Implementation of An Overset Capability To CaMEL Aero Flow Solver And Its Applications To Moving Bodies

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

Jubaraj Sahu
Aerodynamics Branch, U.S. Army Research Laboratory

Patrick Collins
Lockheed martin / Army Research Laborartory DoD Supercomputing Resource Center

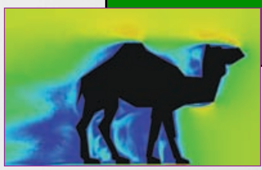
Sponsors
Northrop Grumman Ship Building
Army Research Office
Army Research Laboratory

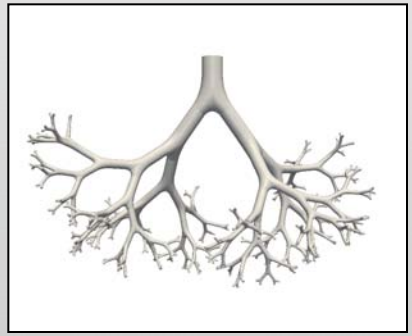
10th Symposium on Overset Composite Grids and Solution Technology, September 20-23, 2010,
NASA Ames Research Center, Moffett Field, CA,USA

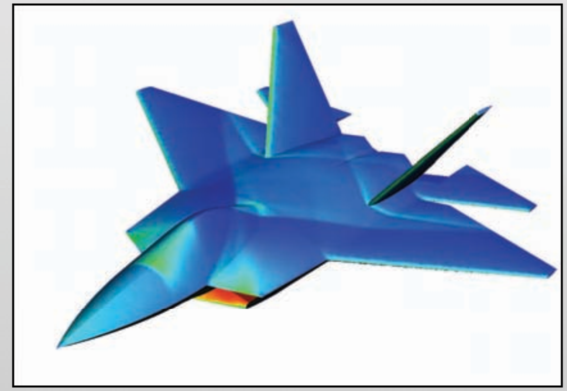
- Overview of CaMEL Solvers
- SUGGAR/DiRTLIB implementation
- Results/Flow Solutions
- Conclusions

