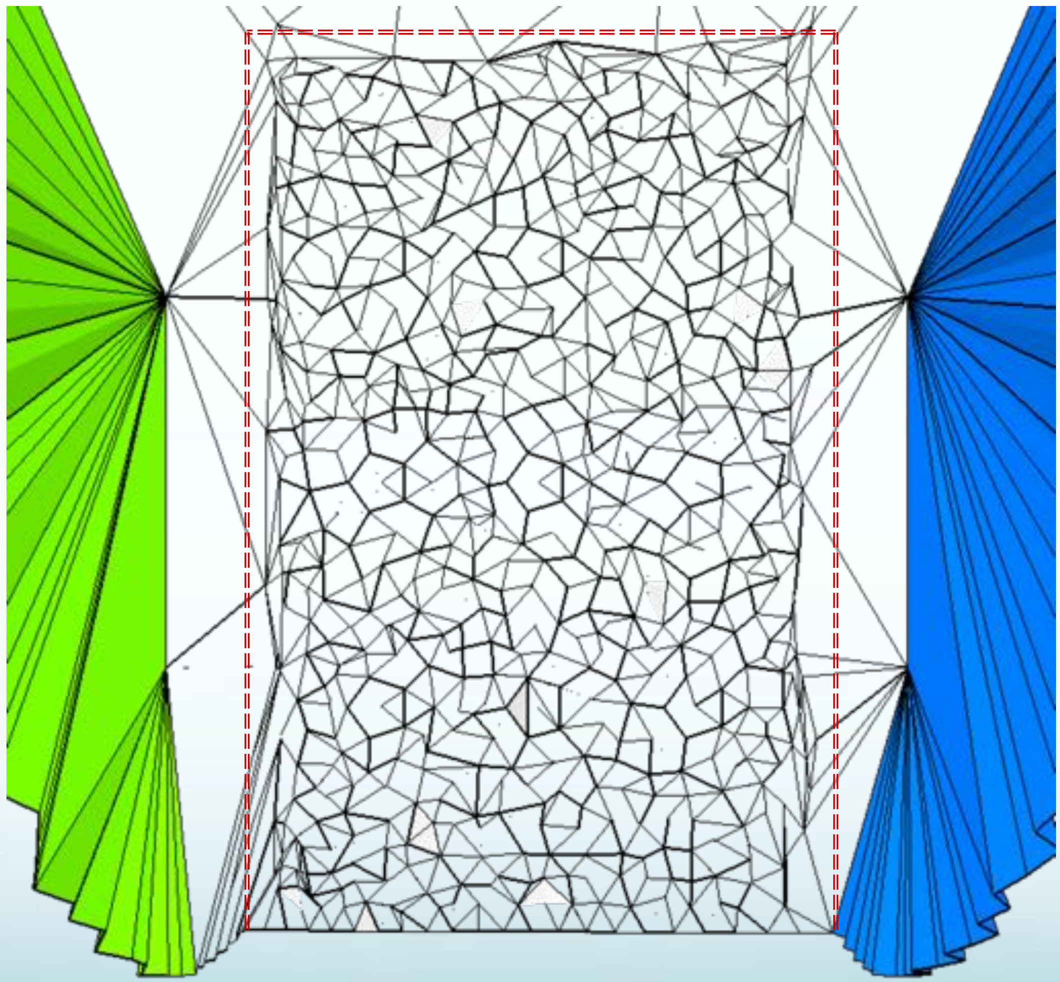
- Mesh generation allowed to advance close to the temporarily constrained sub-domain boundaries.
  - Distance is based on local length scale (typicaly 0.7~1.5 *x* local length scale).
- Since the domain is enclosed in true boundaries, the boundary recovery process is as normal.
- After sub-domains are combined the adjacent regions must be refined. A single closed surface of fully refined faces is extracted around each in at least 3 additional passes (surface, edge and corner point regions).
- Inner boundary surface extraction is robust and rigorous as it is constructed from a fully valid volume (and not pieced together).
- Pass 2, 3, 4, … use sub-comains with true boundary surface constraints from the generated volume mesh.

Pseudo-Constrained Sub-Domain Mesh
Initial Boundary Surfaces

Sub-domain with temporary constraint points and triangulation.



Pseudo-Constrained Sub-Domain Mesh after Pass 1



Automated Domain Decomposition, Initial Results

- Typical single processor run times of about 5 hours or more for 150-200 M elements (isotropic) can be reduced to 5-10 min on 20 processors. Note that the underlying algorithms have an $\vartheta(N^{1.x})$ work effort component so efficiency can appear >100% (and it certainly isn't).

- Overhead of sub-domain creation, extraction and merge is order 3-5% in current form with file based transfer of information.

