CaMEL CFD Technologies for HPC

Erdal Yilmaz
Assistant Professor
Northrop Grumman Center for HPC, Jackson State University, Jackson, MS

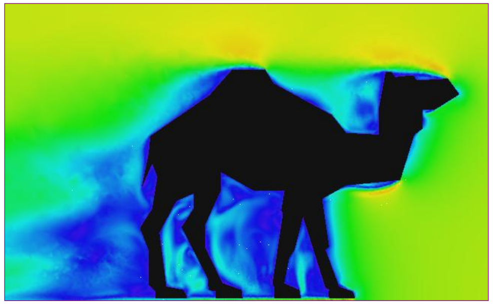
Shahrouz Aliabadi
Professor and Director
Northrop Grumman Center for HPC, Jackson State University, Jackson, MS

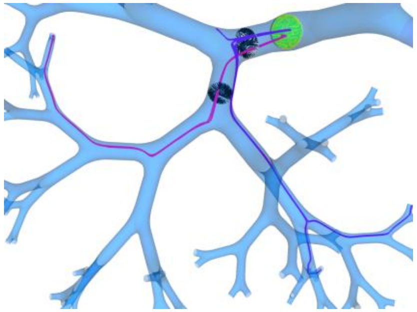
Sponsors
Northrop Grumman Ship Building
Army Research Office
Army Research Laboratory

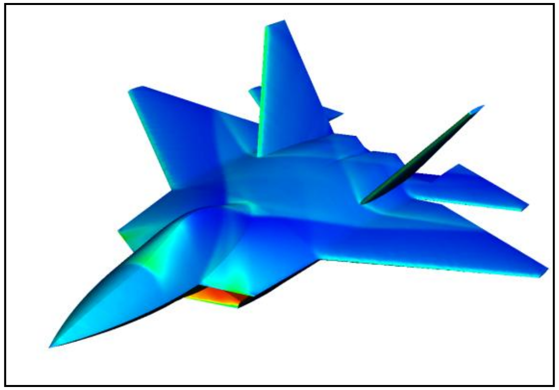
Roadmap

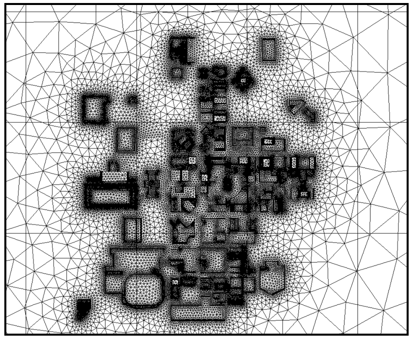
- CaMEL Flow Solvers
  - DoD Supported Flow Solver
  - Designed for Large Scale Applications
- VICE Environment
  - Platform and Solver Independent
  - Automates HPC
- Applications
  - Flapping Wing (Incompressible Flow)
  - Oscillating Projectile (Compressible Flow)
  - Mesh Multiplication

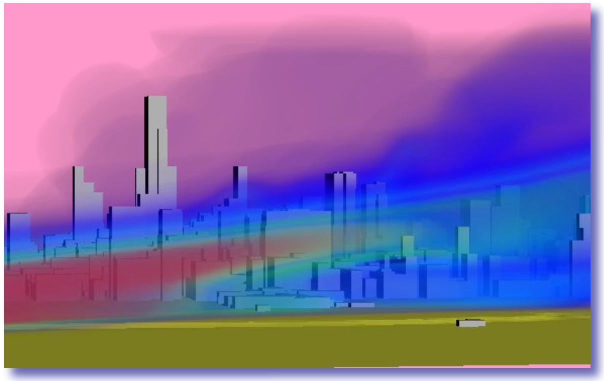
CaMEL Flow Solvers

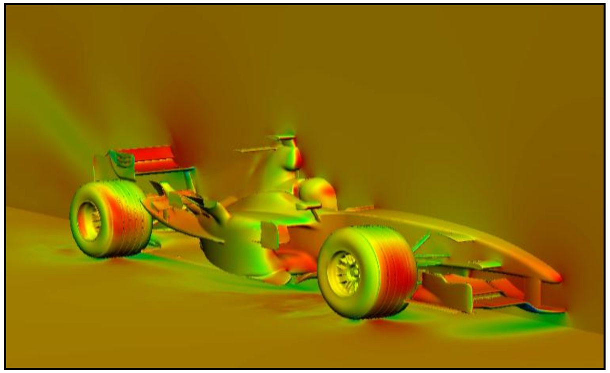


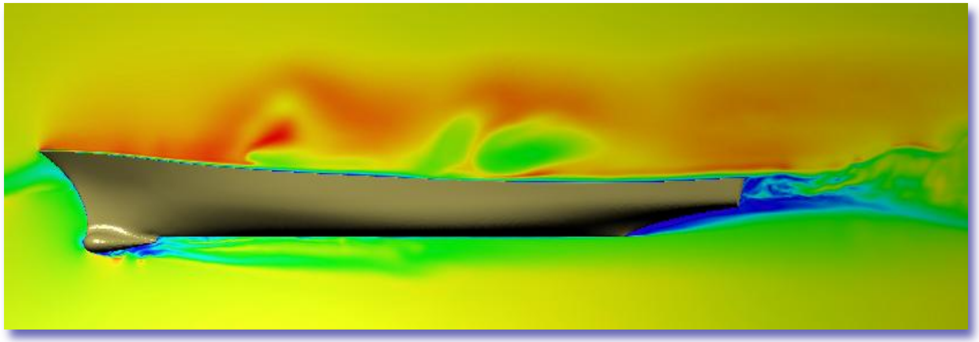


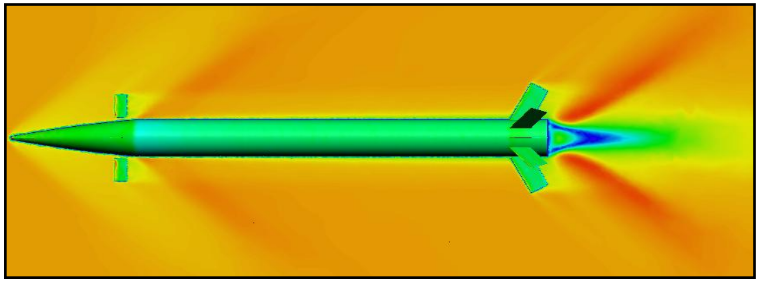












Computation and Modeling Engineering Laboratory (**CaMEL**)

Methodology

CAMEL_HYBRID

- Incompressible Navier-Stokes equations
- Single or two fluid flows with heat and mass transfer
- Hybrid Flow Solver
  - Finite element for pressure equation (node based)
  - Finite volume for momentum equations (cell center)

CAMEL_AERO

- Compressible Navier-Stokes equations
- Finite volume discretization (cell center)

CAMEL FLOW SOLVERS

- Truly second order in time and space
- Hybrid mesh (tet, hex, prism, pyramid)
- Hybrid solvers
- Matrix-free GMRES update technology
- Turbulence model with Detached Eddy Simulation

CAMEL_HYBRID Equations

Apply FV to solve Momentum Eq.

$$\rho \frac{\alpha_1 \tilde{\mathbf{u}} + \alpha_0 \mathbf{u}^n + \alpha_{-1} \mathbf{u}^{n-1}}{\Delta t} + \nabla q^n = \rho(\mathbf{g} - \tilde{\mathbf{u}} \cdot \nabla \tilde{\mathbf{u}}) + \mu \nabla^2 \tilde{\mathbf{u}}$$

Apply FE to solve Pressure Possion Eq.

$$\int_\Omega \nabla \phi^h \cdot \nabla \delta q \, \mathrm{d}\Omega = \frac{\rho \alpha_1}{\Delta t} \int_\Omega \nabla \phi^h \cdot \mathbf{u}^h \, \mathrm{d}\Omega - \int_{\Gamma_D + \Gamma_{sym}} \phi^h \left(\frac{\rho \alpha_1}{\Delta t} \mathbf{n} \cdot \mathbf{u}^h \right) \mathrm{d}\Gamma$$

Update Pressure

$$p_i^{n+1} = \sum_{k=1}^{nen} \left[q_k^{n+1} - \mu \left(\nabla \cdot \mathbf{u}^{n+1} \right)_k \right] \phi_k$$

CaMEL Aero: Equations

Unsteady compressible Navier-Stokes equations in the conservative form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_j}{\partial x_j} - \frac{\partial \mathbf{E}_j}{\partial x_j} = 0 \qquad \mathbf{U} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho E \end{bmatrix} \quad \mathbf{F}_j = \begin{bmatrix} \rho u_j \\ \rho u_i u_j + \delta_{ij} p \\ (\rho E + p) u_j \end{bmatrix} \quad \mathbf{E}_j = \begin{bmatrix} 0 \\ \tau_{ji} \\ -q_j + \tau_{jk} u_k \end{bmatrix}$$

FV discretization : second order accurate in space

$$\left|\Omega_i\right| \frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = 0 \qquad \mathbf{R}(\mathbf{U}) = \sum_{l=1}^{n_f} (\mathbf{H} \cdot \mathbf{n} \Delta s)_l$$

Implicit second-order accurate time integration scheme

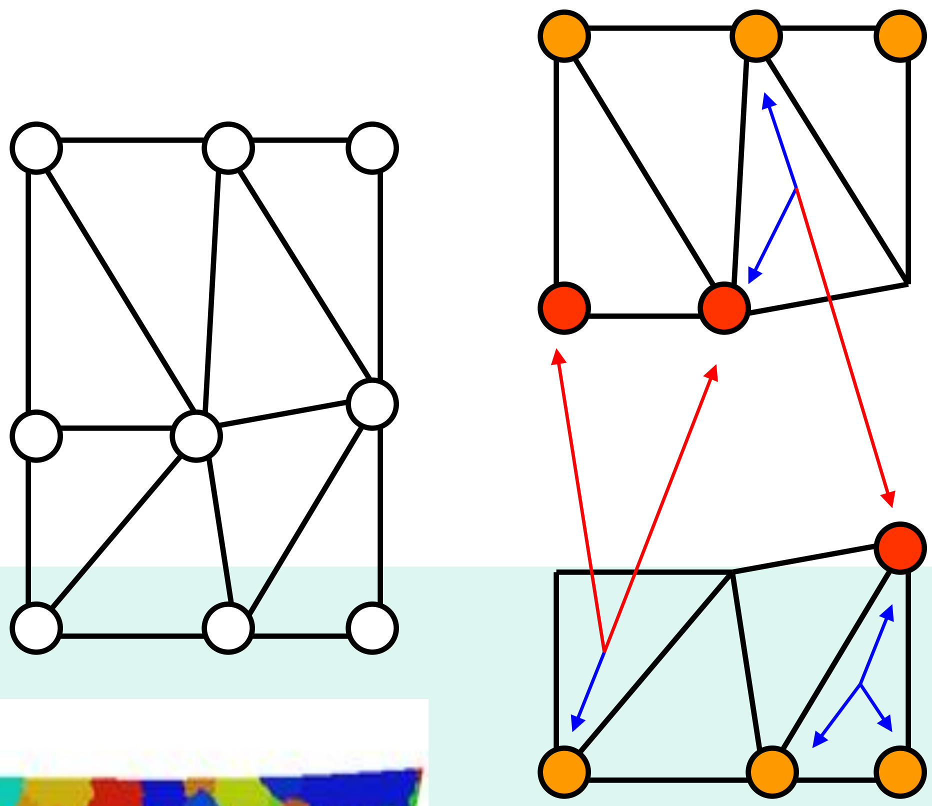
$$\left|\Omega_i\right| \frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} + [(1-\alpha)\mathbf{R}(\mathbf{U}^n) + \alpha \mathbf{R}(\mathbf{U}^{n+1})] = 0$$

α=0.5 for 2nd order

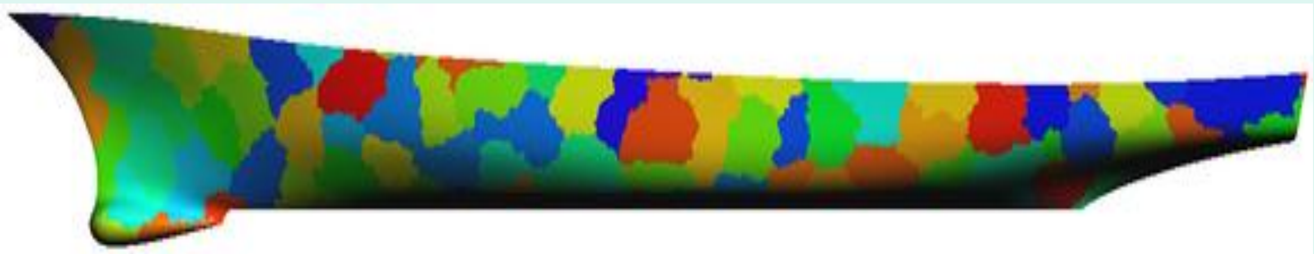
Parallelization

MPI-based parallelism:
Communication between cores are across partitioned boundaries

Data Distribution

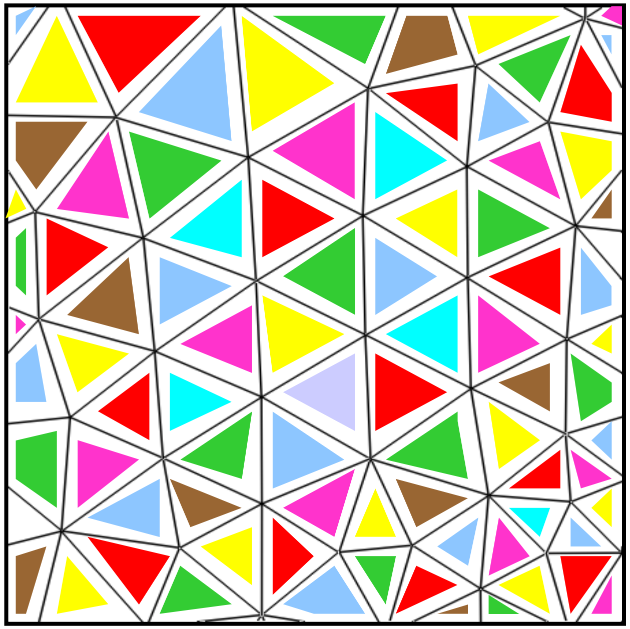


Mesh Partitioning



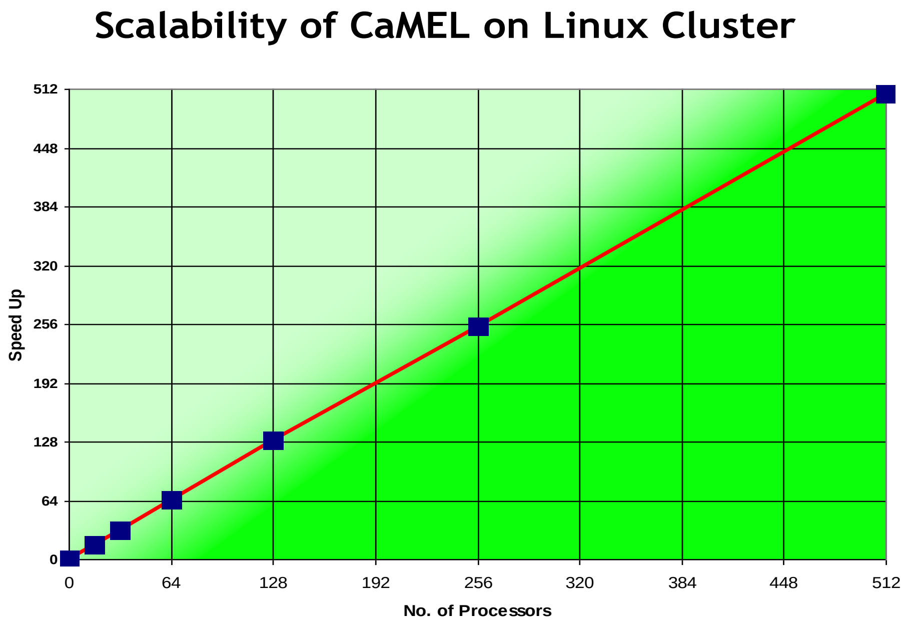
Element grouping to achieve Vectorization

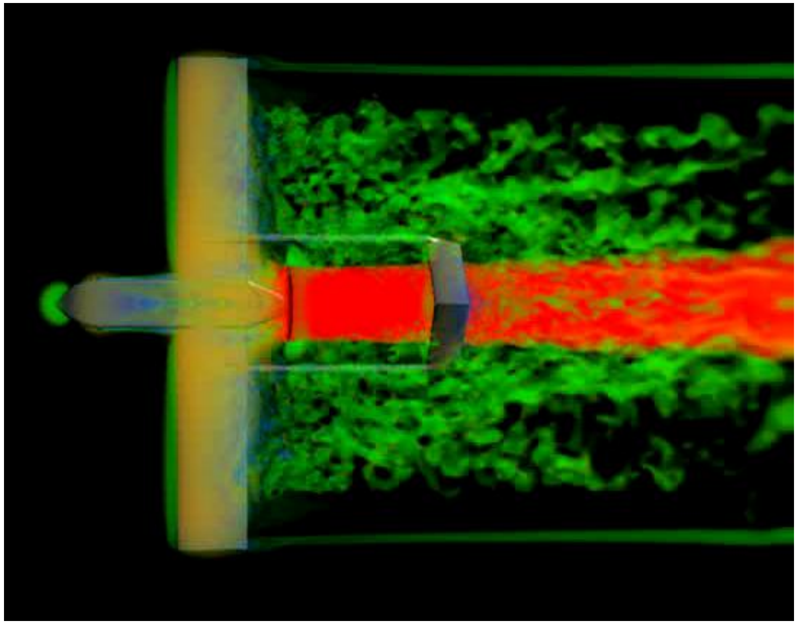
- No two elements in a group (color) can touch each other
- 8 Colors in this example
- Around 40 colors in 3D unstructured meshes
- Vectorization also speeds up the computations in serial computations!