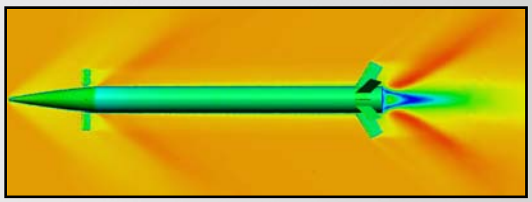
CaMEL Flow Solver

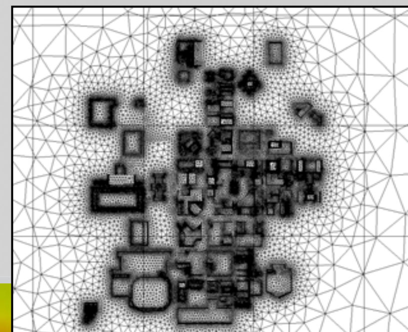
CaMEL Flow Solvers

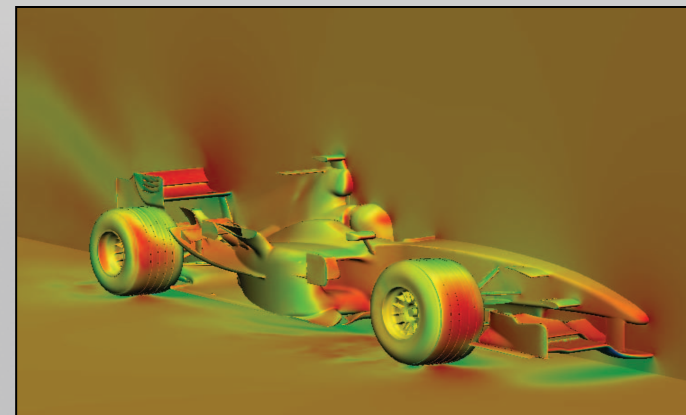


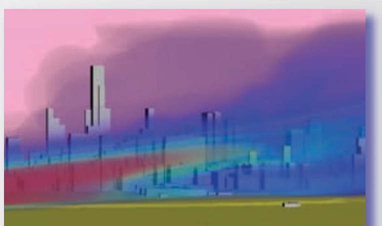


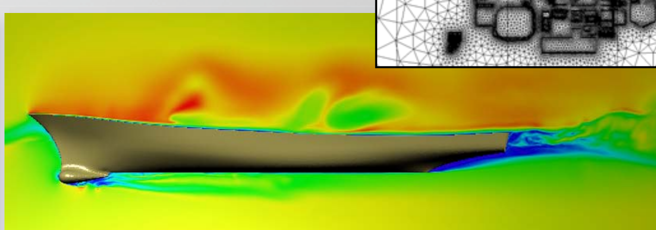








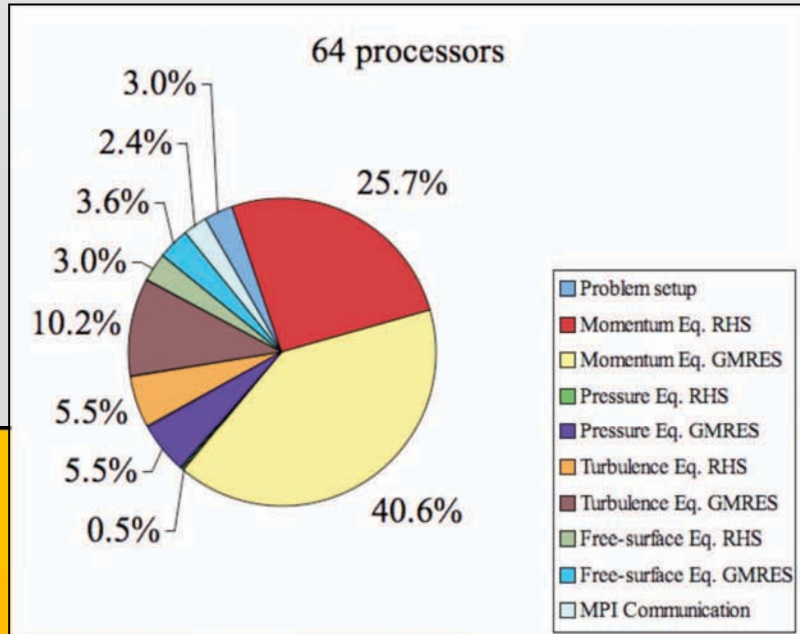


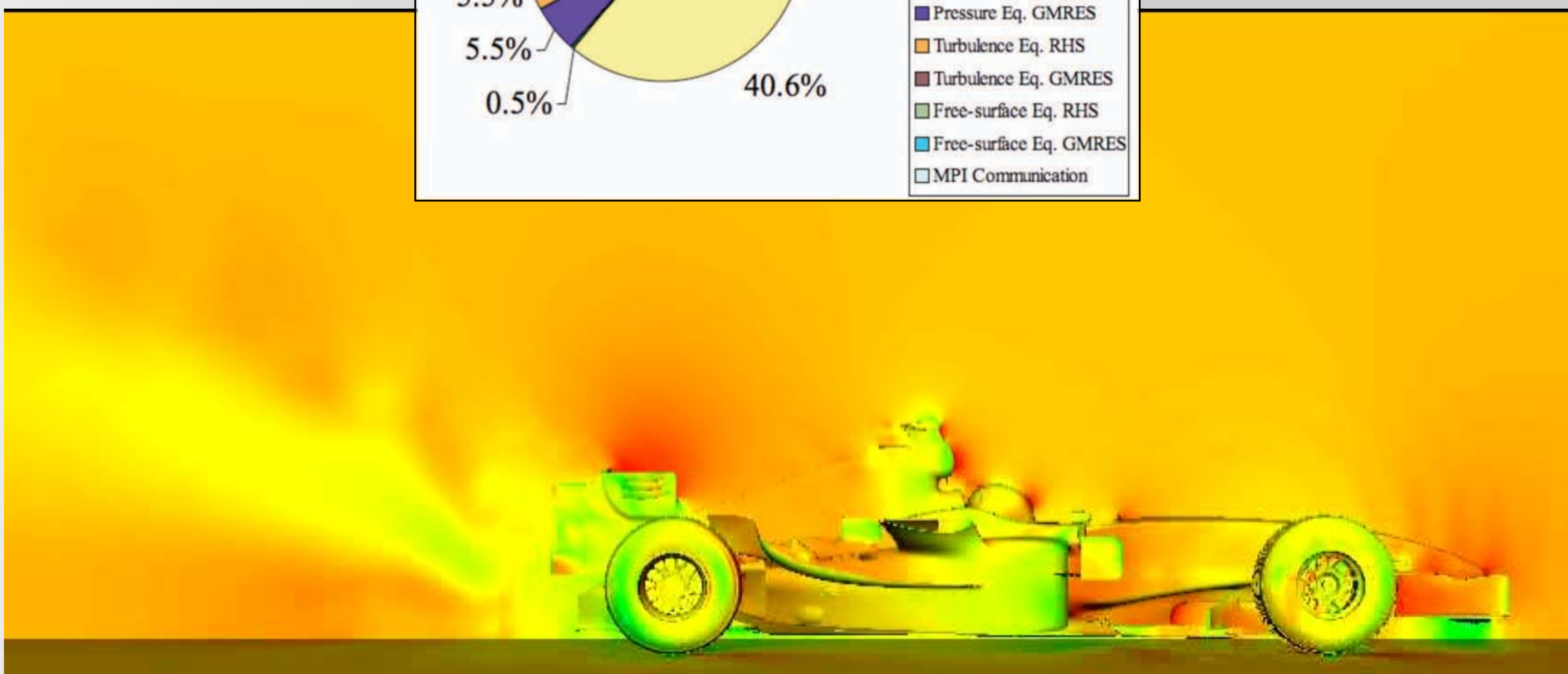


CaMEL^{CHH}
finite element / volume hybrid formulation for incompressible (single / two fluid) flows with heat and mass transfer.