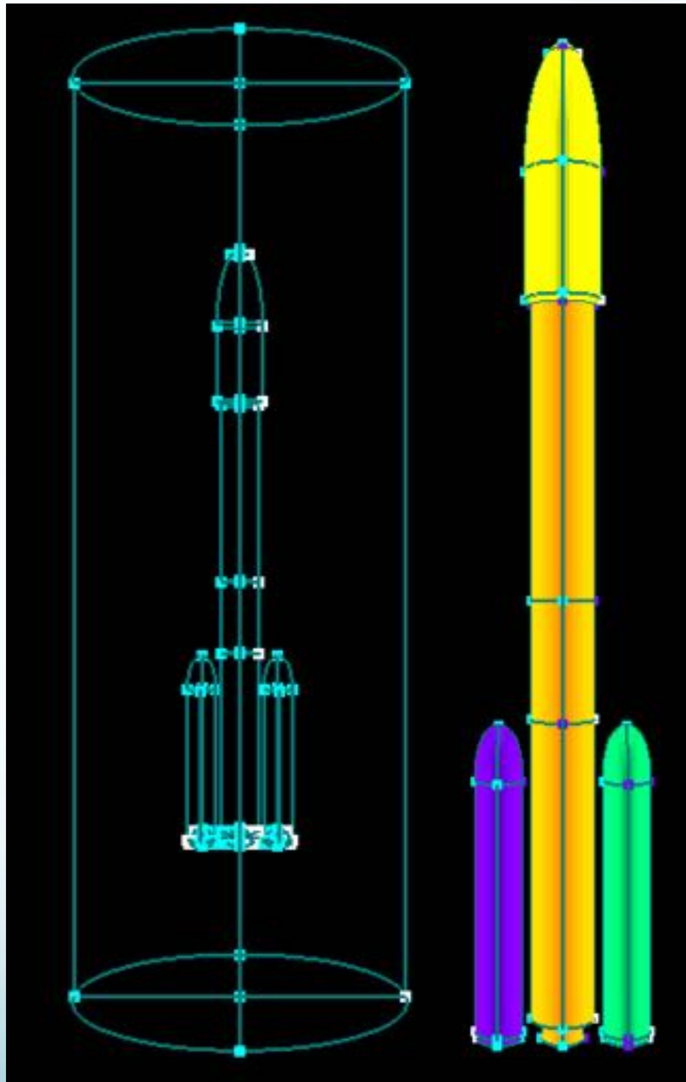
- Load-balancing and/or sub-domain refinement is required to maintain parallel efficiency.

- The present results indicate for large meshes, 100's to even 1000's of processors may be able to be used effectively.

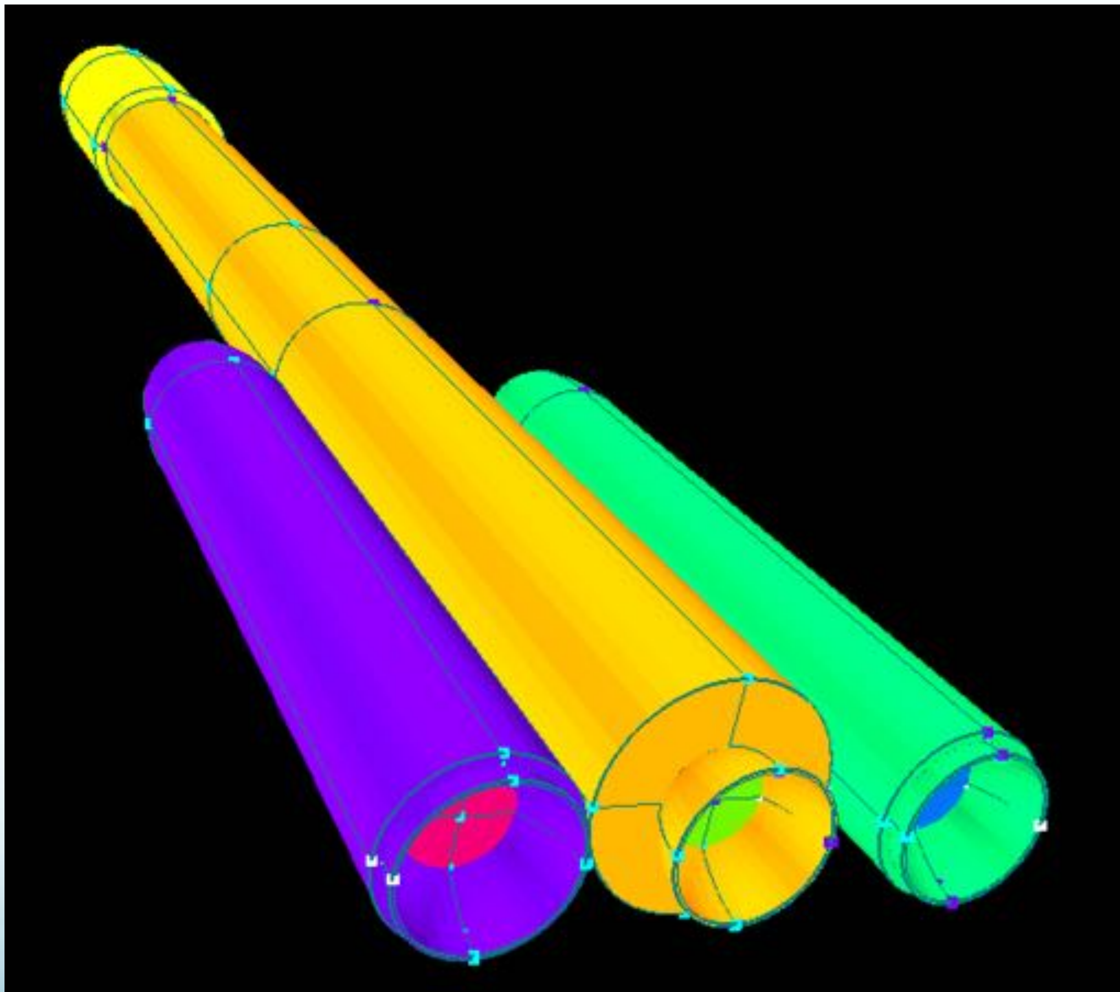
- Further, there are no artifacts from the sub-domains. Artifacts do not occur as there are no imposed internal boundary constraints that are fixed and the element size follows the same methodology (background mesh) as serial mode.