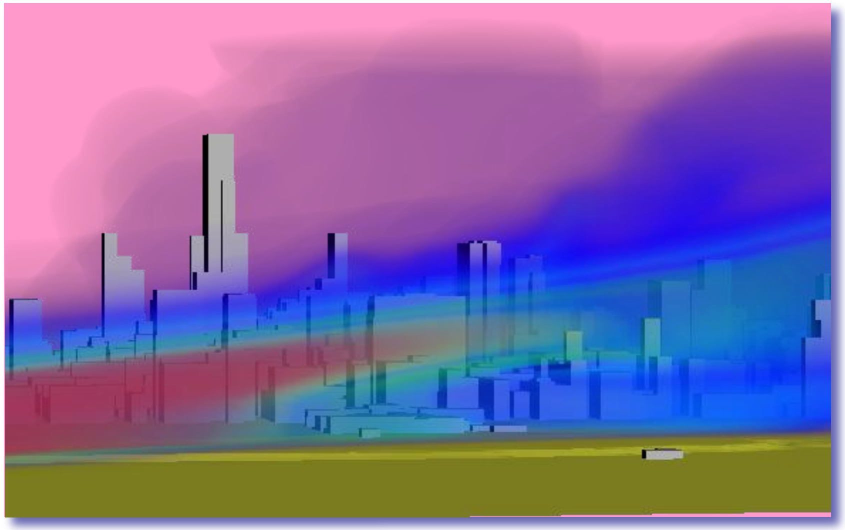


Scalability and Performance

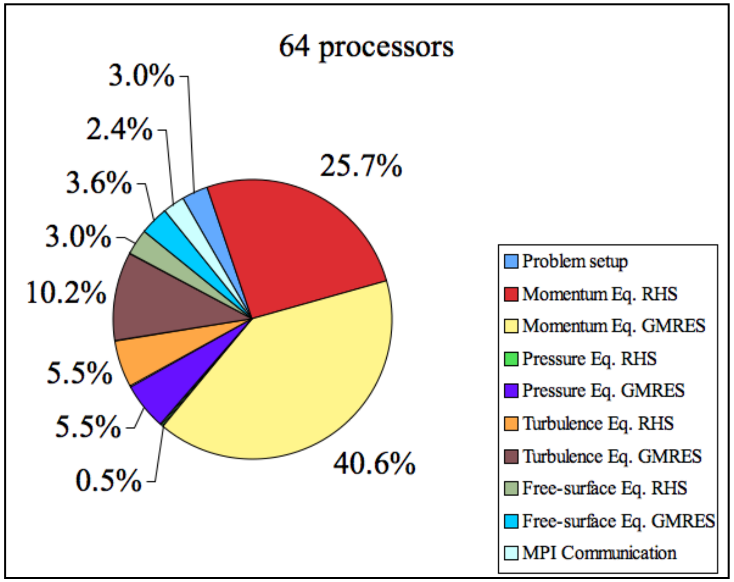
- Mesh: 2.1 Billion Tetrahedral Elements
- Equations: More that 1.4 Billion Unknowns
- Fully Implicit
- Teraflops Sustained Speed
- 3.18% Communication
- CRAY

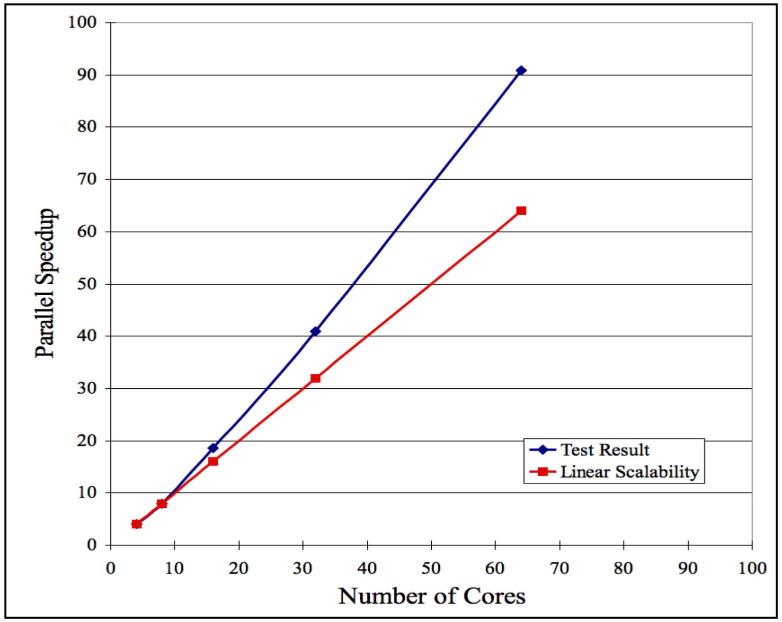


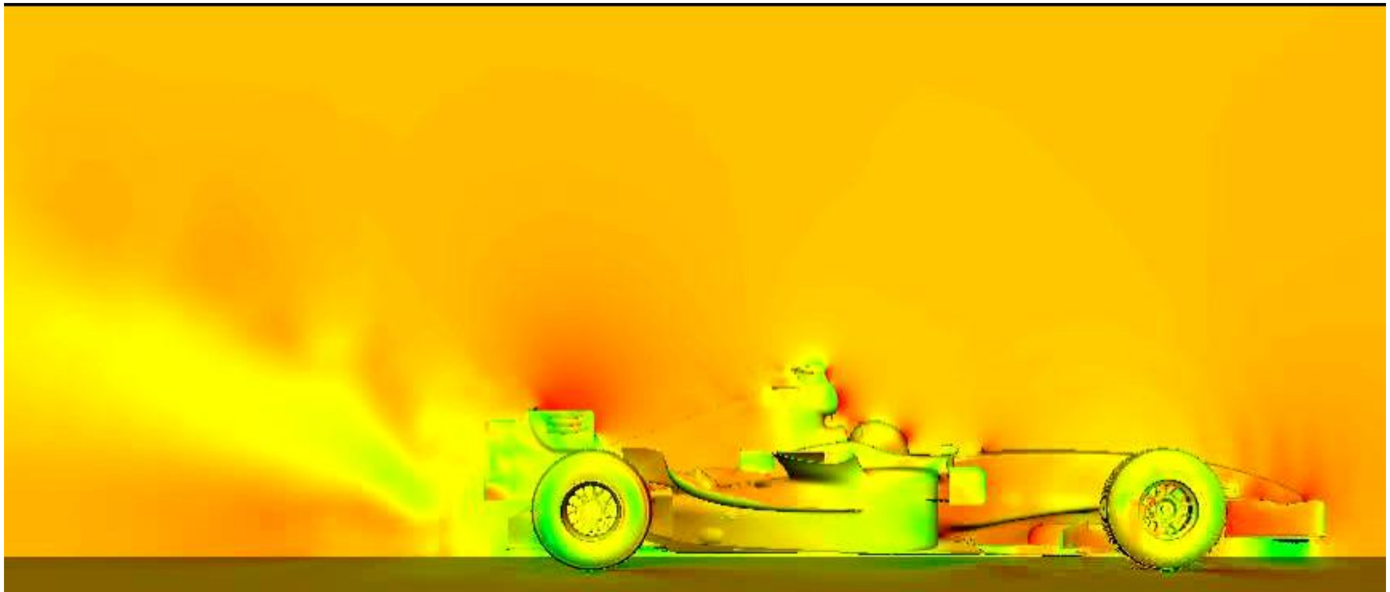




Scalability and Performance







Scalability and Performance

$$T_c = \frac{N_{proc}}{N_{elm} N_{ts} N_{it} N_k} T_{run}$$

- T_c: Cost Per Elements
- N_{proc}: No. of Processors
- T_{run}: Total CPU Time
- N_{elm}: No. of Elements
- N_{ts}: No. of Time Steps
- N_{it}: No. of Nonlinear Iterations
- N_k: No. of Krylov Space

$$1\mu s \leq T_c \leq 2\mu s$$

Spatial Accuracy

$$r = \frac{\log(\| e \|_{h+1} / \| e \|_h)}{\log 0.5} \qquad \|e_m\| = \sqrt{\frac{\sum_{i=1}^{ne} [(u^h - u^e)A_i]^2 + [(v^h - v^e)A_i]^2}{ne}} \qquad \|e_p\| = \sqrt{\frac{\sum_{i=1}^{ne} [(p^h - p^e)A_i]^2}{ne}}$$

$\Delta t = 0.0001$ – small time step is chosen to ensure negligible temporal error.

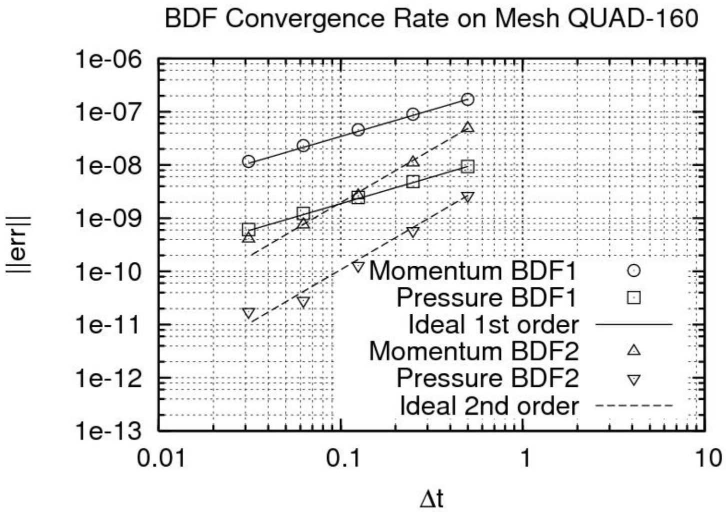
mesh size *h*	$\|\|err_m\|\|$	*r*	$\|\|err_p\|\|$	*r*
0.05	0.53235E-07	-	0.25007E-07	-
0.025	0.37310E-08	3.83	0.30310E-08	3.04
0.0125	0.29247E-09	3.67	0.22693E-09	3.74
0.00625	0.19940E-10	3.87	0.13155E-10	4.11

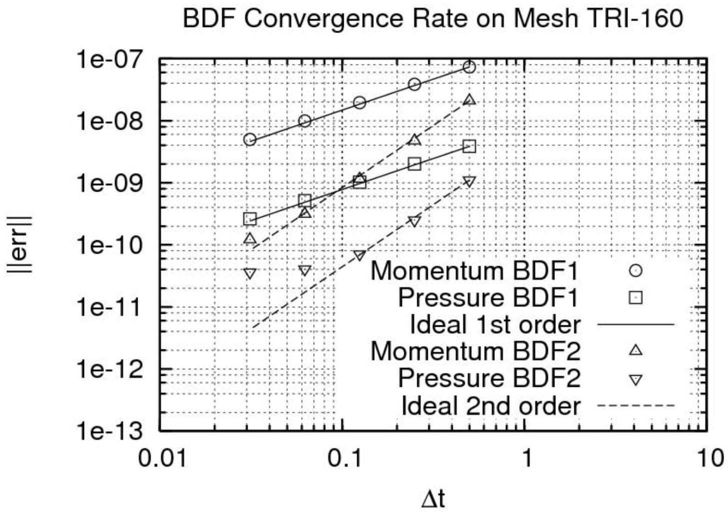
quad mesh

mesh size *h*	$\|\|err_m\|\|$	*r*	$\|\|err_p\|\|$	*r*
0.05	0.20035E-06	-	0.77954E-07	-
0.025	0.10106E-07	4.31	0.41594E-08	4.23
0.0125	0.33023E-09	4.94	0.23846E-09	4.12
0.00625	0.10998E-10	4.91	0.14516E-10	4.04

tri mesh

Temporal Accuracy





1. Shuangzhang Tu and Shahrouz Aliabadi, "**Development of a Hybrid Finite Volume/Element Solver for Incompressible Flows**", International Journal of Numerical Methods in Fluids. 2007; 55:177-203.

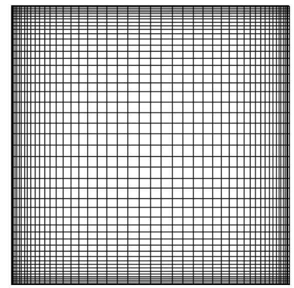
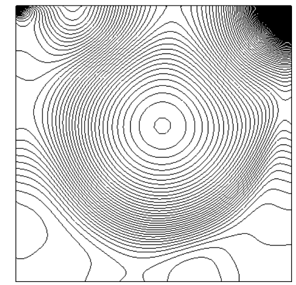
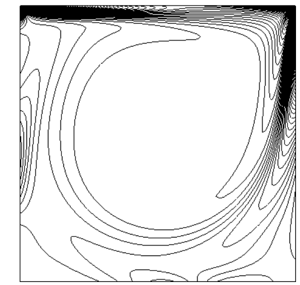
2. Shuangzhang Tu, Shahrouz Aliabadi, Reena Patel, and Marvin Watts, "**An implementation of the Spalart-Allmaras DES model in an implicit unstructured hybrid finite volume/element solver for incompressible turbulent flow**", International Journal of Numerical Methods in Fluids. 2009; 59: 1051-1062.

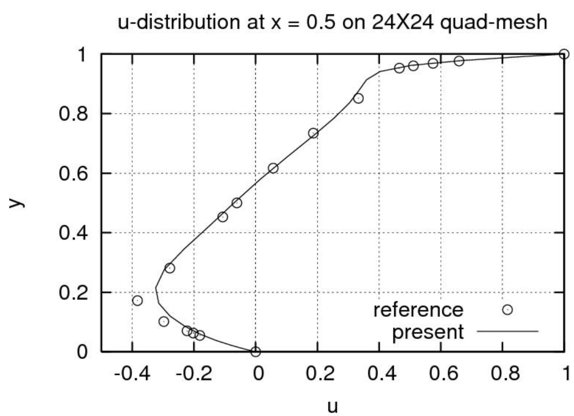
3. Tian Wan, Shahrouz Aliabadi and Christopher Bigler, "**A Hybrid Scheme Based on Finite Element / Volume Methods for Two Immiscible Fluid Flows**", International Journal of Numerical Methods in Fluids. 2009; 61: 930-944.

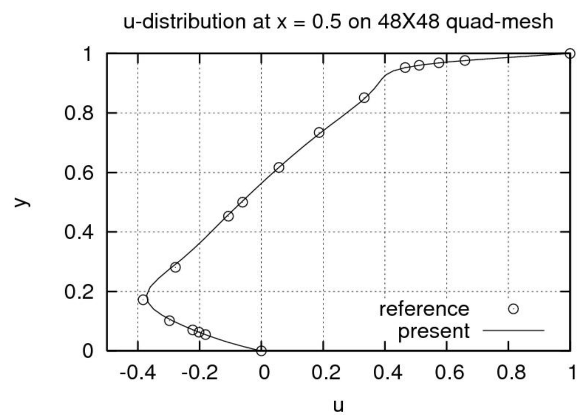
4. Shahrouz Aliabadi, Muhammad Akbar and Reena Patel, "**Hybrid Finite Element / Volume Method for Shallow Water Equations**", submitted to the International Journal of Numerical Methods in Engineering.