CaMEL^{Aero}
finite volume cell-centered flow solver for compressible Euler and Navier-Stokes equations on hybrid meshes.

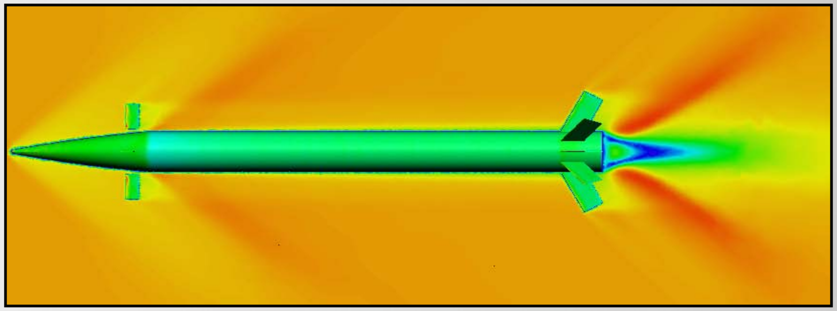
CaMEL Flow Solver

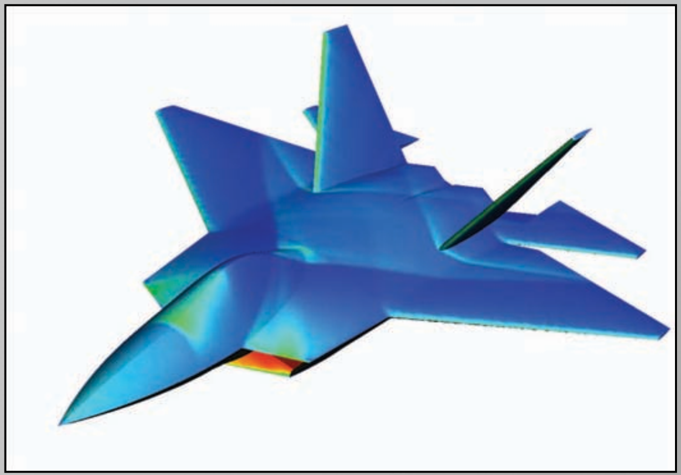




CaMEL Aero

- Compressible Navier-Stokes Equations
- Finite Volume Discretization (cell center)
- Second order in time and space
- Hybrid Mesh (tets, hex, prism, pyramid)
- Matrix-free GMRES iterations (2nd order in time and space)
- Detached Eddy Simulation (Spalart-Allmaras for near wall)
- Validated for several external flow problems





Parallel Clusters

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

- **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

Overset Tool

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 format only for unstructured mesh

Implementation into CaMEL Aero

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 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 the 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.

Implementation into CaMEL Aero

```
!------------------------------------------------
! Initialize dirtlib parallel environmnet
      call drtf_pll_init(0,0)
! get processor mapping
      call mapping(0)
!------------------------------------------------
```

Initialize DirtLib,
Generate Global Processor Mapping

```
!------------------------------------------------------------
! Move mesh block and calculate face velocities
      call movemesh(its,time)
      call meshvelocity()
! volume and area is the same since it is solid body motion
      call meshgeom_noVolArea()
      call motion_xml()
! run Suggar for new coordinates
      call run_suggar(dci_filename)
!------------------------------------------------------------
```

Move mesh and generate SUGGAR motion file then run SUGGAR (SUGGAR runs in advance for Prescribed motion)

```
!--------------------------------------------------------------------
! Load DCI file header
      call drtf_load_dci_file_header(trim(dci_file) // char(0))
! copy the decomposition map of the mesh into the DirtLib's memory
      call drtf_copy_decomposition_map(iepart(1:ne),ne,0)
! Set the number of flow variables for DiRTlib to manage
      call drtf_set_ndata_val_all_grds(nvar)
! Load DCI file data
      call drtf_load_flex_grid_drt_file(trim(dci_file) // char(0))
! initialize dirtlib
      call drtf_init(put_data_value, get_data_value, blank_location,
                     orphan_location, flag_blank_fringe_points)
! get iblank values from database to the solver
      call drtf_fill_iblank_frng_all_grds( iblank, -1)
      call drtf_fill_iblank_out_all_grds(  iblank,  0)
      call drtf_fill_iblank_orph_all_grds( iblank, -2)
!--------------------------------------------------------------------
```

DiRTlib parameters setup,
Read SUGGAR DCI file,
Initialize data exchange parameters
Get IBLANK vector to the solver

Implementation into CaMEL Aero

```
!--------------------------------------------------
! Overset Update
! Generate the donor values and prepare to send
    call drtf_dv_generation_all_grds()
    call drtf_gather_into_send_buffer()
! Send and receive
    call drtf_send_all()
    call drtf_recv_all()
! Collect received values and apply to fringe points
    call drtf_scatter_from_recv_buffer()
    call drtf_dv_apply_rcptval_all_grds()
! Average the values at out and orphan nodes
    call drtf_blank_points_all_grds()
    call drtf_orphan_points_all_grds()
!--------------------------------------------------
```