- There are no discernable differences either visually or through mesh quality statistics.

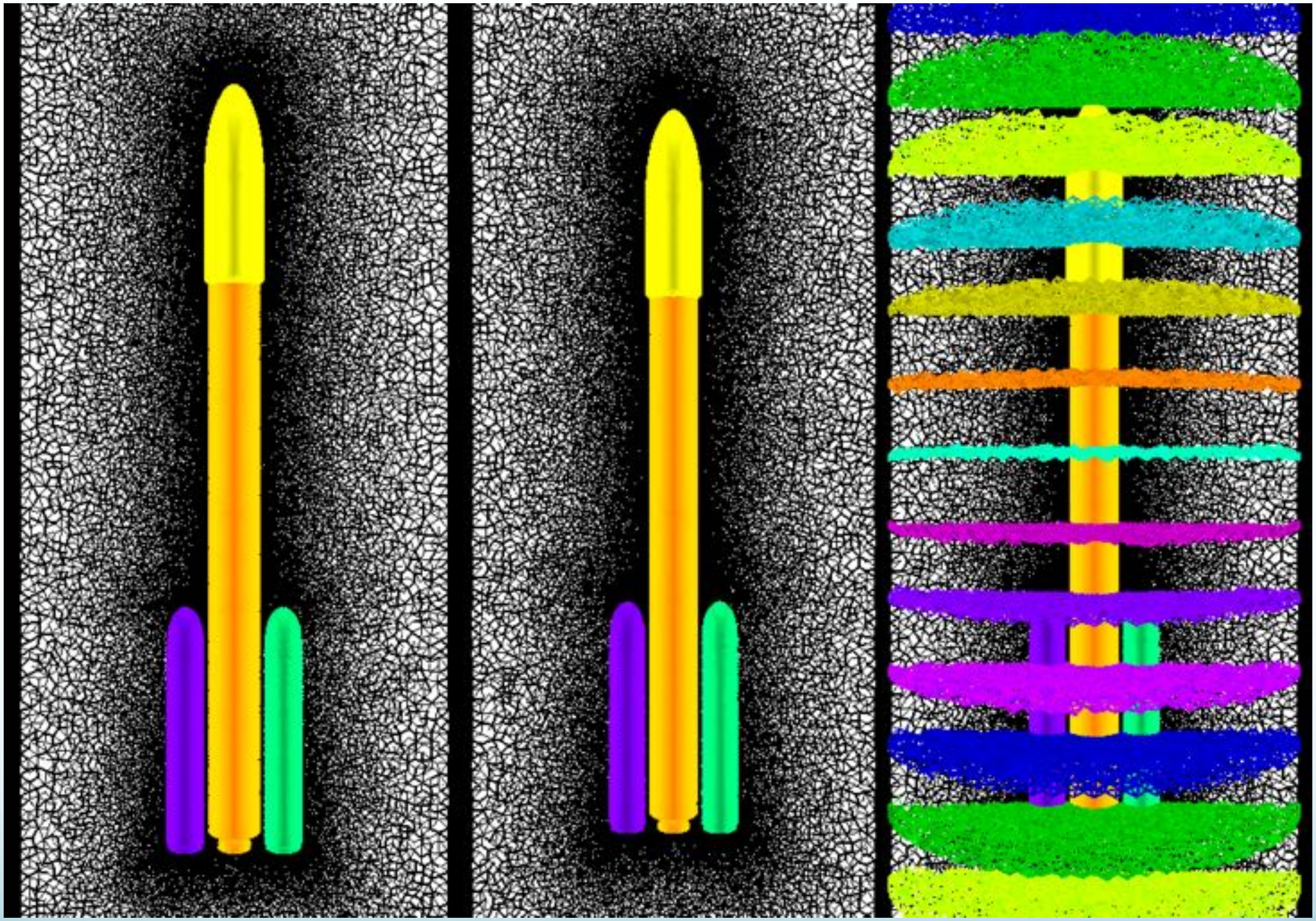
Launch Vehicle Geometry



Launch Vehicle Geometry (continued)