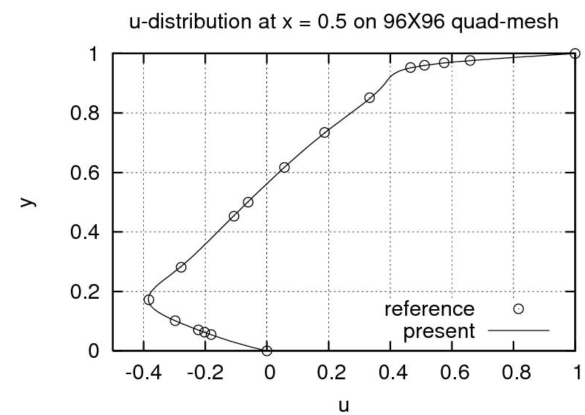
Accuracy

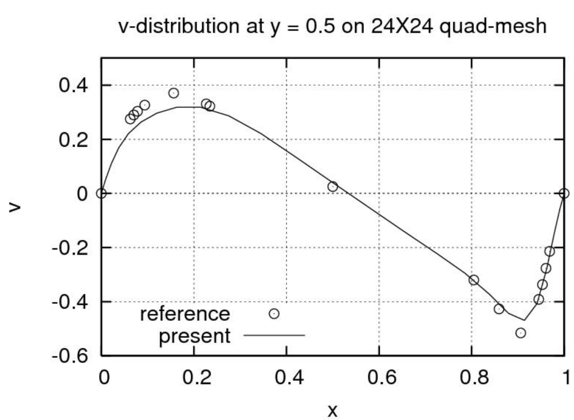
Lid-Cavity
Re =1000

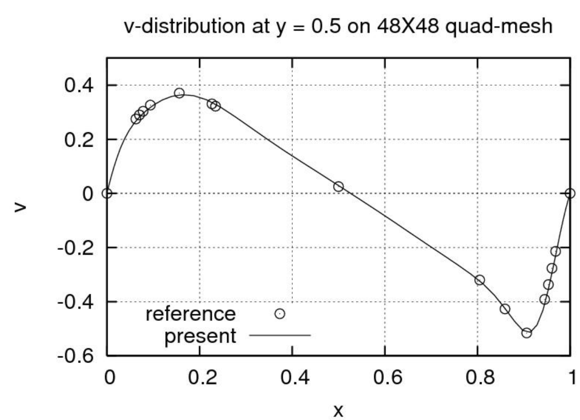




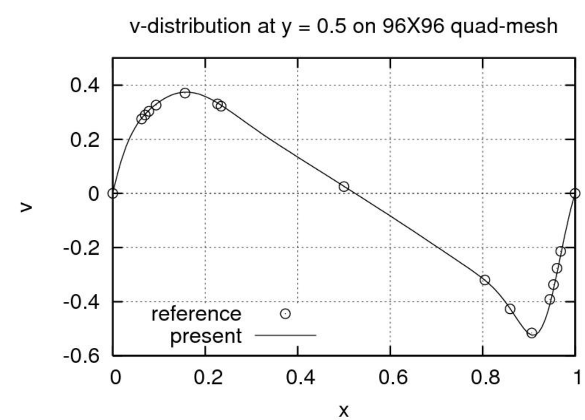




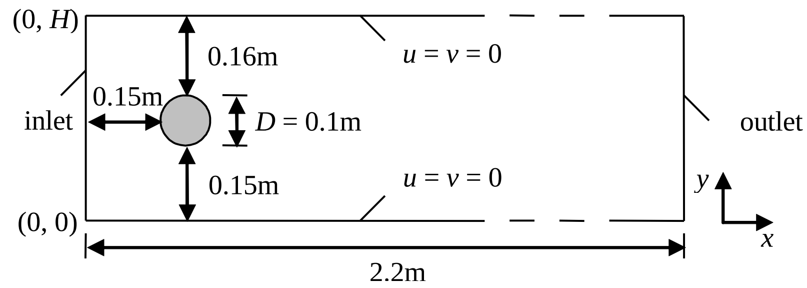


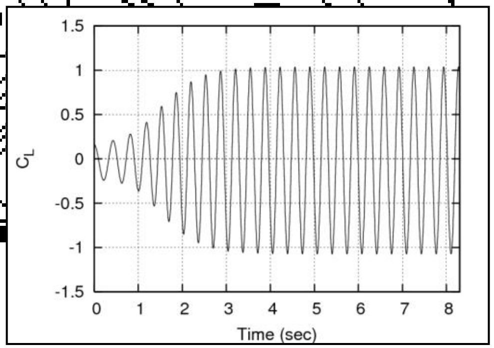




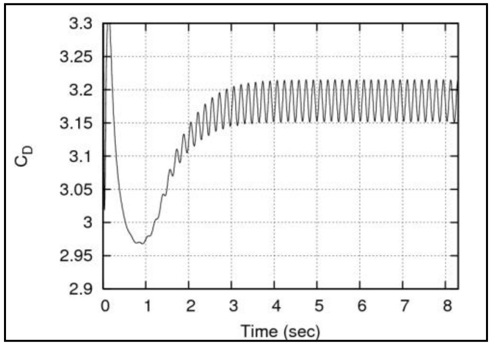


Accuracy





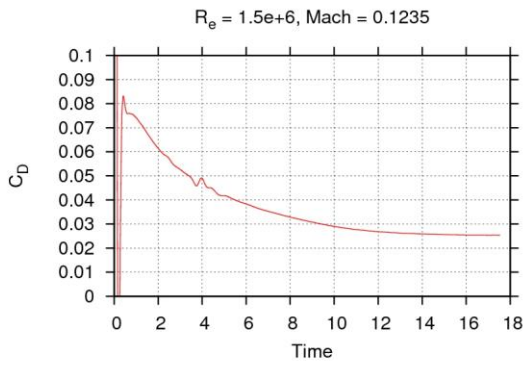
		$C_{D\,max}$	$C_{L\,max}$	*St*	Dp
Reference	lower bound	3.22	0.99	0.295	2.46
	upper bound	3.24	1.01	0.305	2.5
Present		3.215	1.0374	0.2959	2.4687

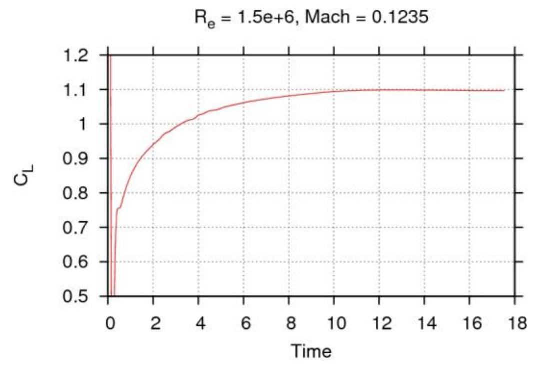


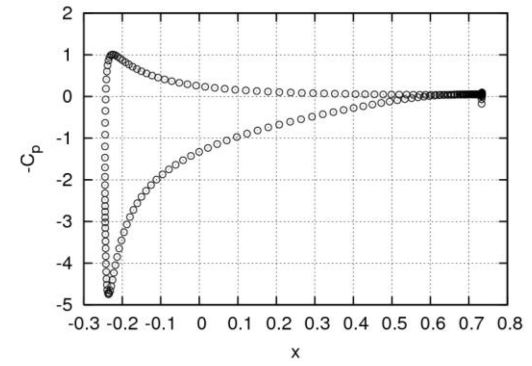
Accuracy: CHF3 & Aero

2D benchmark: NACA 0015 at 12° angle of attack

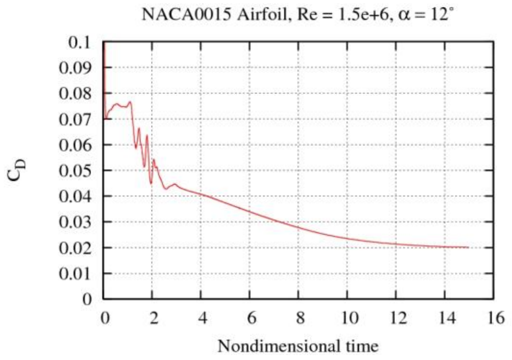
Compressible Flow Results

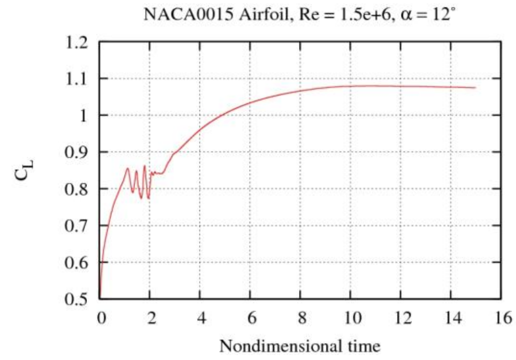


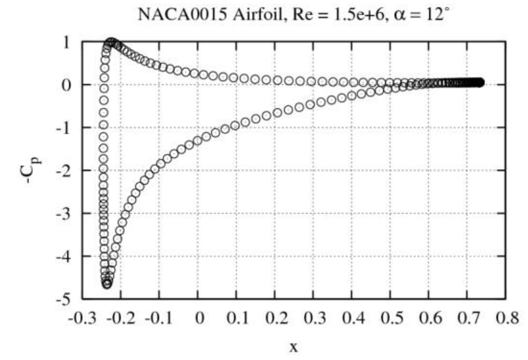




Hybrid Method Results







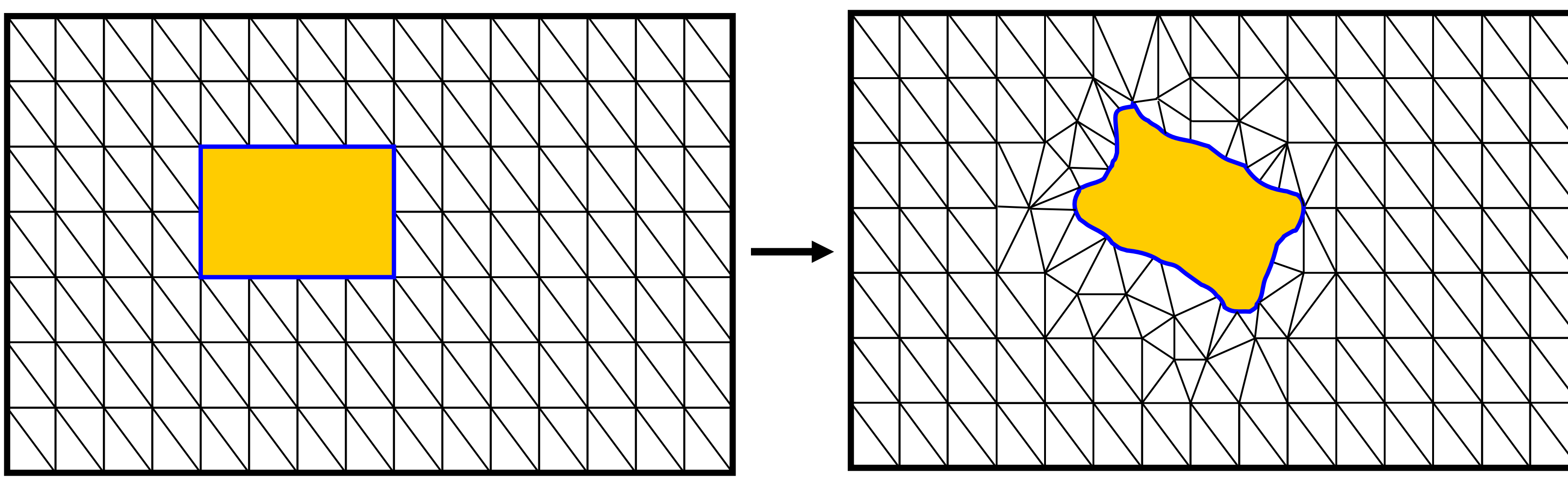
Mesh moving

nodes are moved using nonlinear elasticity equations

$$\int_{V_0} \delta\mathbf{u}\cdot\left(\rho_0 \frac{\partial^2 \mathbf{u}}{\partial t^2}\right) dV_0 + \int_{V_0} \mathbf{S}\&\delta\mathbf{E}\, dV_0 - \int_{V} \rho_0\, \delta\mathbf{u}\cdot \mathbf{F}\, dV_0 = \int_{A_h} \delta\mathbf{u}\cdot \mathbf{t}\, dA$$

$$S_{ij} = \lambda\delta_{ij}E_{mm} + 2\mu E_{ij}$$

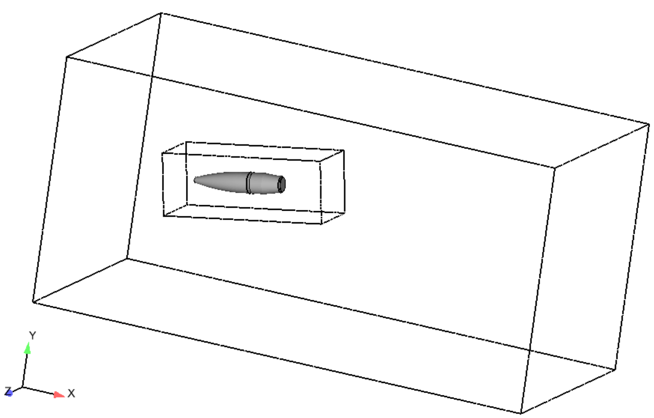
$$E_{ij} = \frac{1}{2}\left(\frac{\partial \mathbf{X}}{\partial x_i}\cdot\frac{\partial \mathbf{X}}{\partial x_j} - \delta_{ij}\right) = \frac{1}{2}\left(\frac{\partial X_p}{\partial x_i}\frac{\partial X_p}{\partial x_j} - \delta_{ij}\right)$$

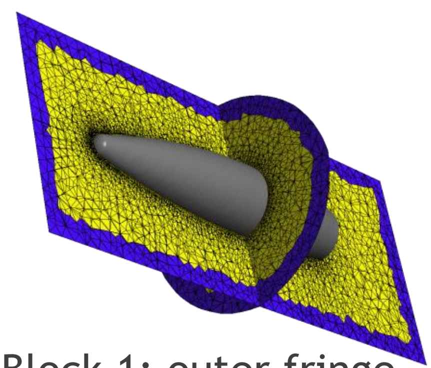


Overset mesh:basics

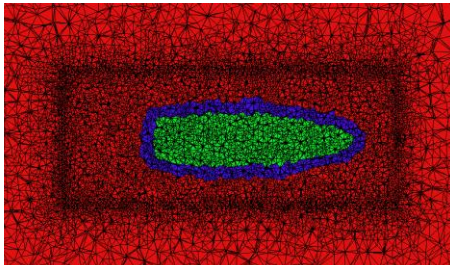
- Use multi mesh blocks and interpolate flow solution among them at the overlapped region.
- Complex objects or moving body is meshed separately.
- Avoids mesh deformations. Note that for deforming body mesh movement is needed

- **Hole cutting**: Cut solid, user defined bounding boxes and facets, sphere, cylinder, etc
- **Fringe**: boundary points between hole points and active solution points
  - Inner fringe adjacent to holes
  - Outer fringe at outer/overlap boundary
- **Donor**: cells/nodes that fringe points are interpolated.
- **Orphans**: are those fringe points/cells that can not find any donor cells/points





Block 1: outer fringe



Block 2 inner fringe

Overset Tool: DiRTlib/SUGGAR package

DiRTlib

- Donor interpolation/Receptor Transaction library
- Add overset capability to flow solver
- Requires partition mapping of elements/nodes for parallel runs

SUGGAR

- Structured, Unstructured, Generalized overset Grid AssembleR
- General overset grid assembly
- Node and/or cell centered formulation
- XML based input file
- Hierarchical grouping for mesh blocks, octree or binary search
- Multi-treaded, not fully parallel
- Supports VGRID and COBALT formats for unstructured mesh
- Successfully working with codes such as FUN3D, USM3D,WIND,
- COBALT, CFDSHIP-IOWA, (now CaMEL).
- ***Developed by Dr. Ralph Noack, ARL @ PSU***

SUGGAR/DiRTlib Implementation into CaMEL

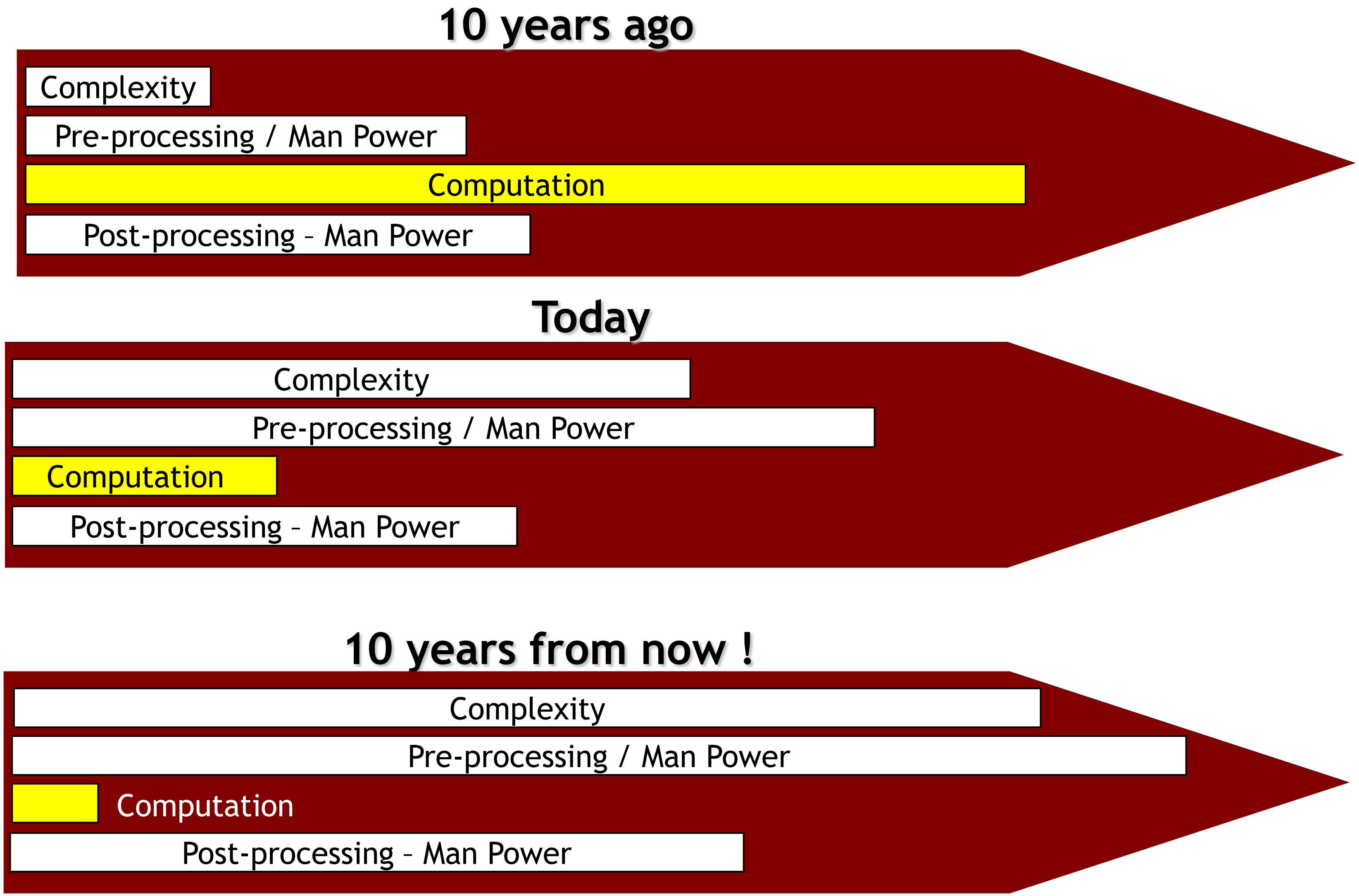
Mesh pre-processing:
Generate individual mesh blocks. Write it in Cobalt Format for SUGGAR and examine for orphans
Combine mesh blocks in to one for CaMEL Aero

Flow Solver:

1. Use ParMETIS partition data to generate global mapping for DiRTlib
2. Move mesh and generate SUGGAR motion file
3. Run SUGGAR (libsuggar) to generate DCI data, or use previously generated DCI file
4. DiRTlib reads DCI data and mesh partition mapping to initialize solution exchange
5. Get IBLANK vector using DiRTlib functions
6. Blankout values (residuals) at out and fringe nodes/cells
7. Make sure any orphan nodes/cells are taken care of not to cause flow solver crush
8. DiRTlib Update values (flow variables, unknowns) at fringe nodes/cells
9. Go to step 2 until the end of the motion

Solution post-processing:
Split solution into original block to Viz. individually or use combined blocks.