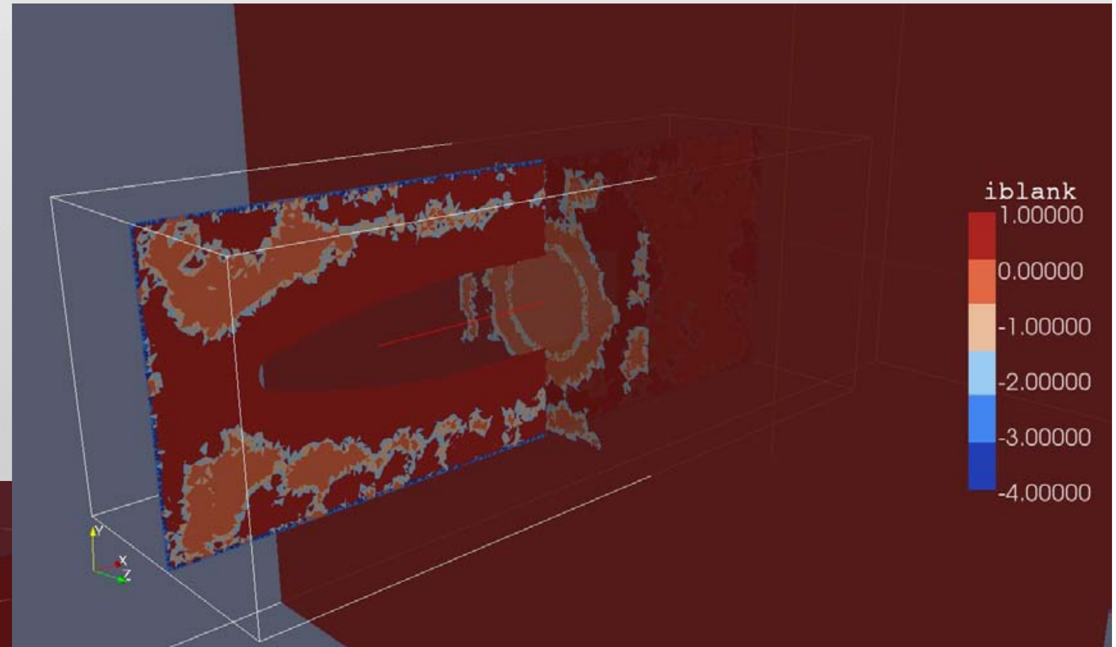
Communicate between blocks,
Send/recv values (fringe/donor interpolation)

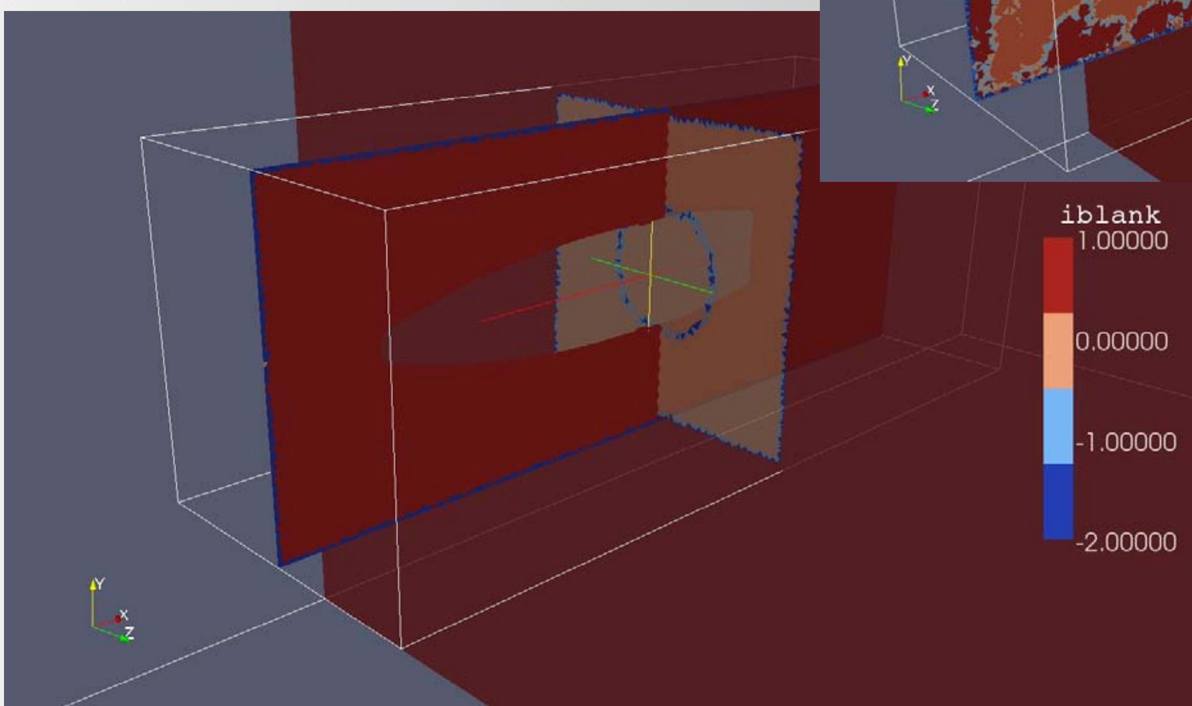
```
!--------------------------------------------------
! Blankout rediduals at hole-cut cells
! Steady Residual
   DO ie=1,nec
      IF (iblank(ie).le.0) RHS(1:ndf,ie) = 0.0D0
   ENDDO
! unsteady Residual
   DO ie=1,nec
      IF (iblank(ie).le.0) FUNC(1:ndf,ie) = 0.0D0
   ENDDO
!--------------------------------------------------
```