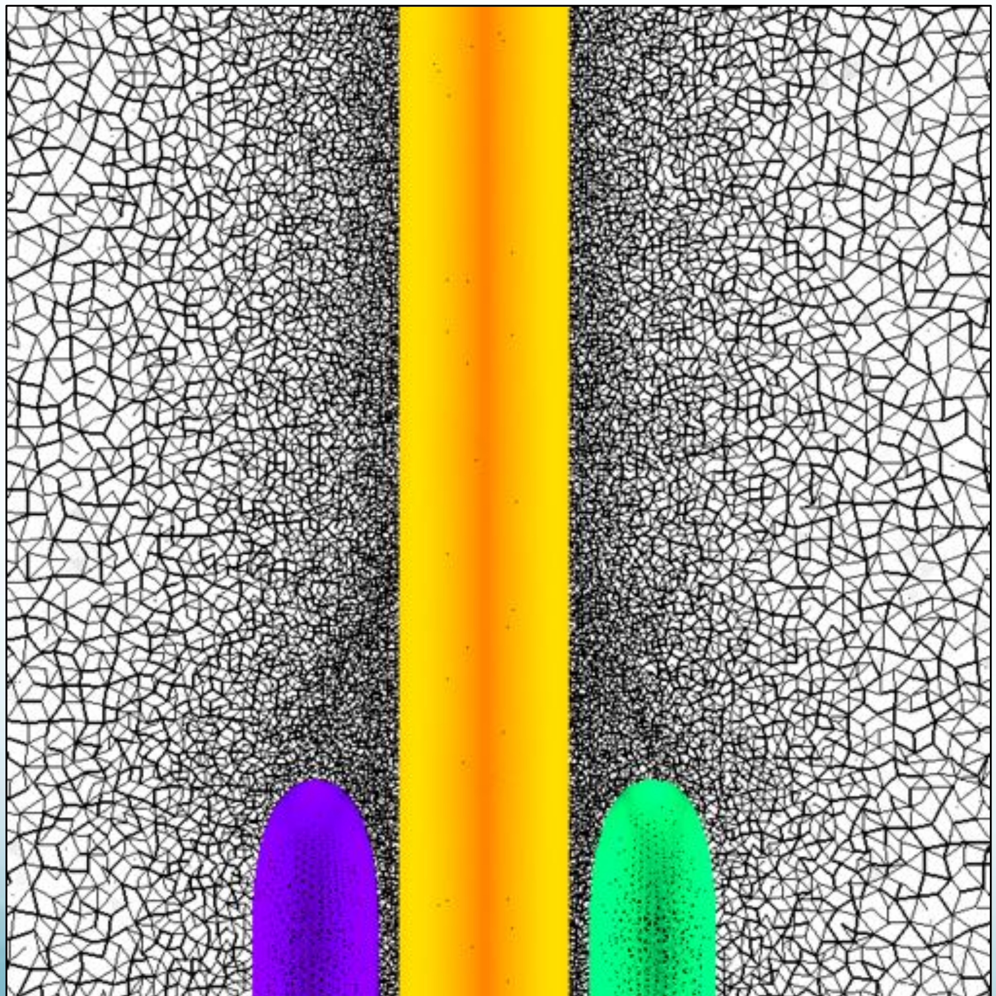
Results With Sub-Domains



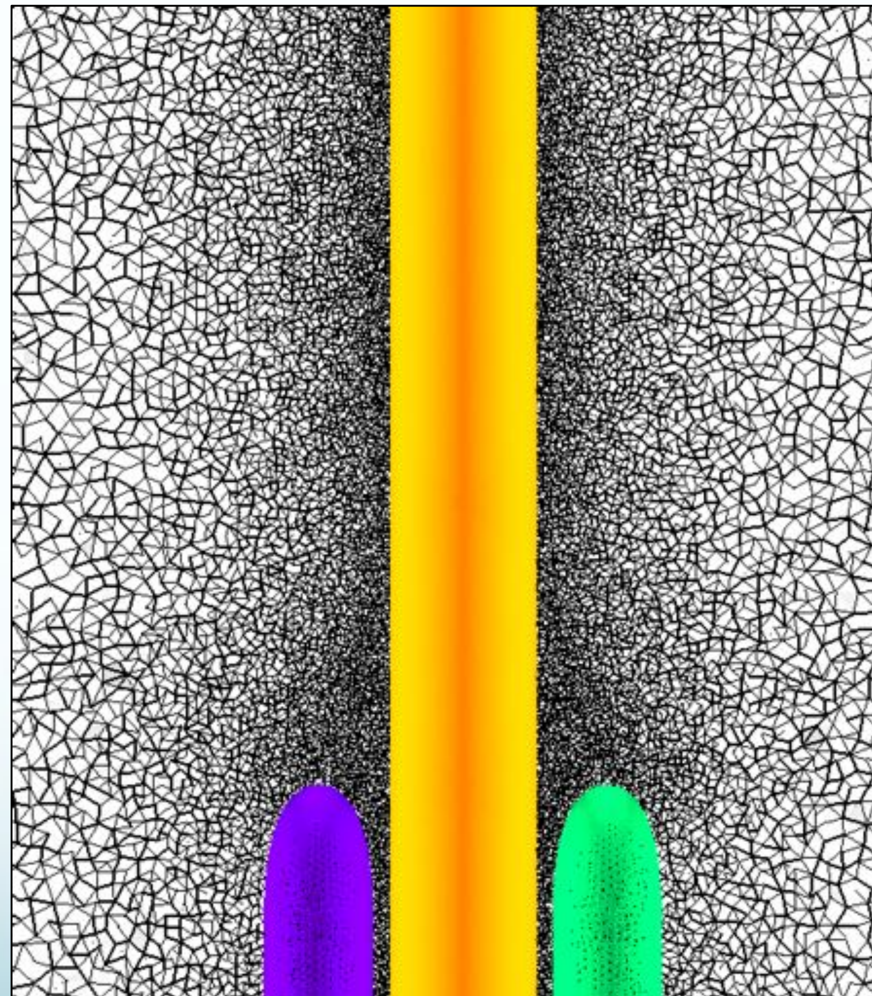
One domain

With sub-domains

Results With Sub-Domains (continued)