HPC Project Components' Timeline



High Performance Computing Complexity

- Large-Scale computations are driven by complexity of the problem.
- We generate very refined meshes to better represent this complexity.
- Refined meshes are relaxation to complexity.

Complexity !

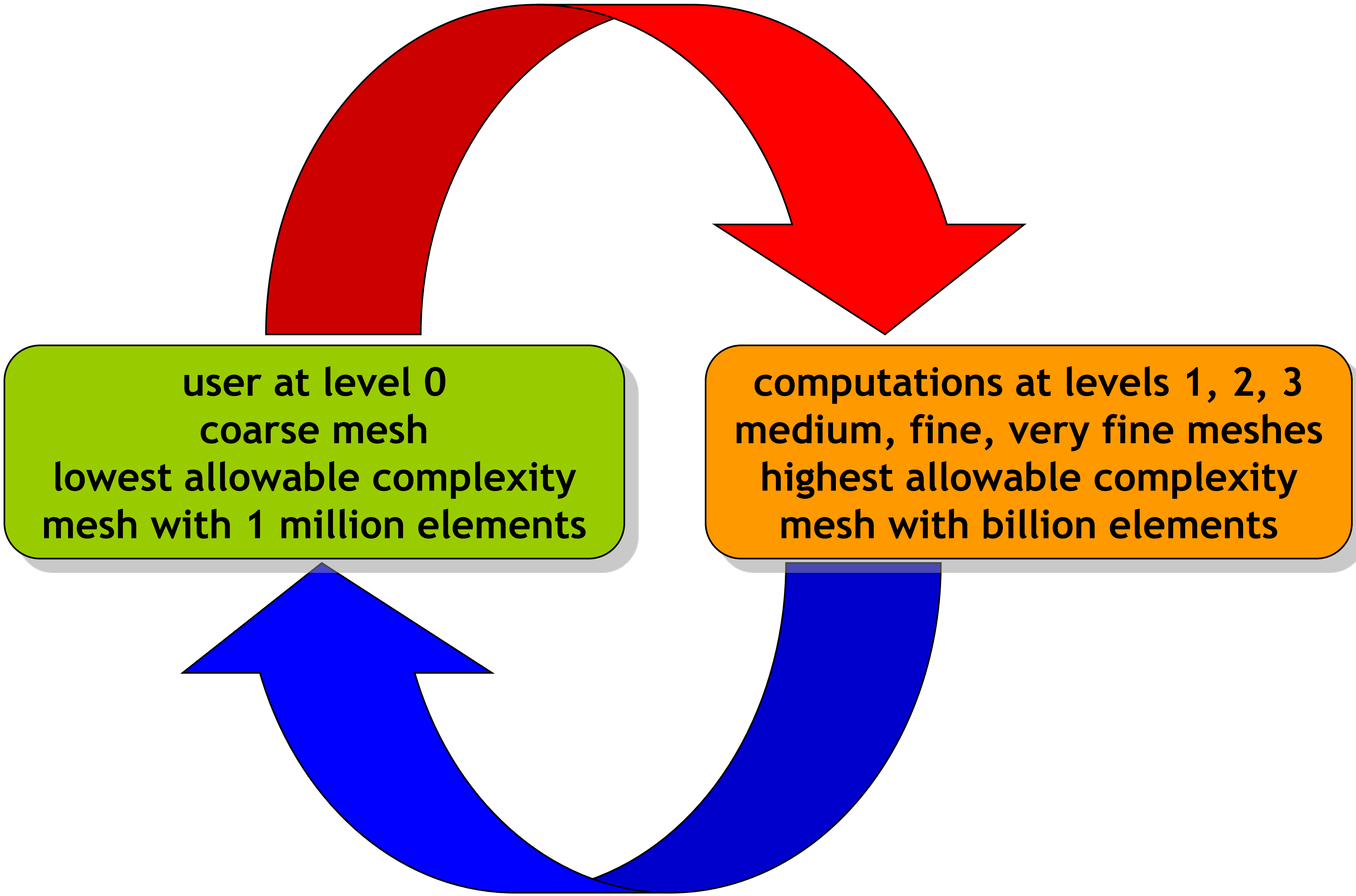
- Better representation of geometry.
- Better representation of physics.
- Improve accuracy.
- IMPROVE STABILITY OF HIGHER ORDER SCHEMES.

Approximate size of Mesh Data Set !
1 Billion elements:
Mesh: 64 Gigabytes Connectivity + 24 Gigabytes Coordinate
Solution (100 frame outputs): 4800 Gigabytes (4.8 Terabytes)

Variable Intensity Computational Environment (VICE)

- Computational environment which shields user from handling large meshes.
- User is interfaced ONLY with the original base-mesh (Level_0)
- Computation will be automatically be lunched (in parallel) at Level_1, Level_2, … without user being in the loop
- The accuracy of the computation is reflected in the highest intense level.
- Since the base-mesh data-structure is preserved, data is written (stored) to the disk only at base-mesh level (data compression).
- Visualization will be performed at base-mesh level.
- ….

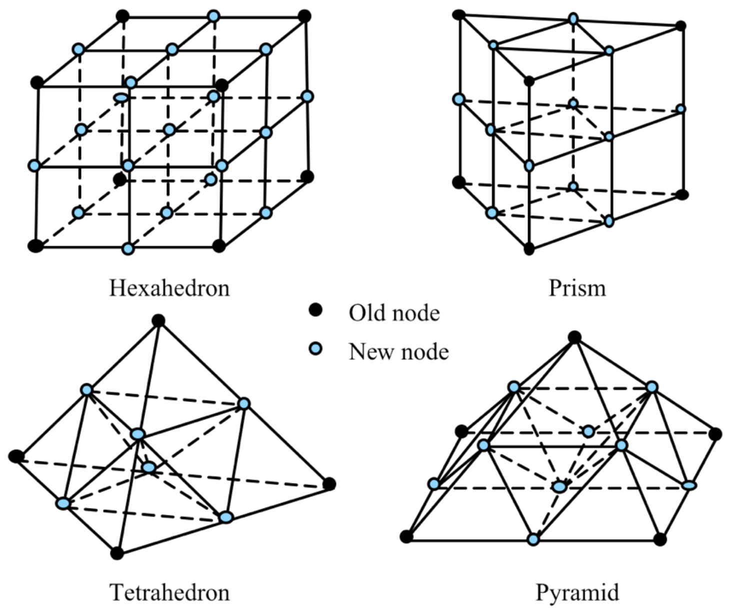
Variable Intensity Computational Environment (VICE)



Parallel Hybrid Mesh Multiplication

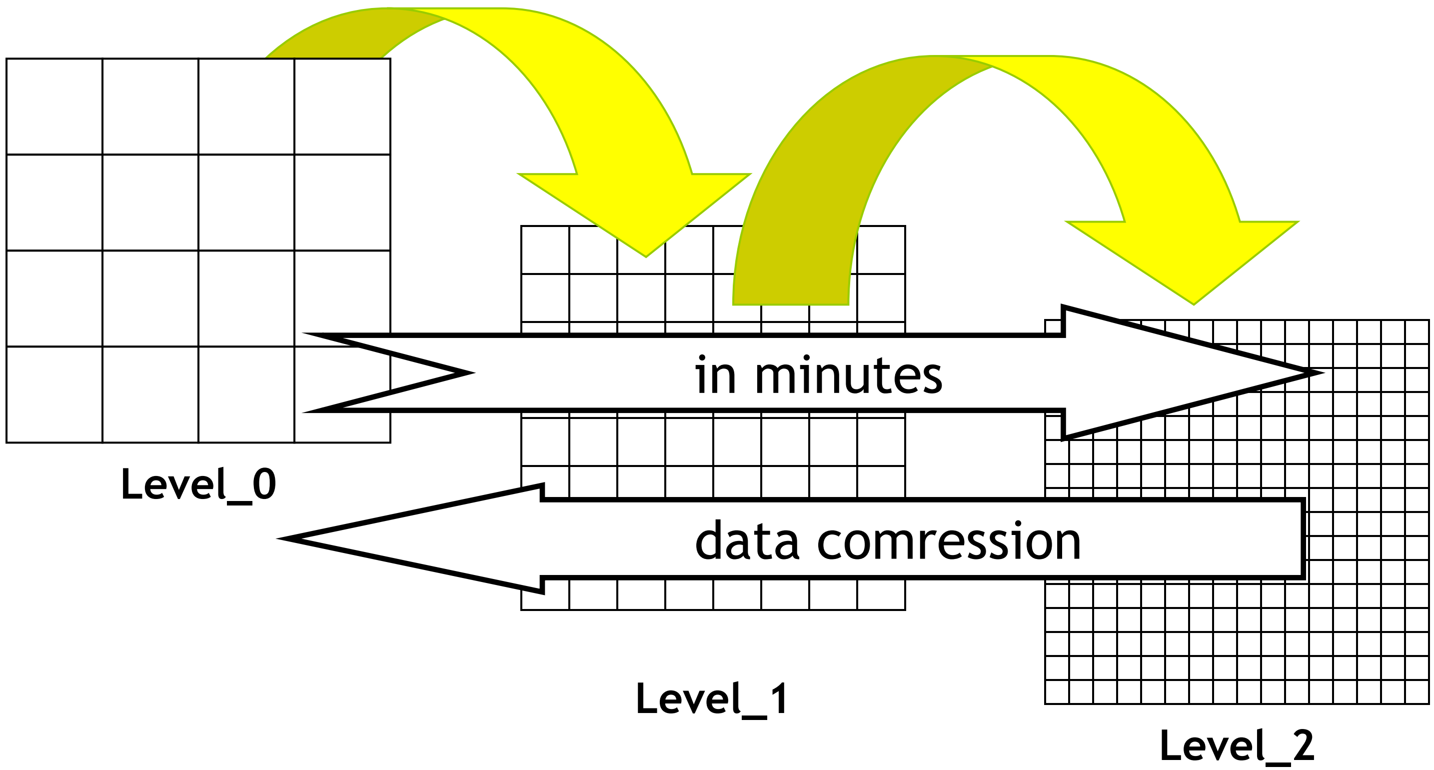
- Totally Unstructured Hybrid Based Mesh (Level_0)
- Subdivides the mesh to Level_1, Level_2, Level_3 in semi-structured way
- Preserves aspect ratio
- Totally Parallelizable
- Much Faster
- Access to Distributed Memory to Generate Very Large Meshes

Isotropic subdivision of elements of various types	
Parent element	Child elements
Hexahedron	8 child hexahedral
Prism	8 child prisms
Pyramid	6 child pyramids + 4 child tetrahedral
Tetrahedron	8 child tetrahedral



Parallel Hybrid Mesh Multiplication

- Starts with the *base mesh* (Level_0)
- Subdivides Level_0 mesh to Level_1
- Can be repeated to higher levels



Surface Correction

Problem:

- Original multiplication subdivides linearly on all mesh domain,
- No access to original CAD surface geometry during subdivision,
- Curved solid surface feature is lost from one level to other,
- Hence, geometry is still at coarse level even though mesh is finer,
- Therefore, original accurate surface geometry should be recovered.

Remedy:

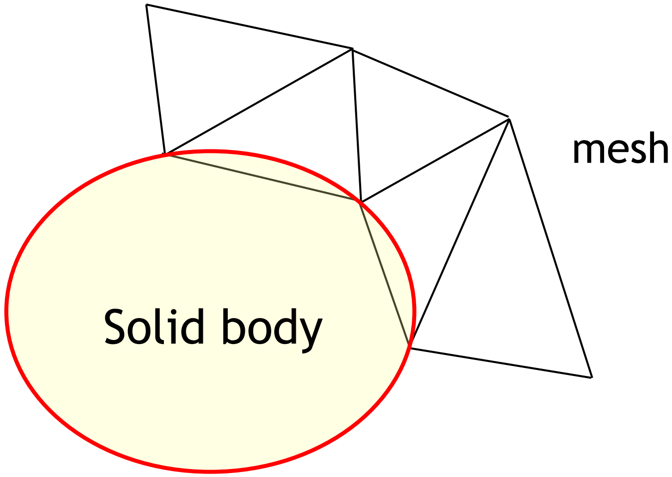
- Generate a fine master surface mesh,
- After each subdivision, map linearly subdivided surface mesh on to fine master surface mesh.

Surface Correction

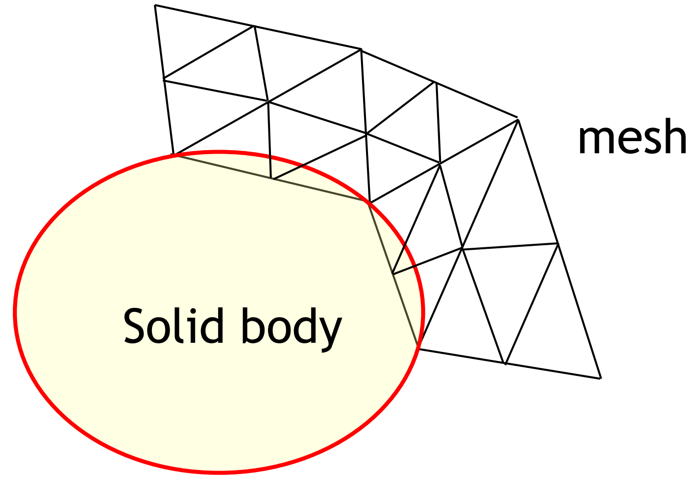
Process:

- Generate a fine master surface mesh,
- For all surface nodes of linearly subdivided mesh, search for nearest surface node/element in the fine master surface mesh, KD-tree search data structure is used
- Find nodal displacement of linearly subdivided mesh nodes
- Move surface mesh nodes to new positions
- Use mesh deformation for internal nodes (spring-analogy is used)

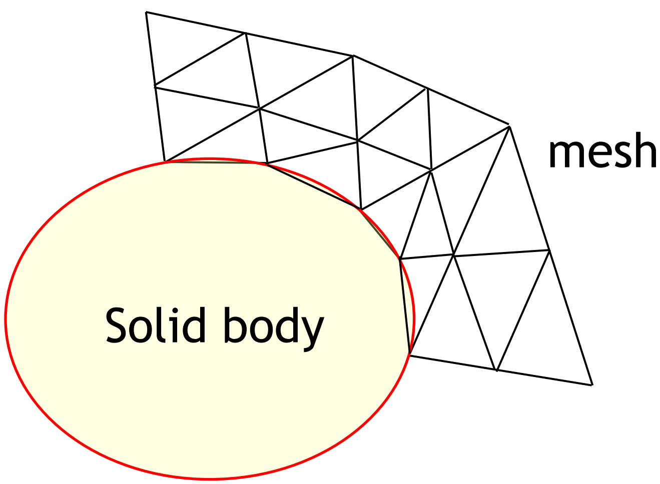
Surface Correction Steps



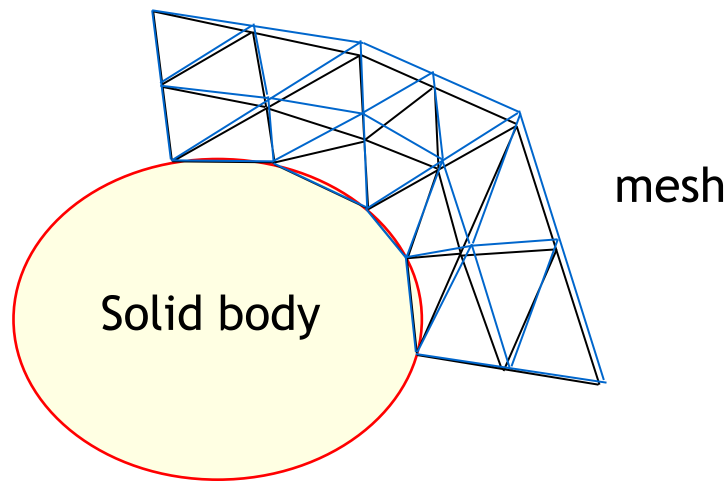
1) Coarse mesh, L0



1) Subdivided mesh□, L1



3) Surface correction□, L1



4) Move Internal mesh, L1

Parallel Clusters

NGC @JSU owns and maintains two clusters and has access to the Army Supercomputing Resources.