Some parts of the solvers need modifications,
Example: At blanked out nodes residuals vanish

SUGGAR fringe elements

<Minimize overlap/>

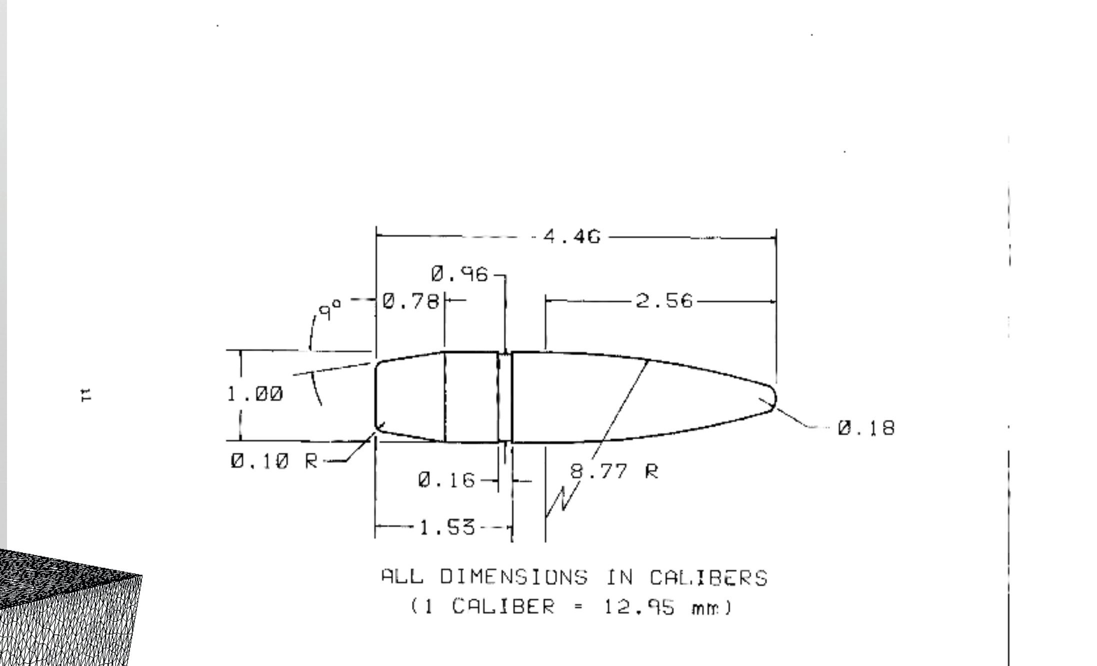


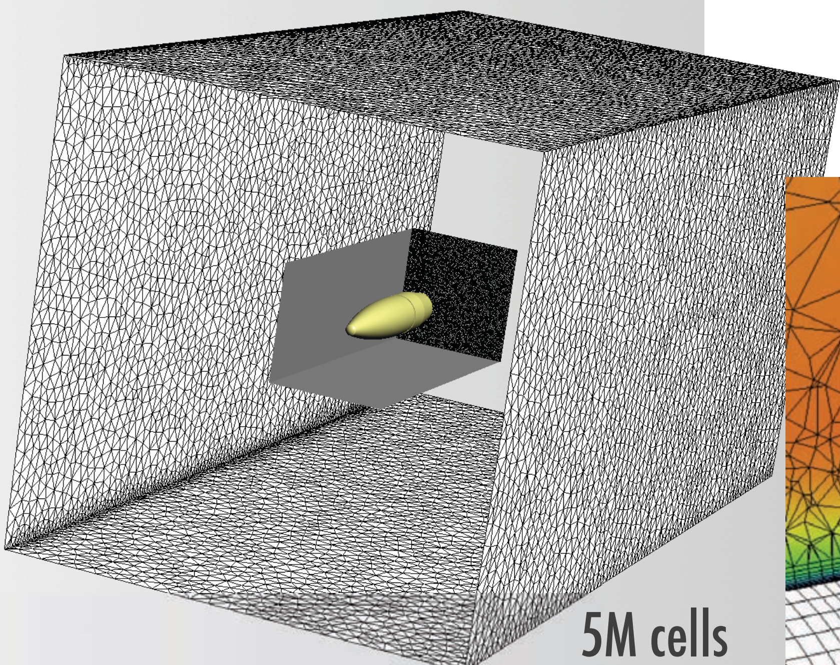


Use <Skip_overlap_opt set_dsf_value="0.0"/> for projectile volume mesh

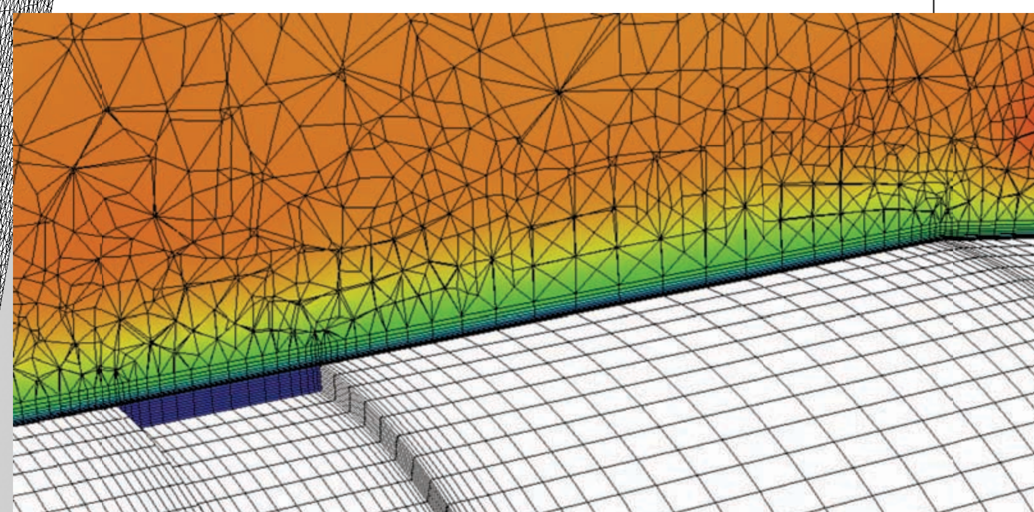
Projectile: geometry and mesh

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.