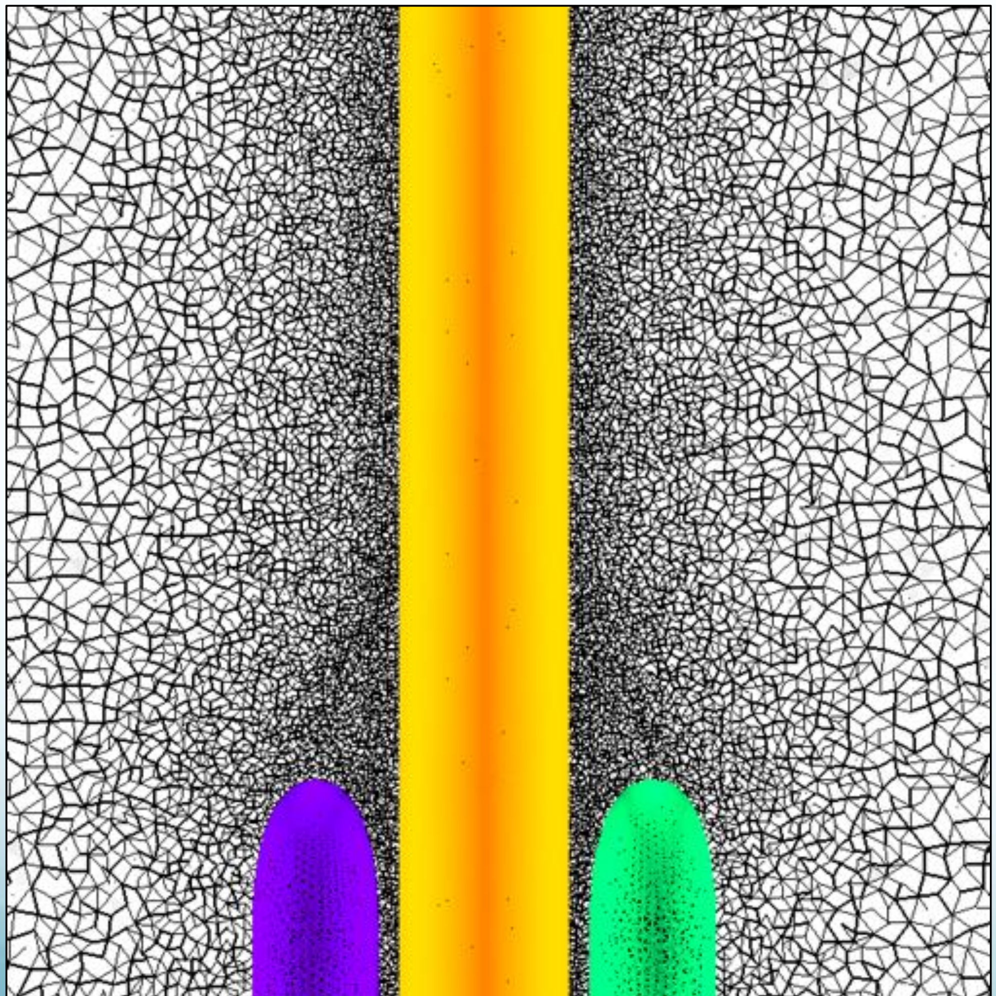


One domain

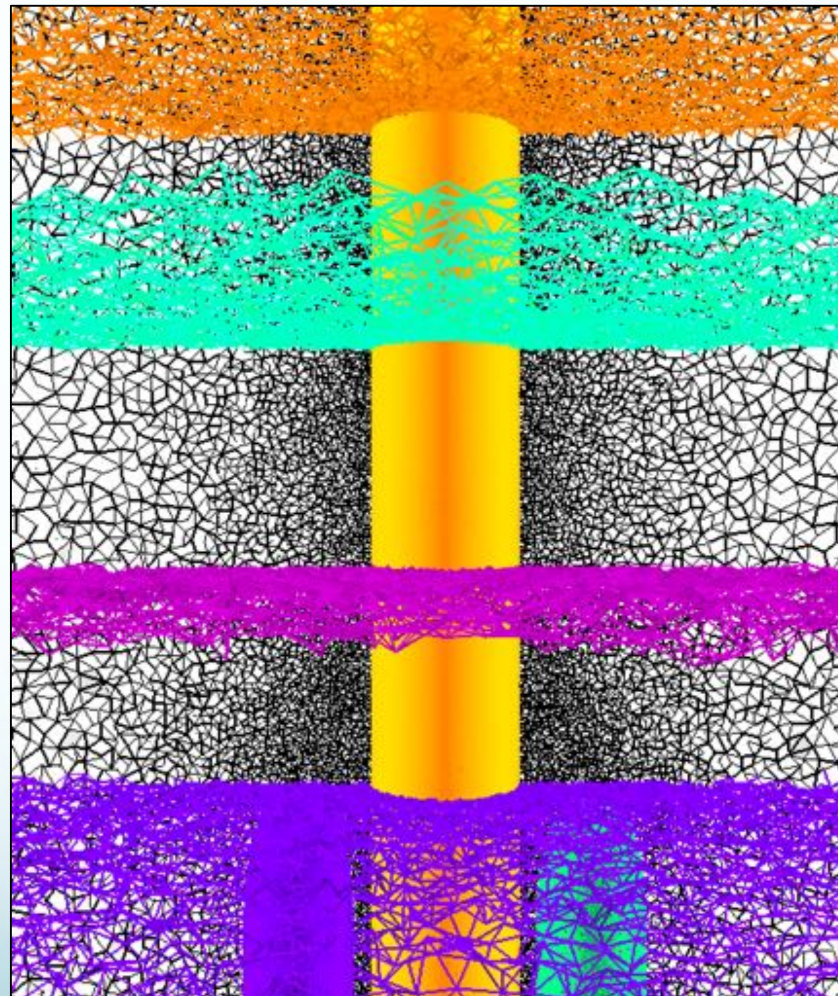


With sub-domains

Results With Sub-Domains (continued)