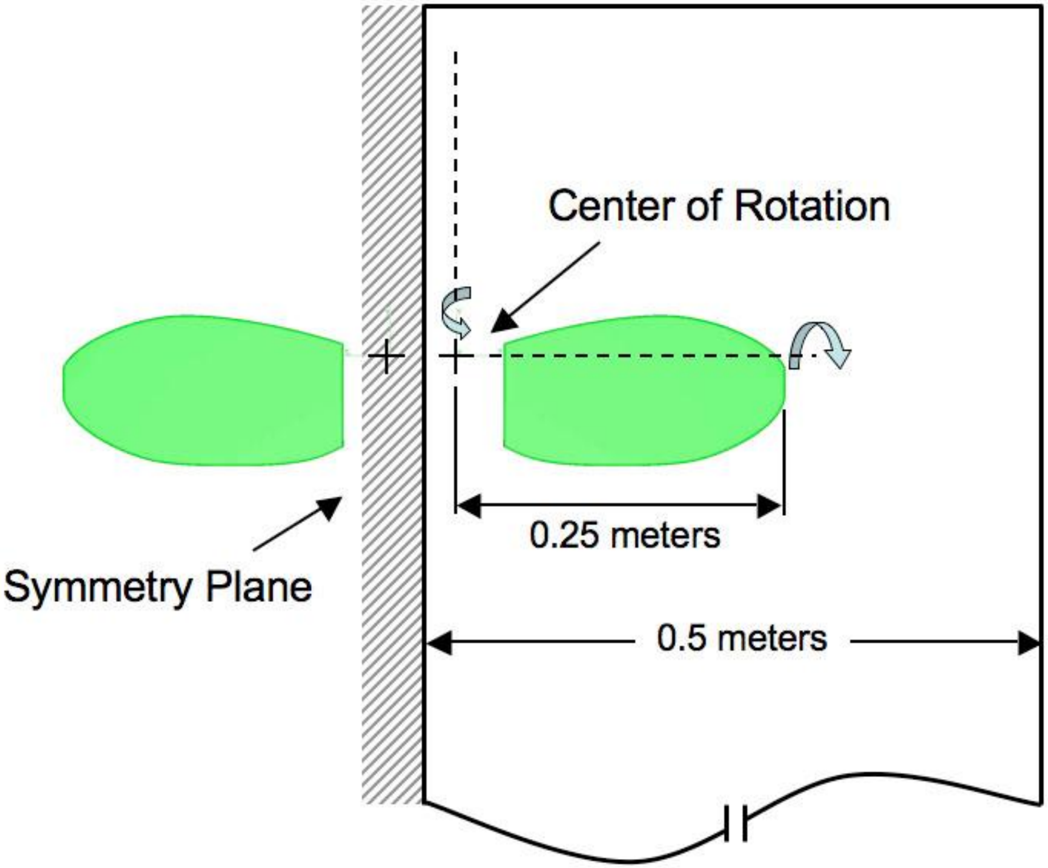
○**Cluster 1, HPC-1: (used for parallel performance study)**

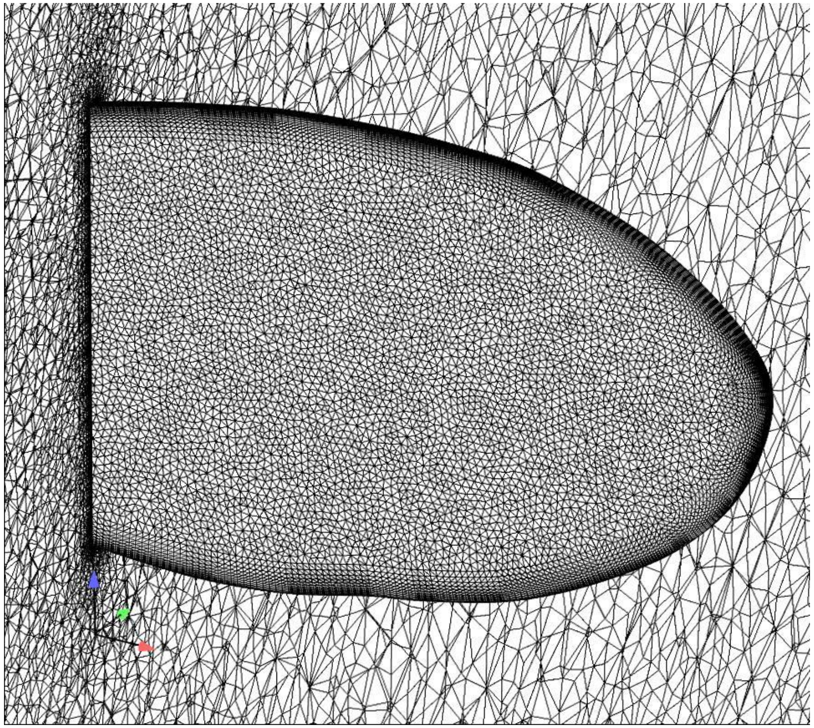
- Beowulf Cluster by PSSC Labs,
- 10 nodes/80 cores, 320GB Memory, 6TB HD, Xeon e5450 3Ghz
- Infiniband III Lx

○**Cluster 2, HPC-2: (used for long runs and large data sets)**

- SUN Fire X4200 server, X4540 data server, and X2200 compute nodes,
- 64 nodes/512 cores compute nodes, 1TB Memory, 48TB HD data server,
- Opteron 2354 2.2Ghz

Flapping Wing: Robofly





Flapping Wing: Motion

SWING ANGLE

$T_{shift} = 0, \ T_1 = 0, \ T_3 = 0$

$T_2 = T_4, \ \phi_{\min} = -\phi_{\max}$

$T_{total} = T_2 + T_4$

PITCH ANGLE

$T_1 = 0, \quad T_3 = 0$

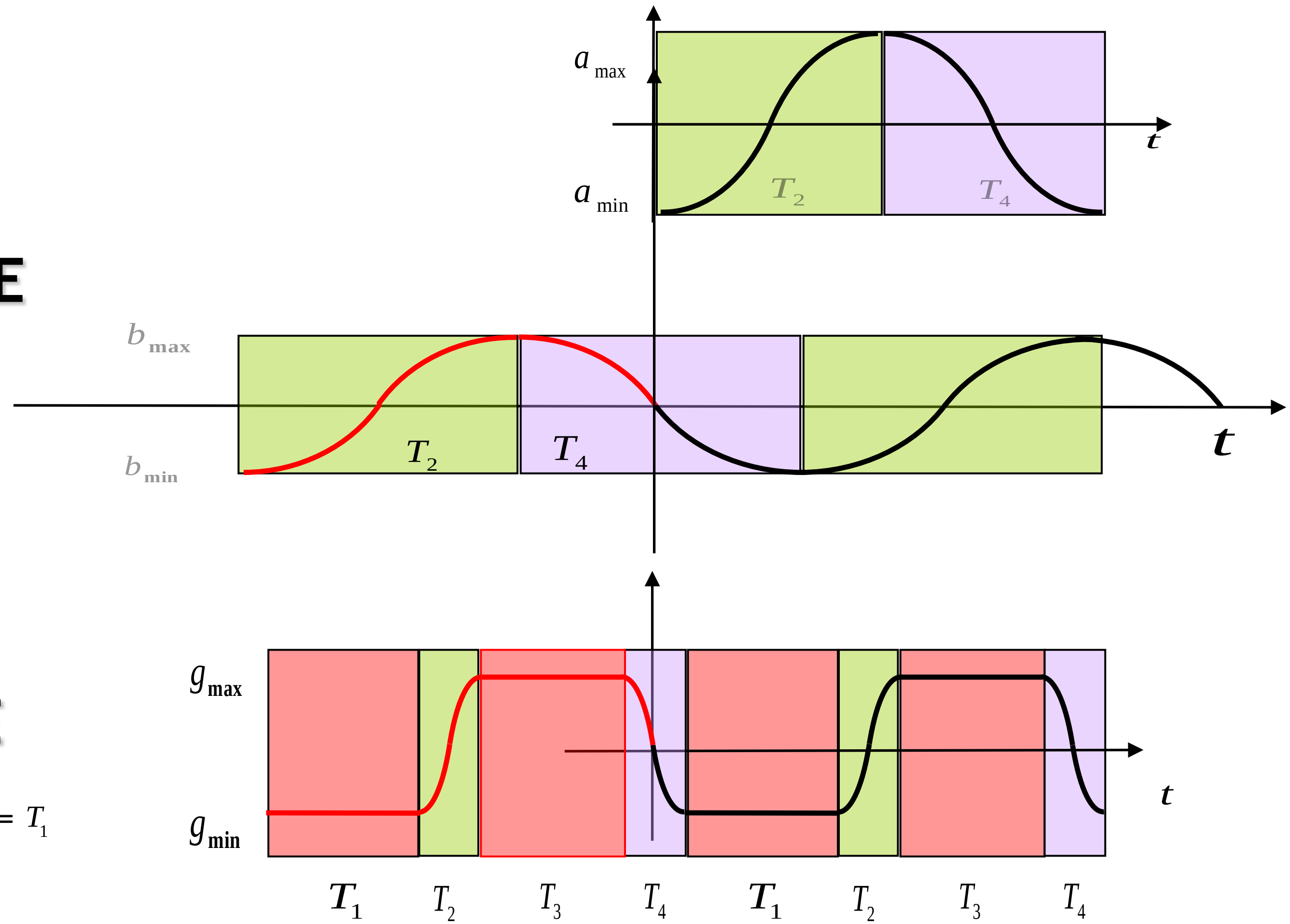
$T_4 = T_2, \quad T_{shift} = T_2 + \frac{T_4}{2}$

$\beta_{\max} = -\beta_{\min}$

ROLL ANGLE

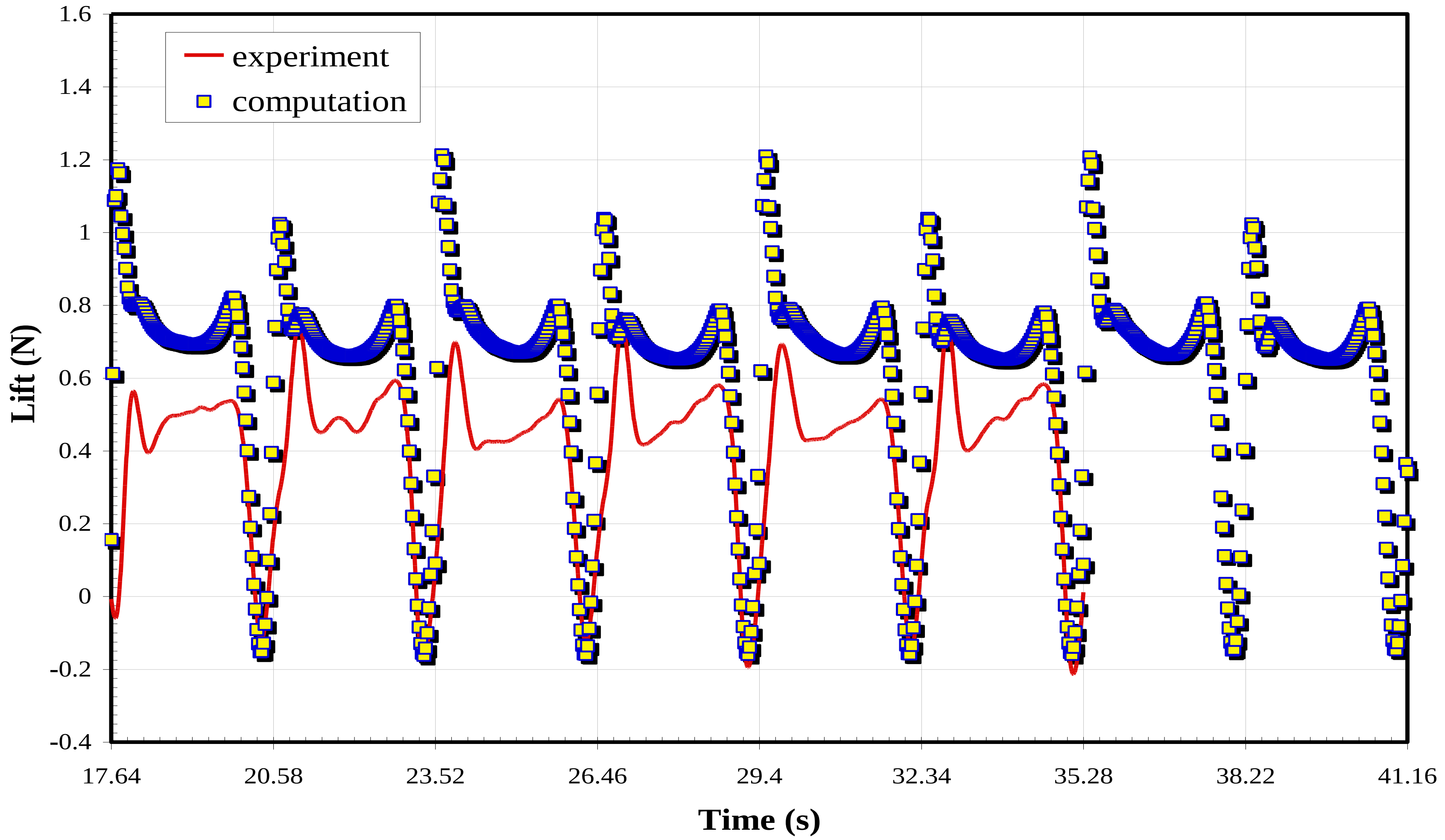
$T_{shift} = T_1 + T_2 + T_3 + \frac{T_4}{2}, \quad T_3 = T_1$

$T_4 = T_2, \quad \alpha_{\min} = -\alpha_{\max}$



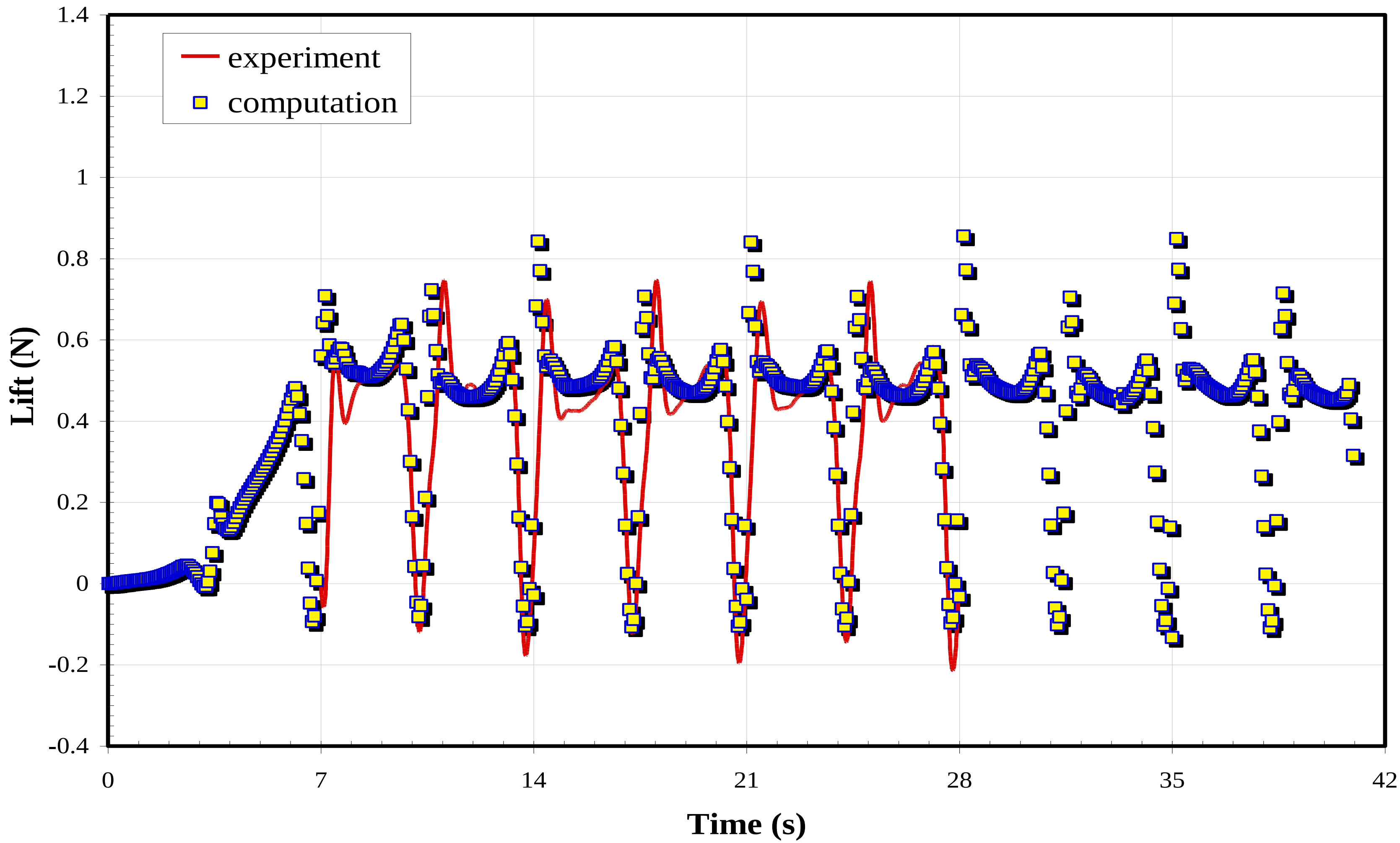
Flapping Wing: solution

We designed a motion of the wings to try to reproduce as best we could the motion curves in Dickinson 1999 which were not based on simple sinusoidal rotations.