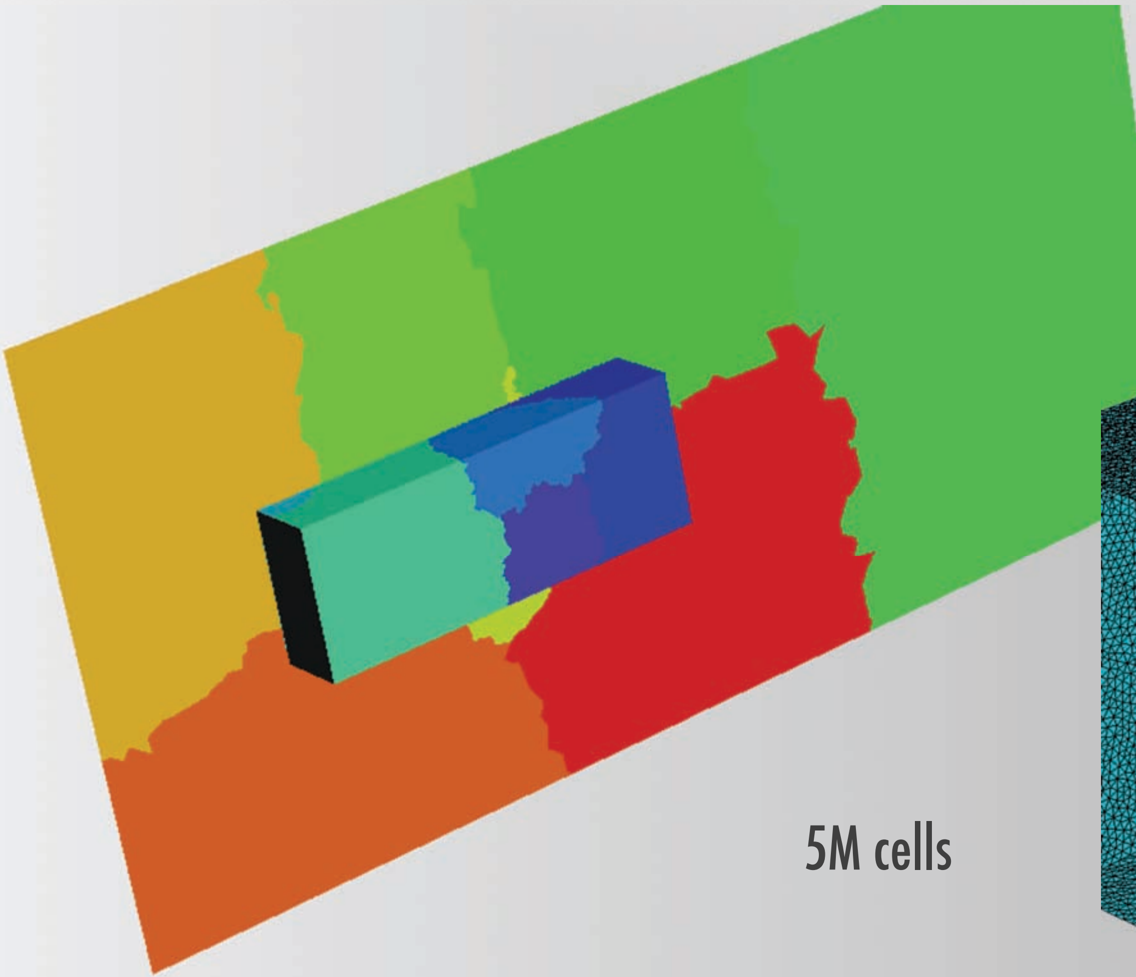


5M cells

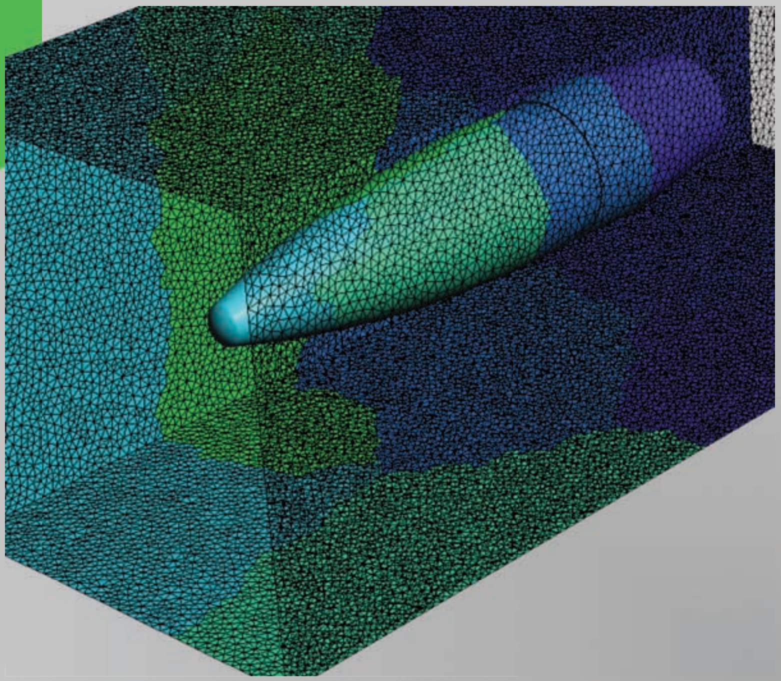


Projectile: mesh partitions