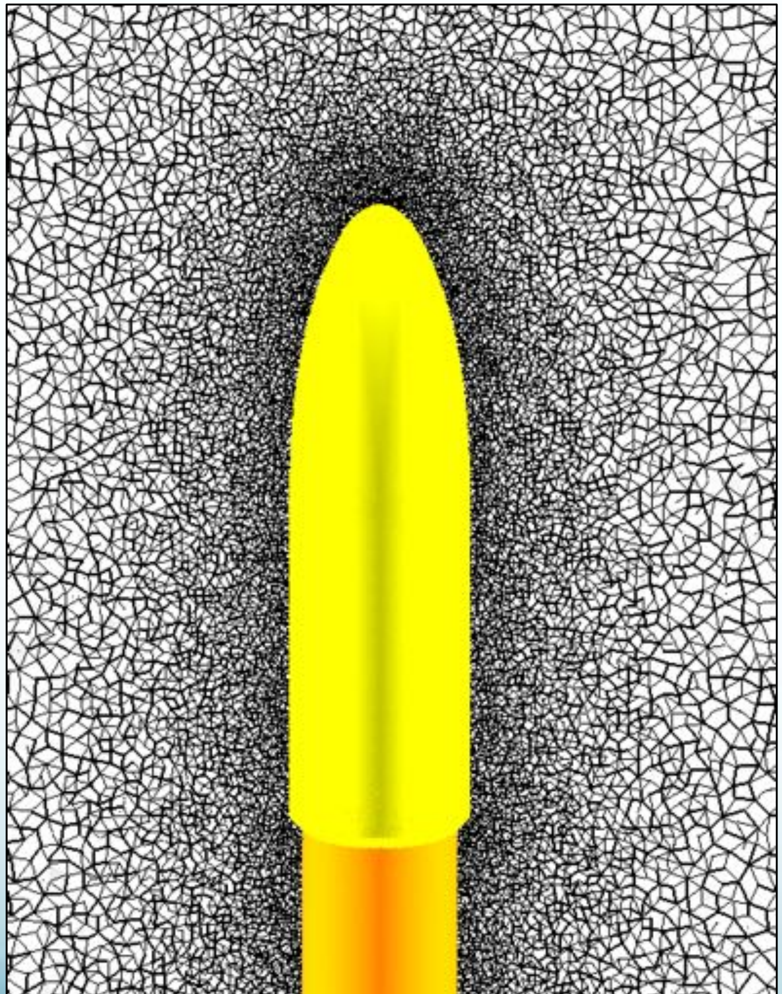


One domain

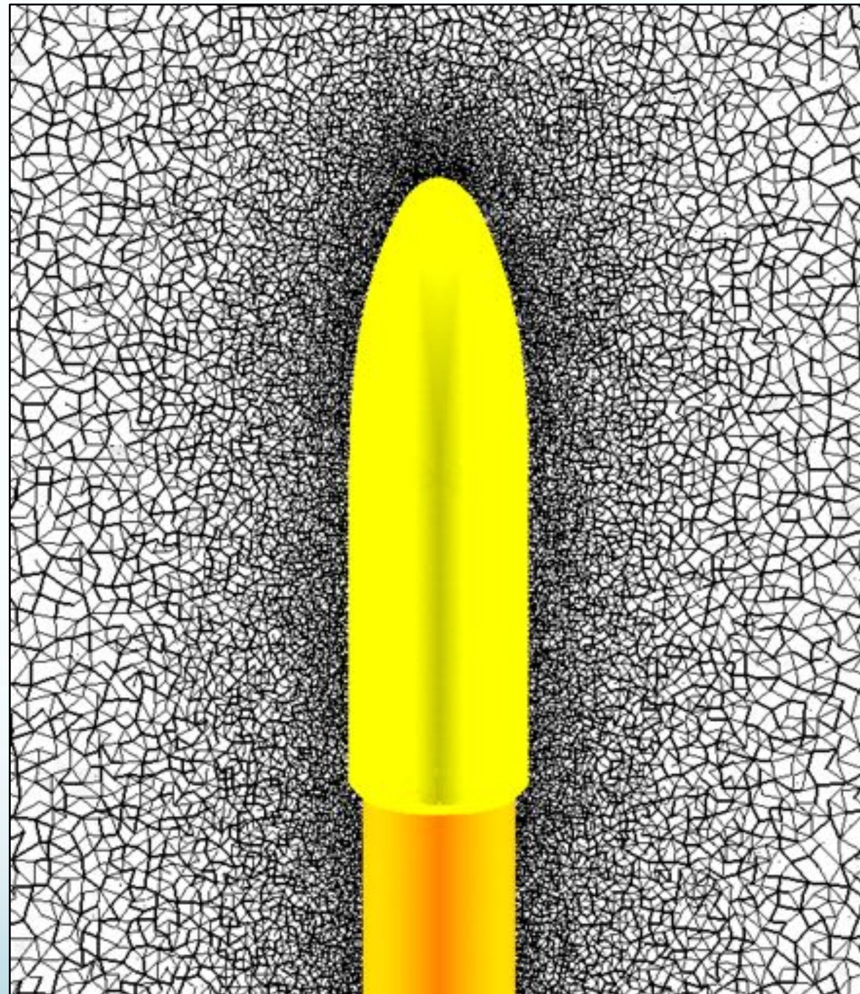


With sub-domains

Results With Sub-Domains (continued)