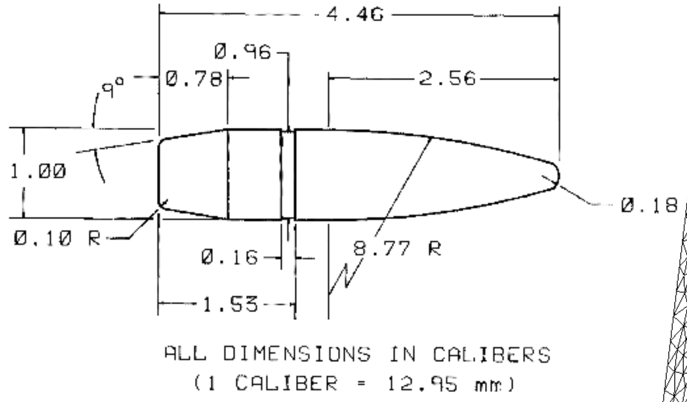
Flapping Wing: solution

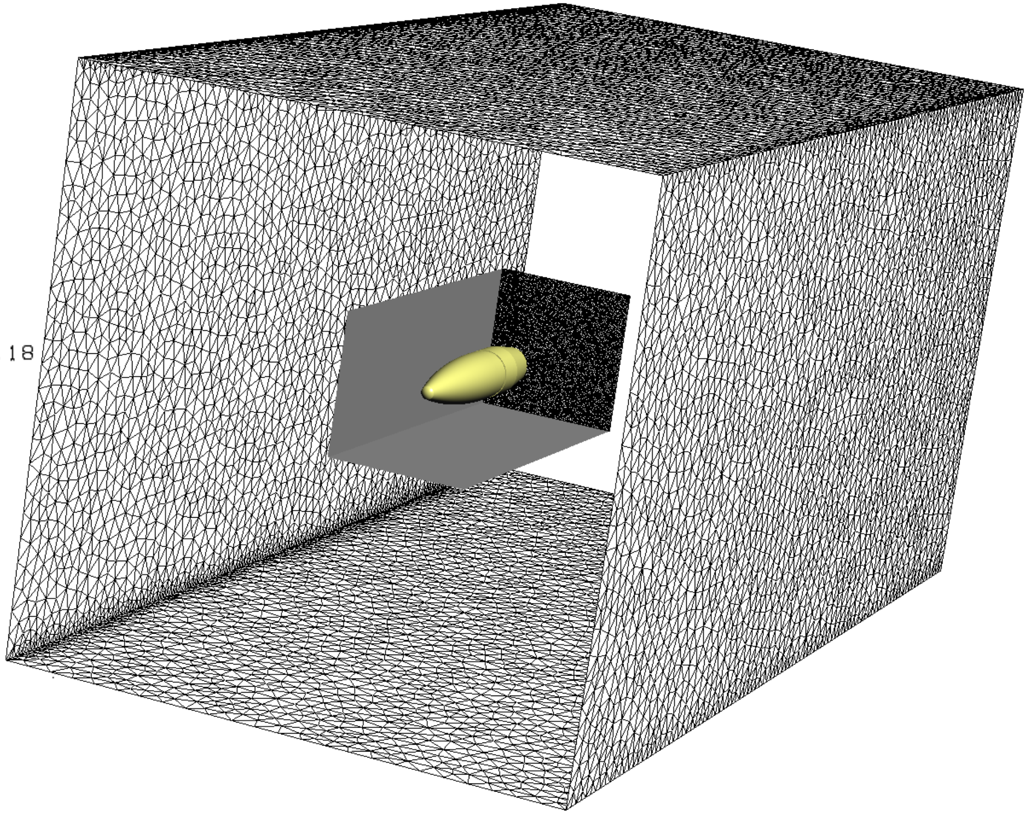
Results with Different Frequency



Projectile: geometry and overset mesh

Two hybrid mesh blocks are appended to form a single block for the flow solver and SUGGAR in the same order to match one-to-one index mapping.

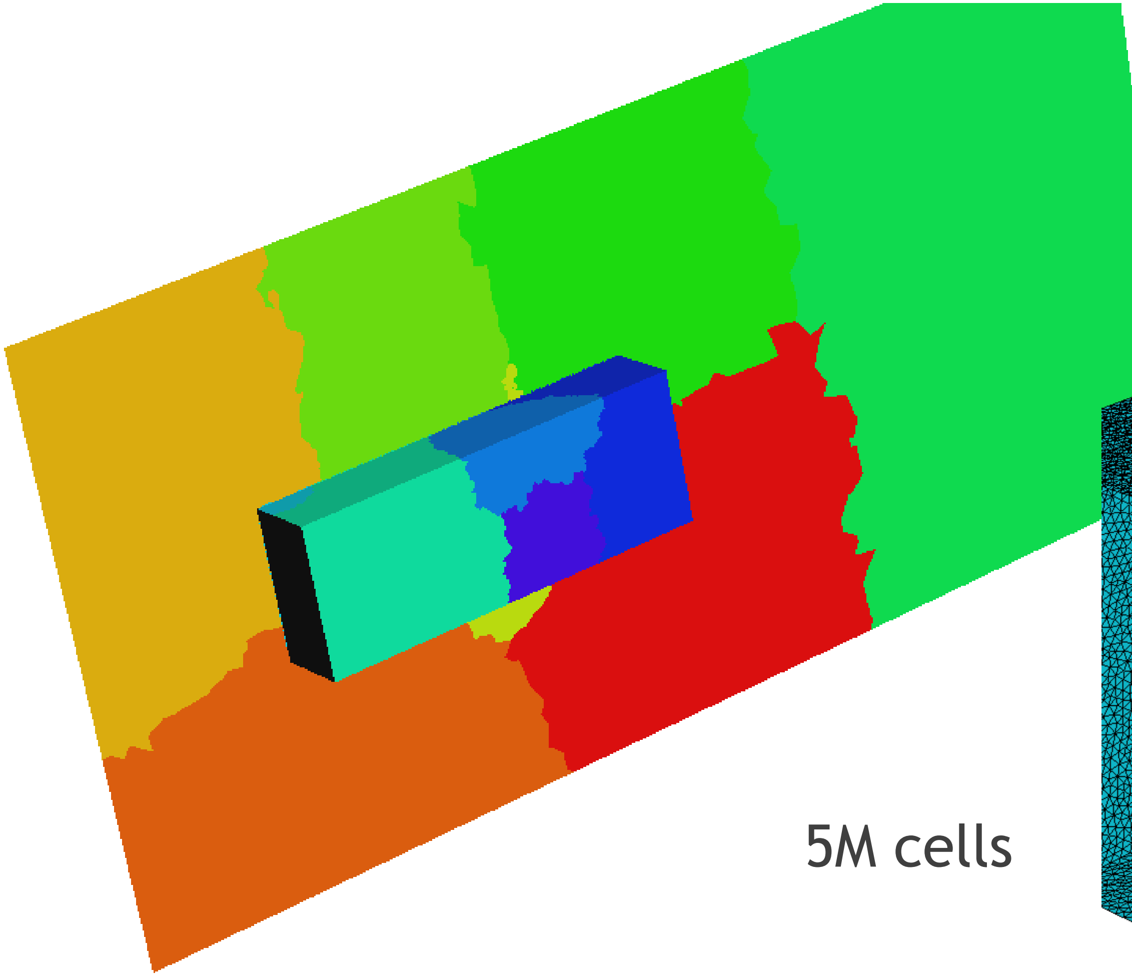




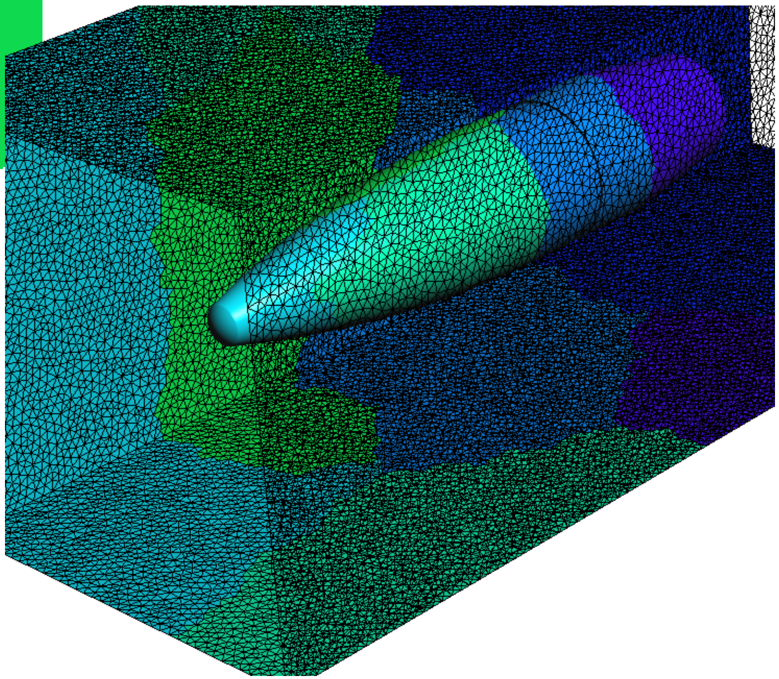
5M cells

Projectile: mesh partitions

Mesh partitions:
METIS parallel partitioner.
It is done inside the code.
Global partition mapping
is used by DiRTlib.