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

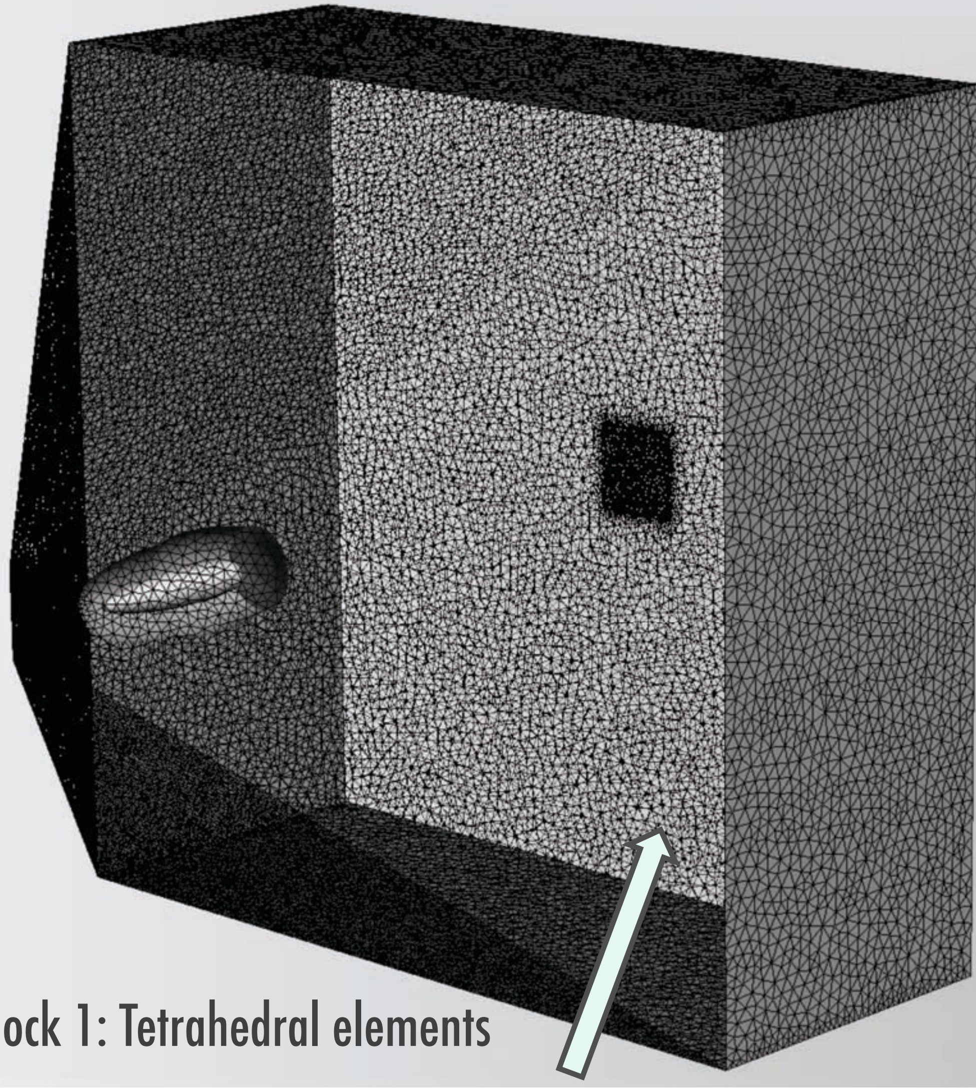