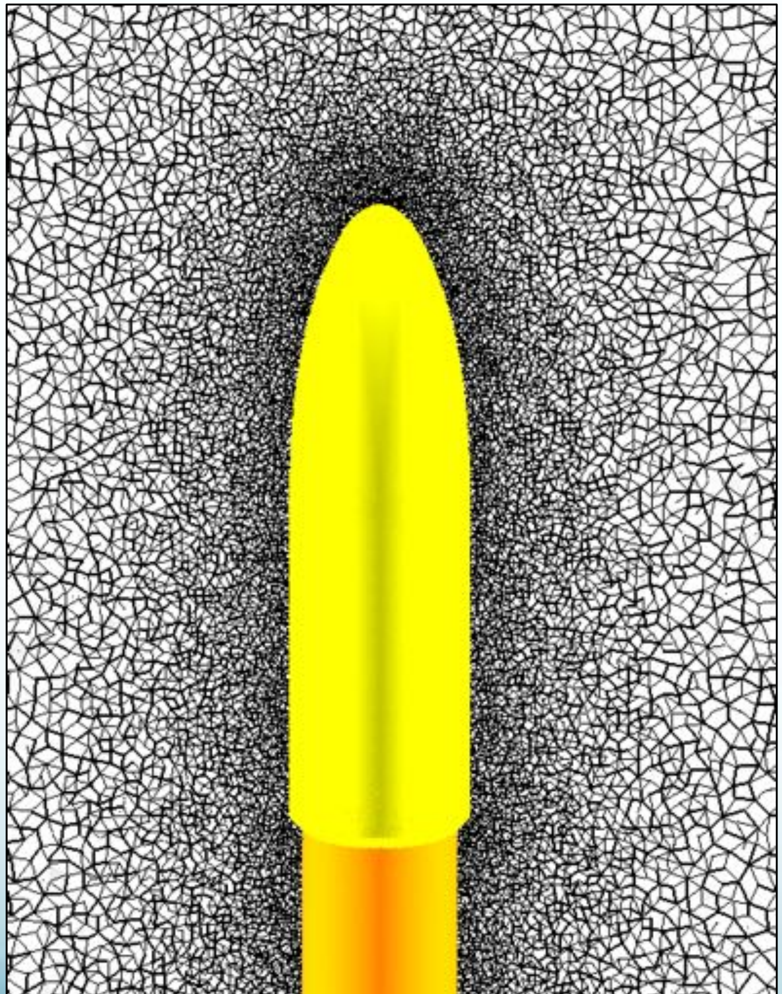


One domain

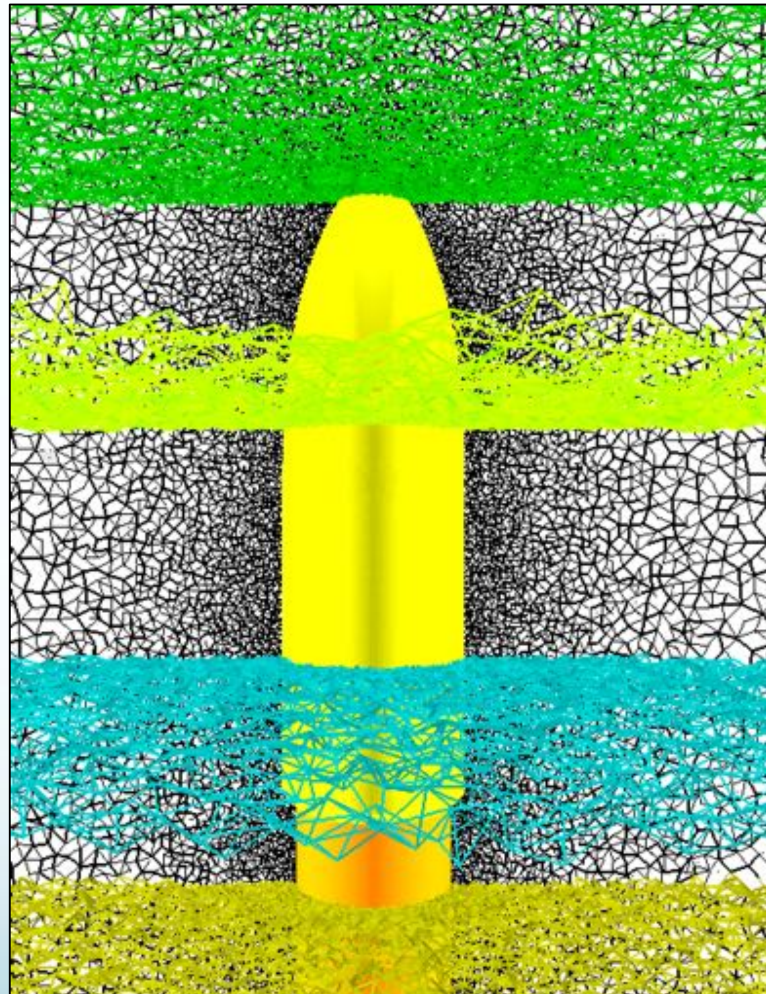


With sub-domains

Results With Sub-Domains (continued)