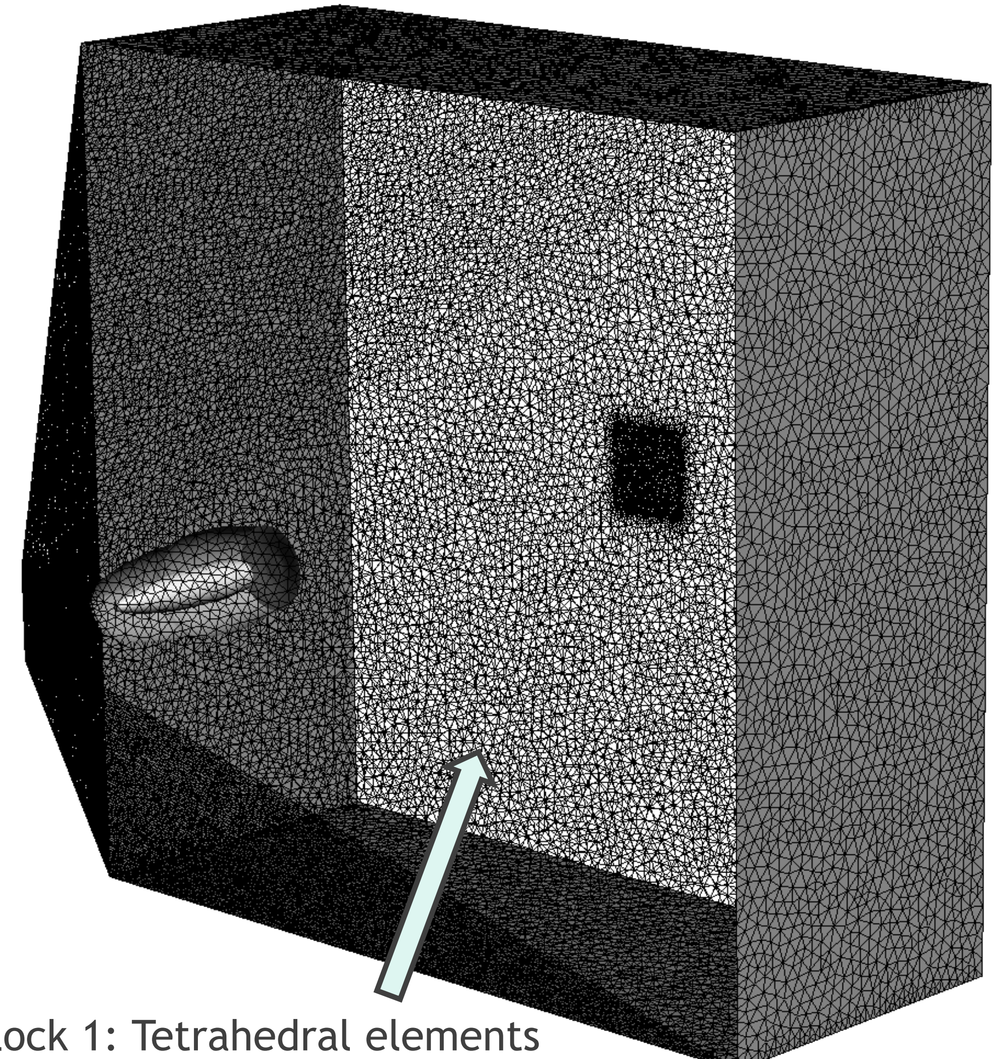


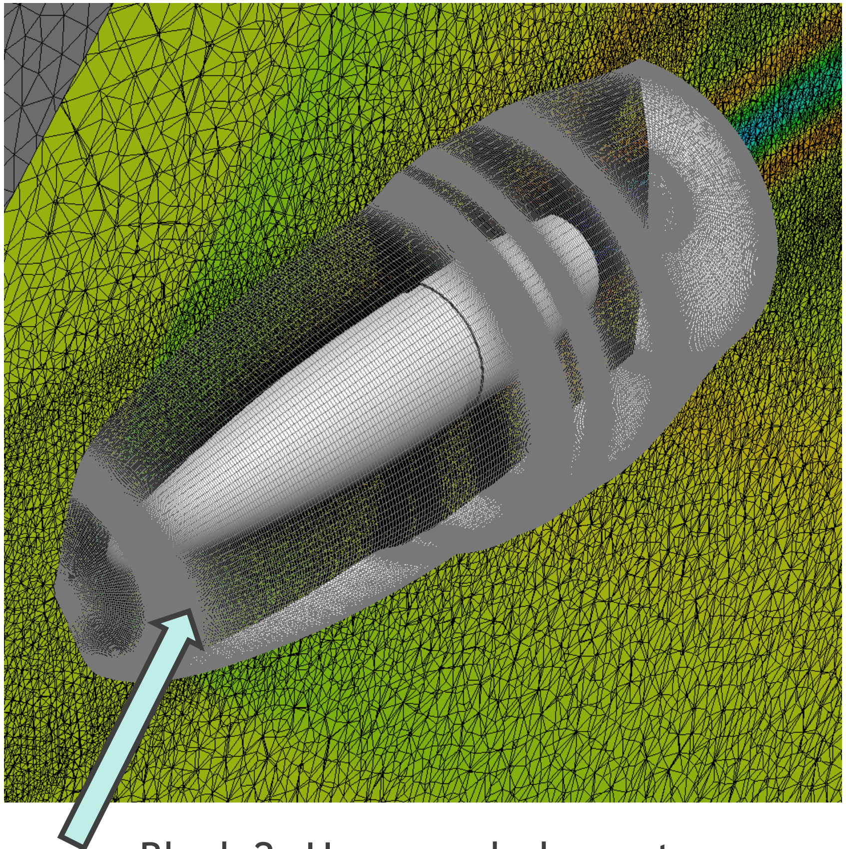
5M cells



Projectile: finer mesh

26 M Cells



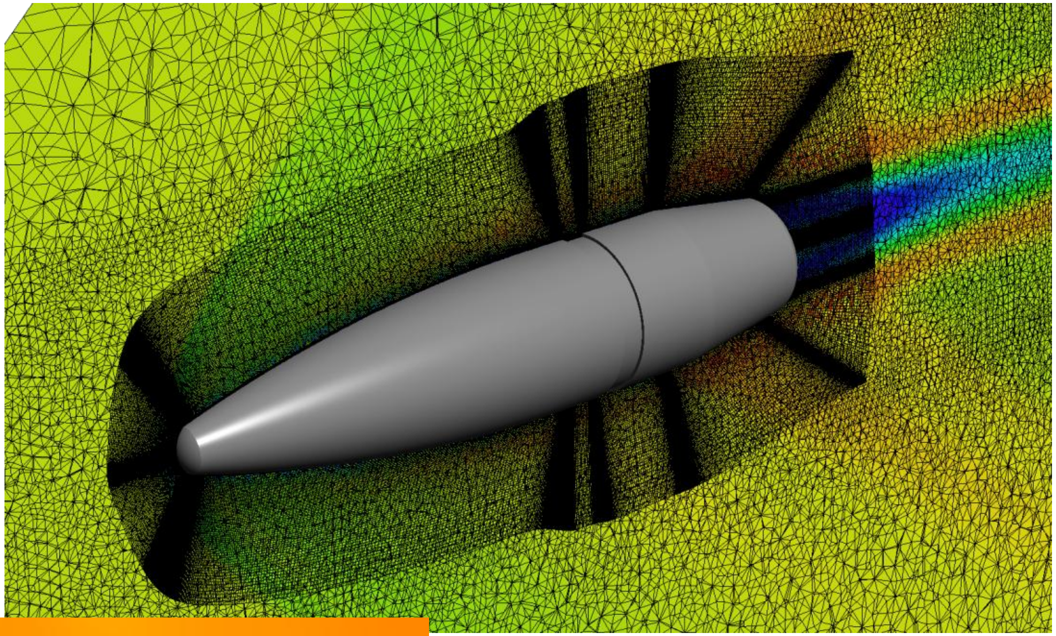


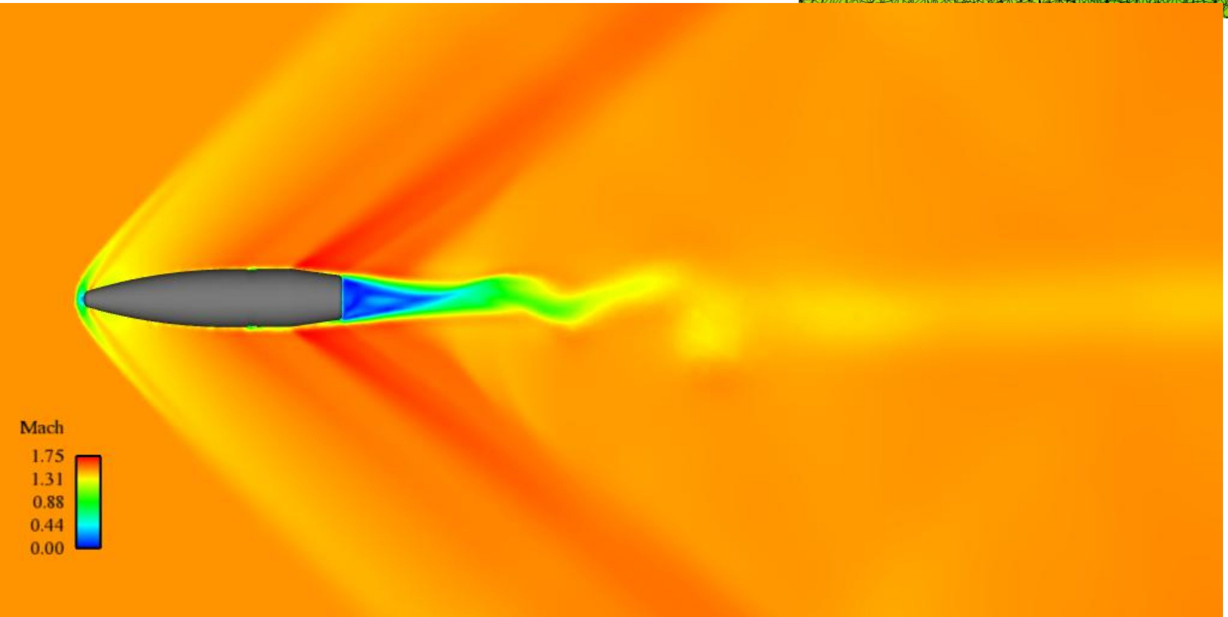
Block 1: Tetrahedral elements

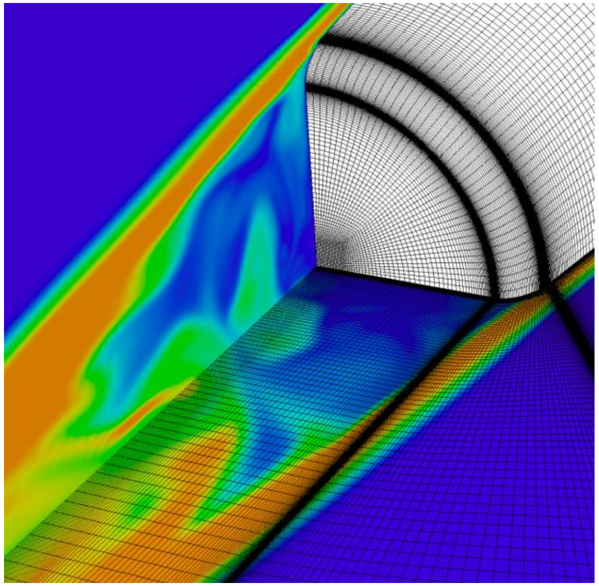
Block 2: Hexagonal elements

Projectile: finer mesh solution

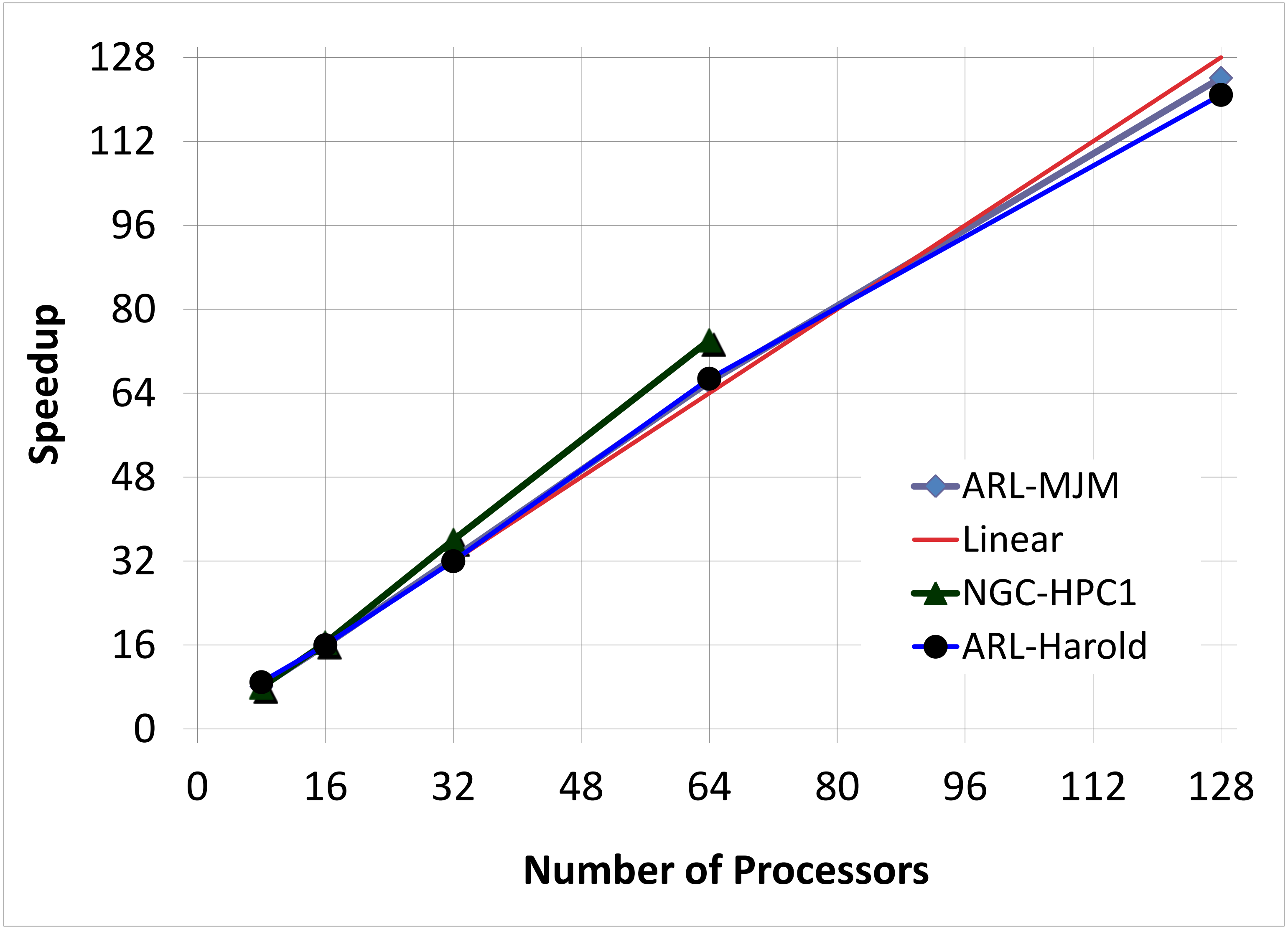
26 M Cells
Second order solution captures unsteady wake field at zero AoA



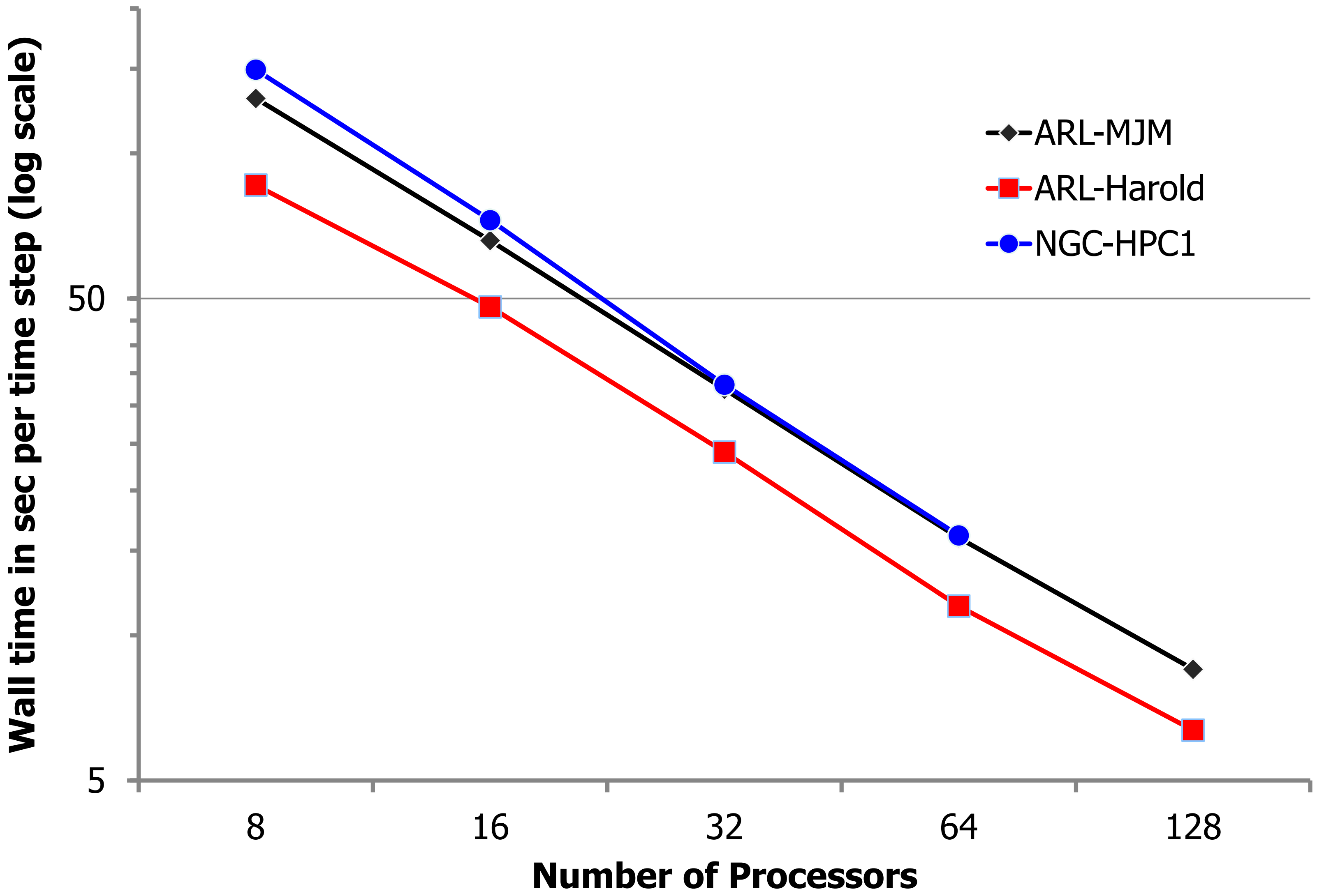




Parallel performance



Parallel timing



Parallel timing

NGC-HPC1

Processors	Overset Update	Communication	Navier-Stokes	Turbulence	Total
8	0.32	1.08	226.44	70.66	298.51
16	0.17	2.53	110.81	31.92	145.43
32	0.09	1.92	51.46	12.78	66.25
64	0.65	1.76	24.11	5.73	32.24

ARL-MJM

Processors	Overset Update	Communication	Navier-Stokes	Turbulence	Total
8	0.36	0.50	103.23	25.98	130.08
16	0.21	1.10	52.57	12.09	65.98
32	0.11	0.86	26.04	5.41	32.41
64	0.06	0.67	12.67	2.55	15.95
128	0.04	0.63	6.52	1.33	8.51

ARL-Harold

Processors	Overset Update	Communication	Navier-Stokes	Turbulence	Total
8	0.28	0.25	68.56	16.90	85.98
16	0.17	0.64	37.99	9.24	48.04
32	0.09	0.66	19.18	4.08	24.01
64	0.05	0.90	8.86	1.69	11.51
128	0.03	0.73	4.71	0.89	6.36

Timing decomposition of CaMEL Aero with overset module among major modules of the code.

* Timing for Overset comprises only DiRTlib NOT SUGGAR time. All time values are in seconds.

Timing for SUGGAR

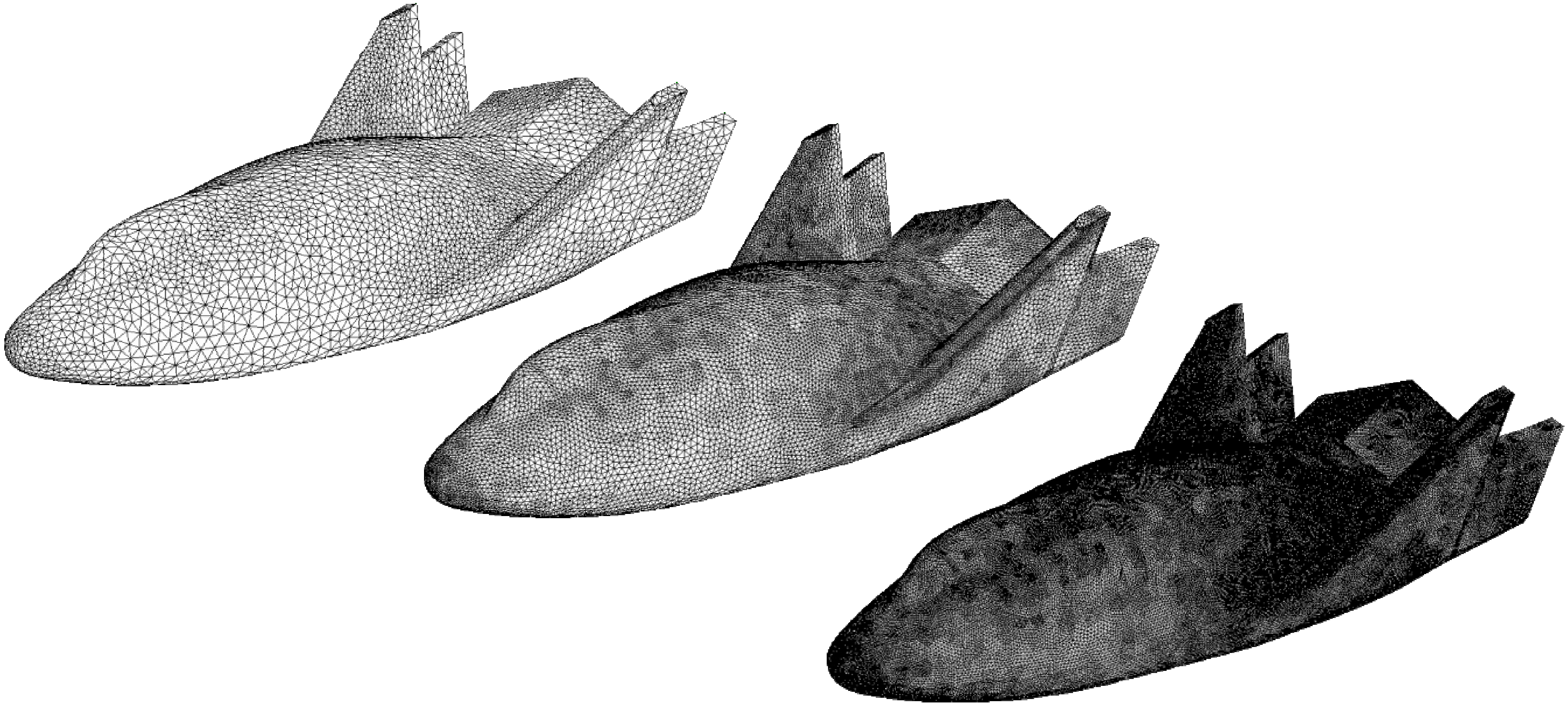
Suggar functions	1 thread	8 threads
Parsing Input File	13.32	13.22
Computing gen grid cell centers and verifying for grid block1	12.06	12.04
Computing gen grid cell centers and verifying for grid block2	11.58	11.21
filling Donor Search octree	11.05	12.22
Octree Setup	12.05	**6.97**
marking outer fringe 1	3.23	3.55
marking outer fringe 2	3.62	3.95
marking inner fringe 1	2.98	3.31
marking inner fringe 2	3.63	3.95
finding donor cells	8.91	**4.59**
Performing Overset Assembly	27.02	**17.57**
freeing memory	2.04	2.08
Others (total wall time minus all of the above)	*-33.69*	*-31.42*
Total Wall Clock Time	**77.8**	**63.24**

Mesh Size 5M elements
Blk#1: 2,339,820
Blk#2: 2,703,584

Larger mesh is computationally expensive
Suggar Time for 26M cells 442 sec (1thread)
(5.7 times of 5M cells, for 50M ~ 15min)

Mesh Subdivision for Generic Reentry Vehicle

	Number of Elements	Number of Nodes	Subdivision Time (8procs)
Level 0: starting mesh	442,290	74,495	-
Level 1	3,538,320	600,498	1.77 s
Level 2	28,306,560	4,776,194	14.2 s



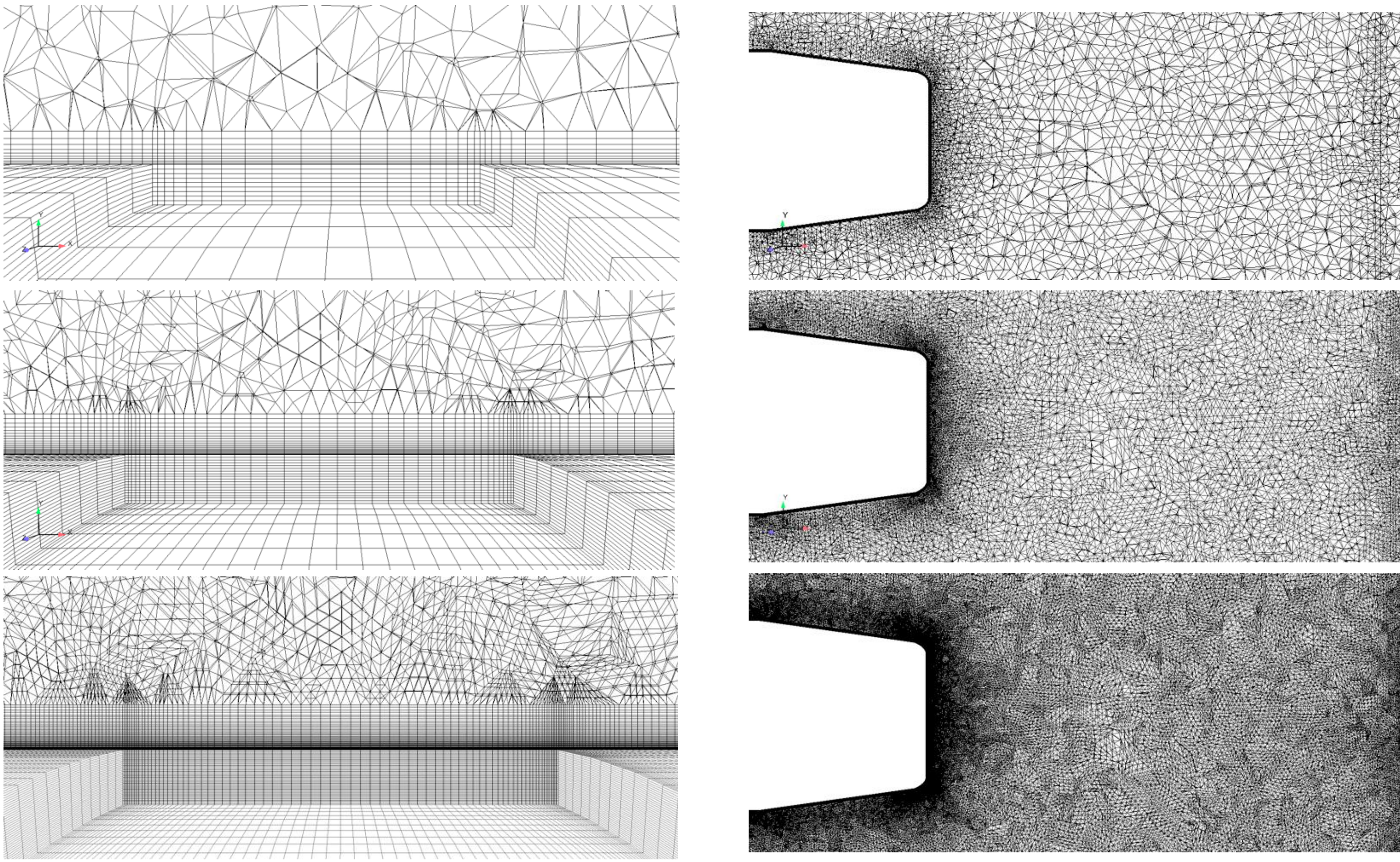
Mesh Subdivision for Projectile

Grove section | Boat-tail and base section

Level_0

Level_1

Level_2



Projectile with L/D=4.46, with grove, and boat tail.

Mesh Subdivision for Projectile

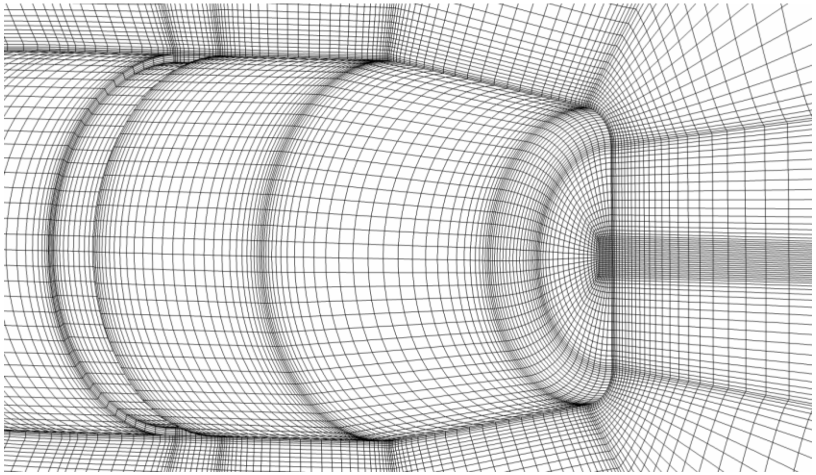
UNSTRUCTURED MESH

	Number of Elements	Number of Nodes	Multiples of Mesh Size	Subdivision Time For 8 procs, in second	Multiples of Subdivision Time
Level 0	1,120,899	199,962		-	-
Level 1	8,969,168	1,573,738	8.0018	5.14	-
Level 2	71,765,200	12,468,606	8.0013	38.79	7.55
Level 3	*574,192,736*	*99,228,682*	*8.001*	*51.54*	--

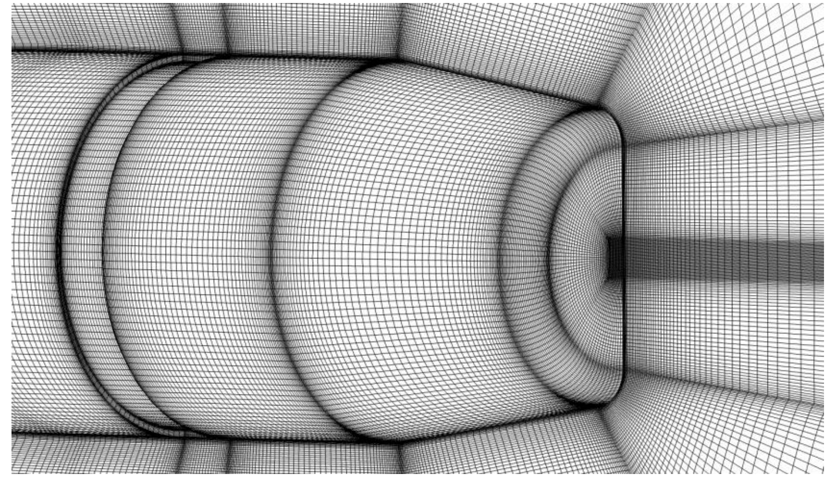
STRUCTURED MESH

	Number of Elements	Number of Nodes	Multiples of Mesh Size	Subdivision Time For 8 procs, in second	Multiples of Subdivision Time
Level 0	619,652	635,664	1	-	-
Level 1	4,957,216	5,021,078	8	2.46	-
Level 2	39,657,728	39,912,810	8	28.22	11.47

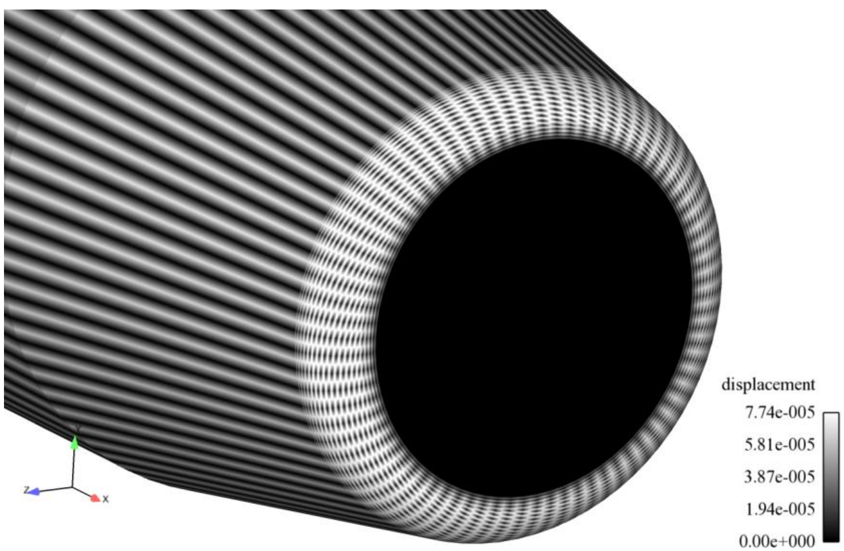
Projectile mesh surface correction



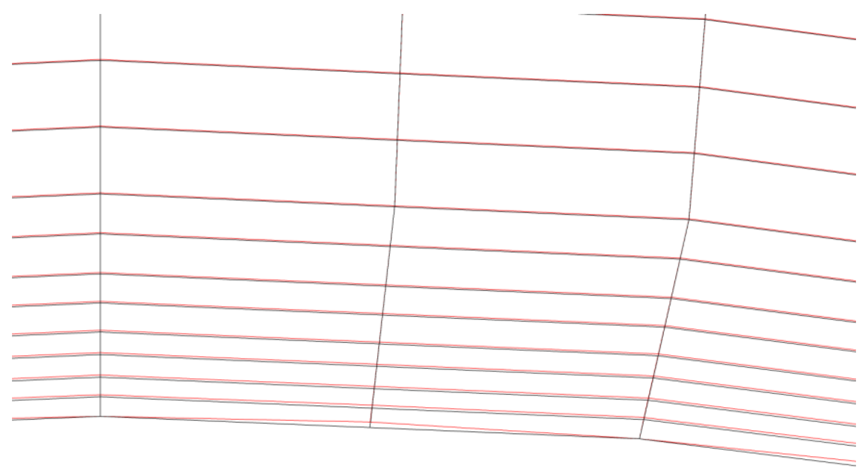
Level 0 mesh



Level 1 mesh

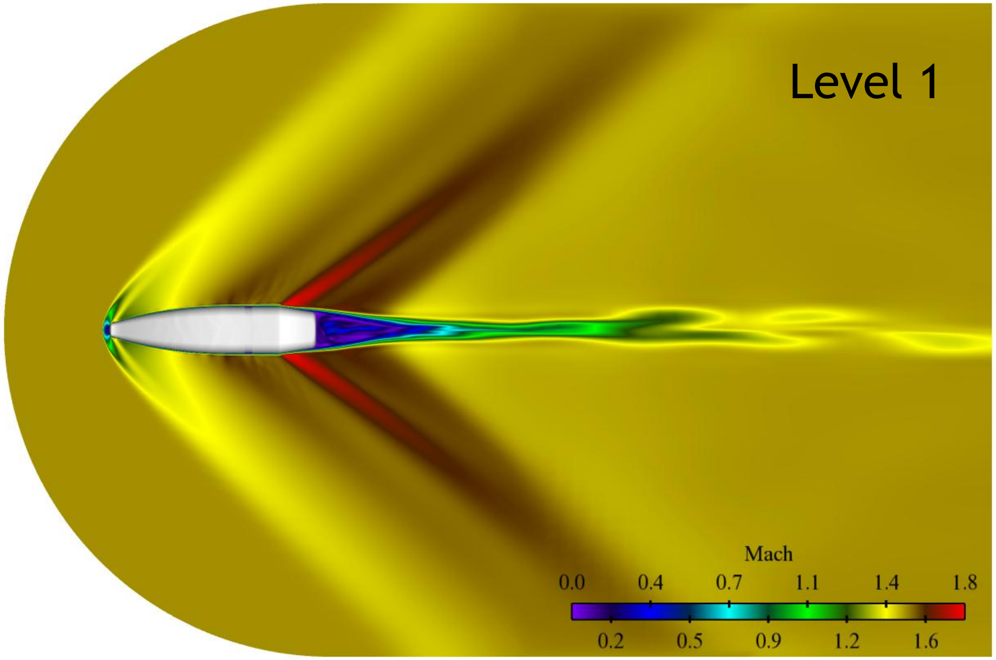


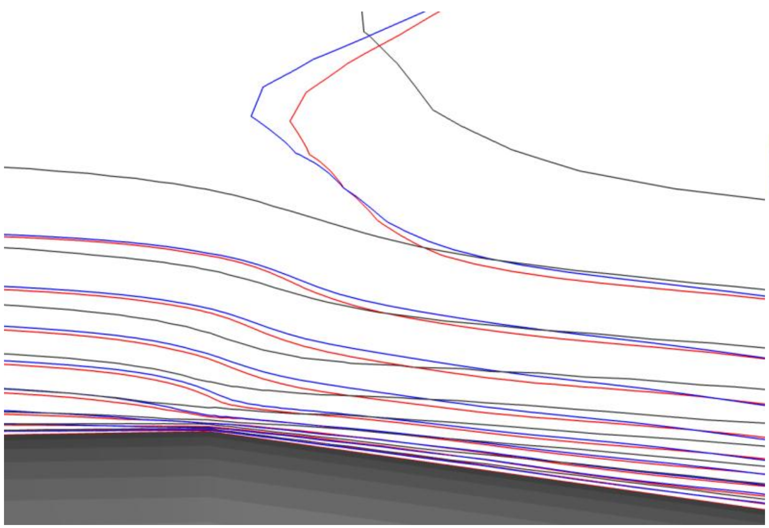
Surface displacement, Level 1

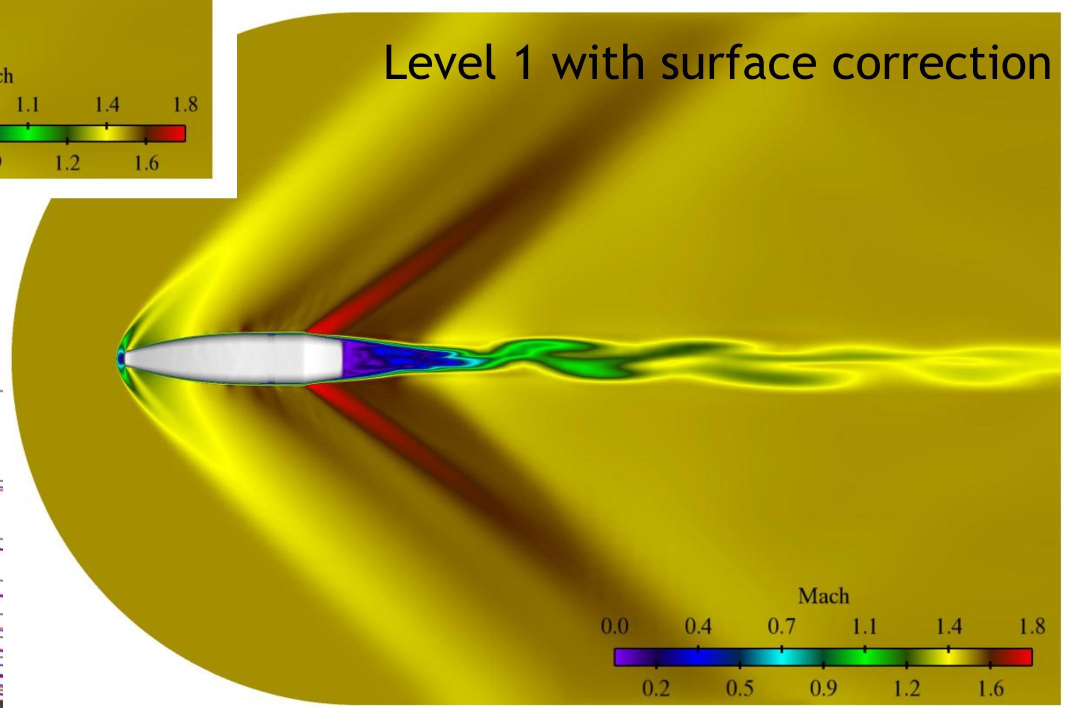


Near body mesh deformation, Level 1

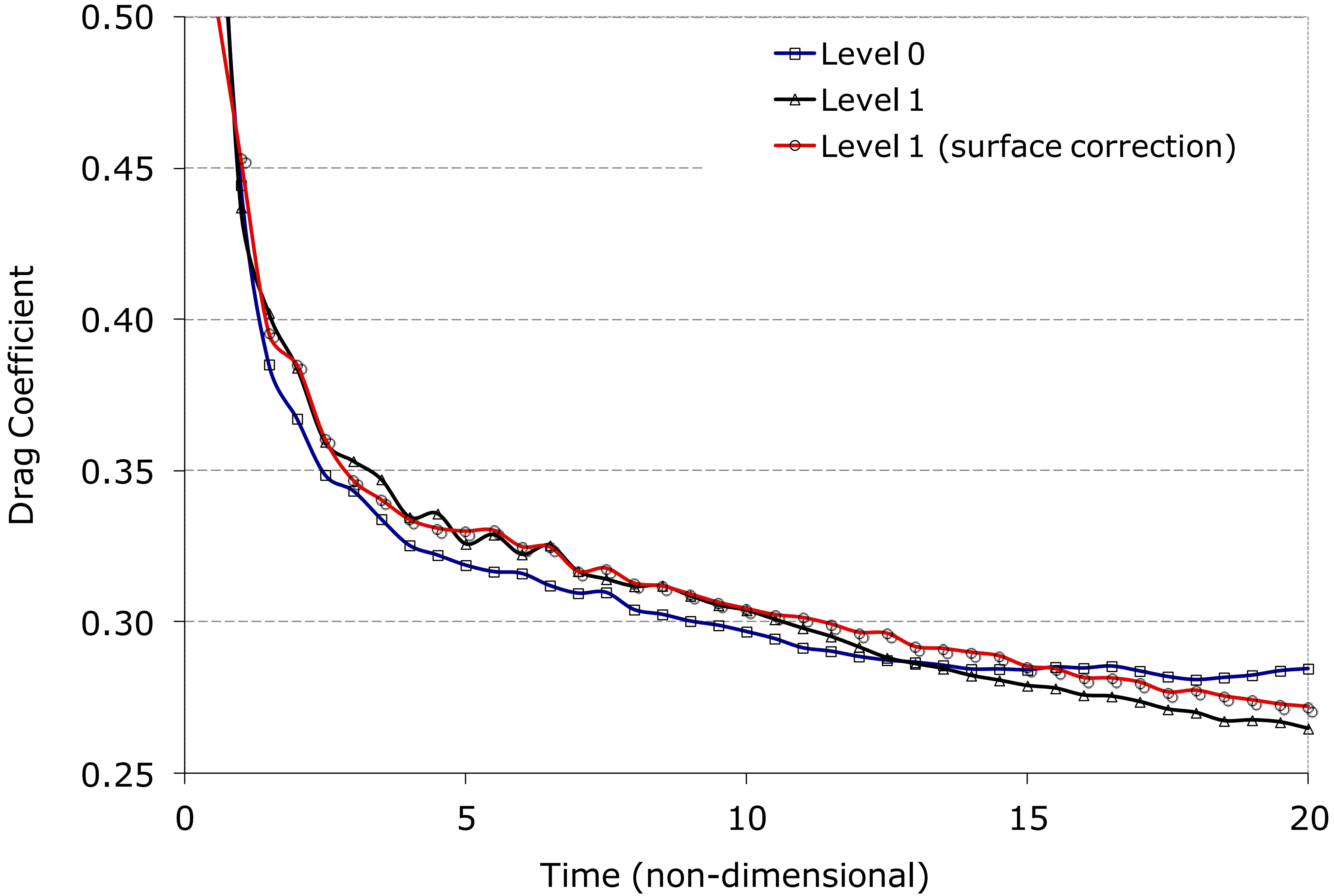
Projectile flow solution







Projectile flow solution



Future Works

- FSI Technology
  - Air Drop Delivery Systems
  - Biomedical Applications
- Long Term Environmental Modeling
  - Nuclear Risk Assessment
  - Oil Spills Simulations
  - Iceberg Motions for Navy Ships

THANK YOU

QUESTIONS ?