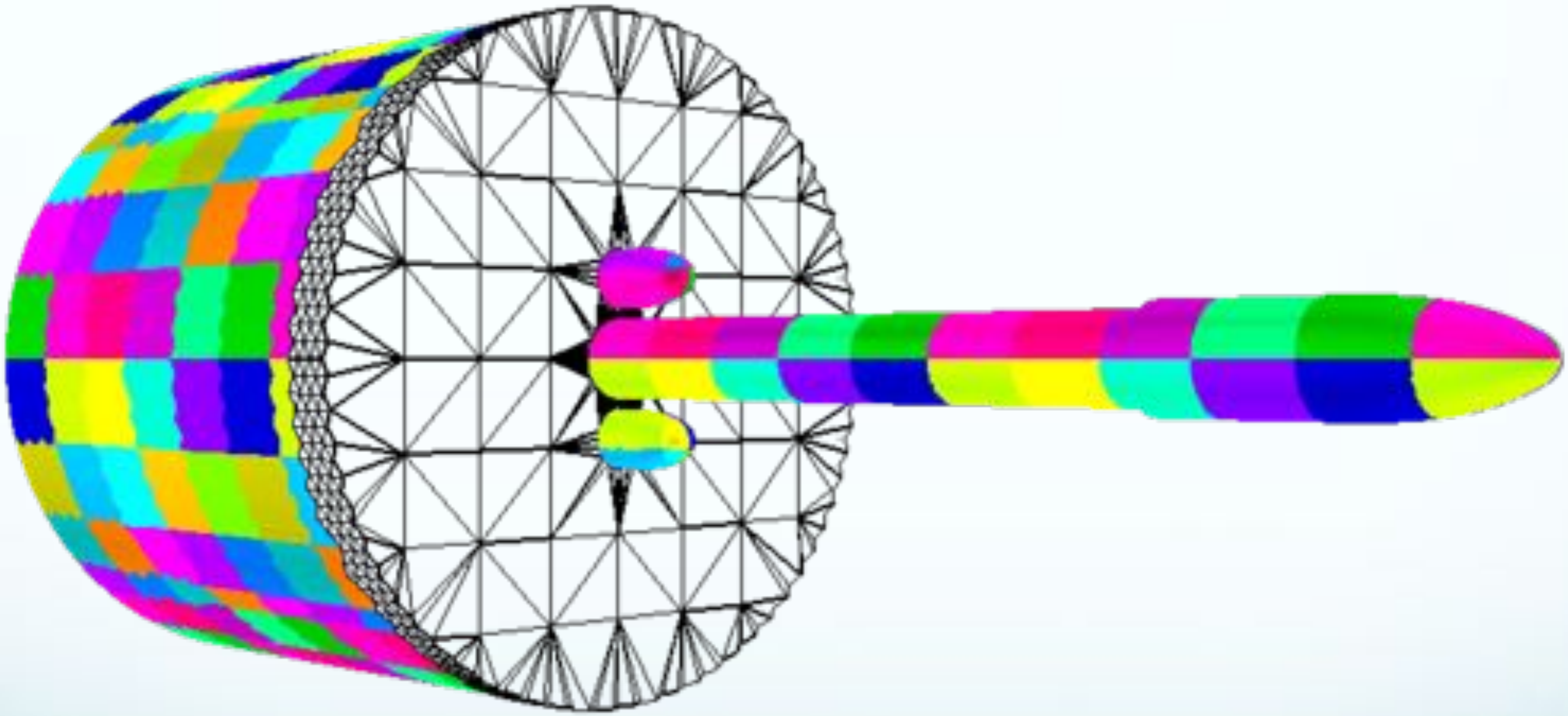


One domain



With sub-domains

Launch Vehicle Geometry with *Pseduo-Constrained* Sub-Domains



Domain split into 1355 sub-domains.

Summary

- *Virtual* sub-domain boundary concept needs more work (heuristic based) to be fully viable for insuring valid derived sub-domain boundaries.
- *Pseudo-constrained* sub-domain boundary approach is more rigorous and appears quite viable as one key part of a truly scalable process.
- Resulting mesh really is essentially the same with either *virtual* or *pseudo-constrained* boundaries.
- Current work will focus on further developing the *pseudo-constrained* sub-domain boundary approach.
- Future mesh generator work focused on both completing sub-domain approach and fine-scale parallelization.
- Future controller work focused on improving load-balancing and will investigate alternatives to the simple oct-tree like decomposition now used. Many alternatives are directly usable including local refinement and use of irregular shaped sub-domains (they don't have to be planes).