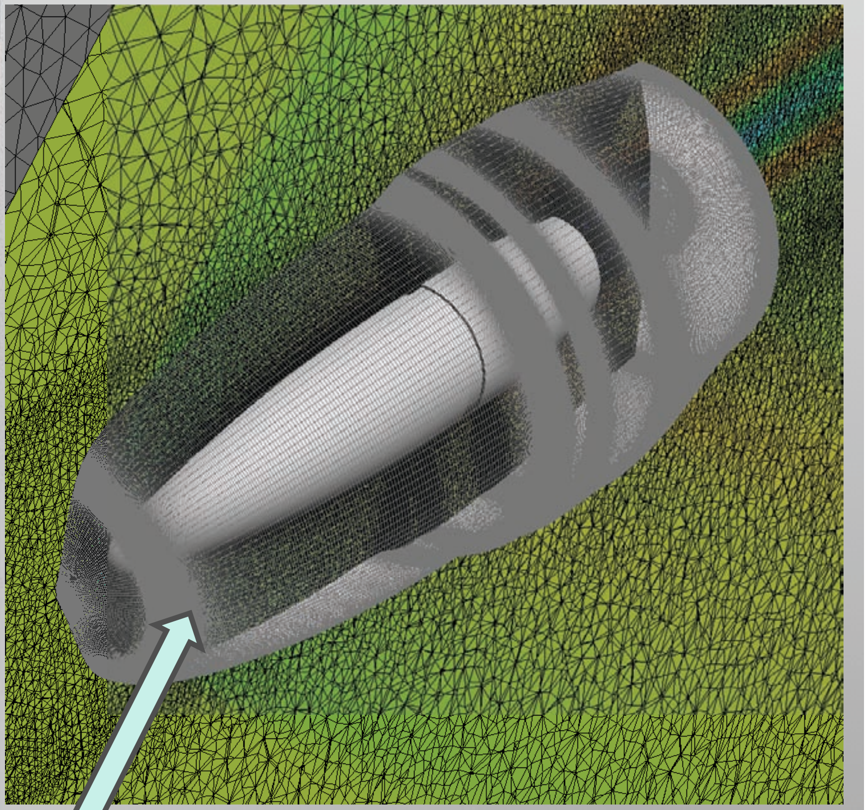
5M cells



Projectile: finer mesh

26 M cells



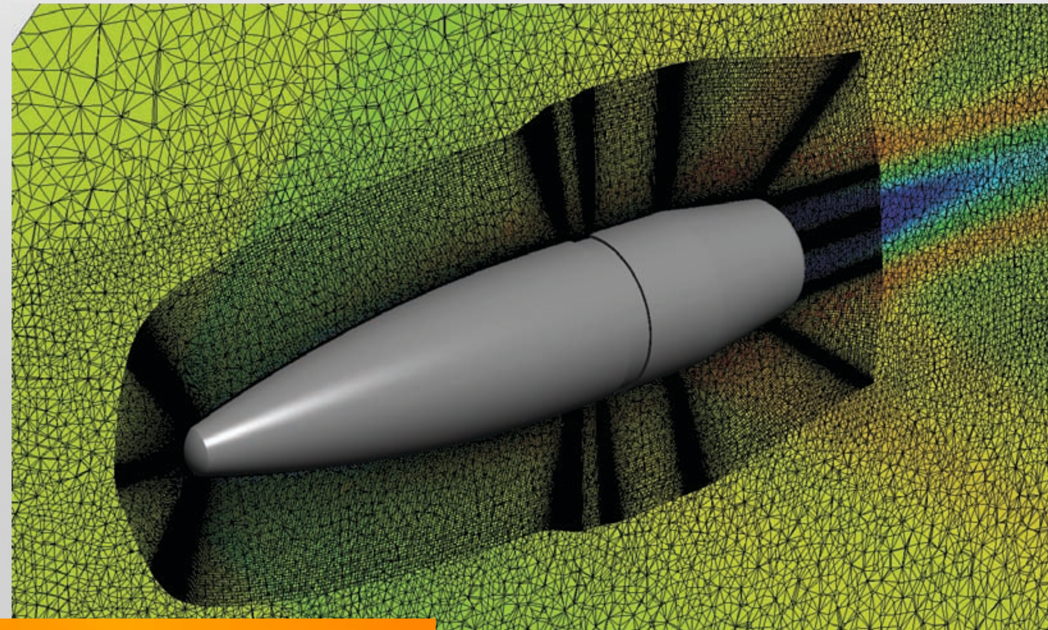


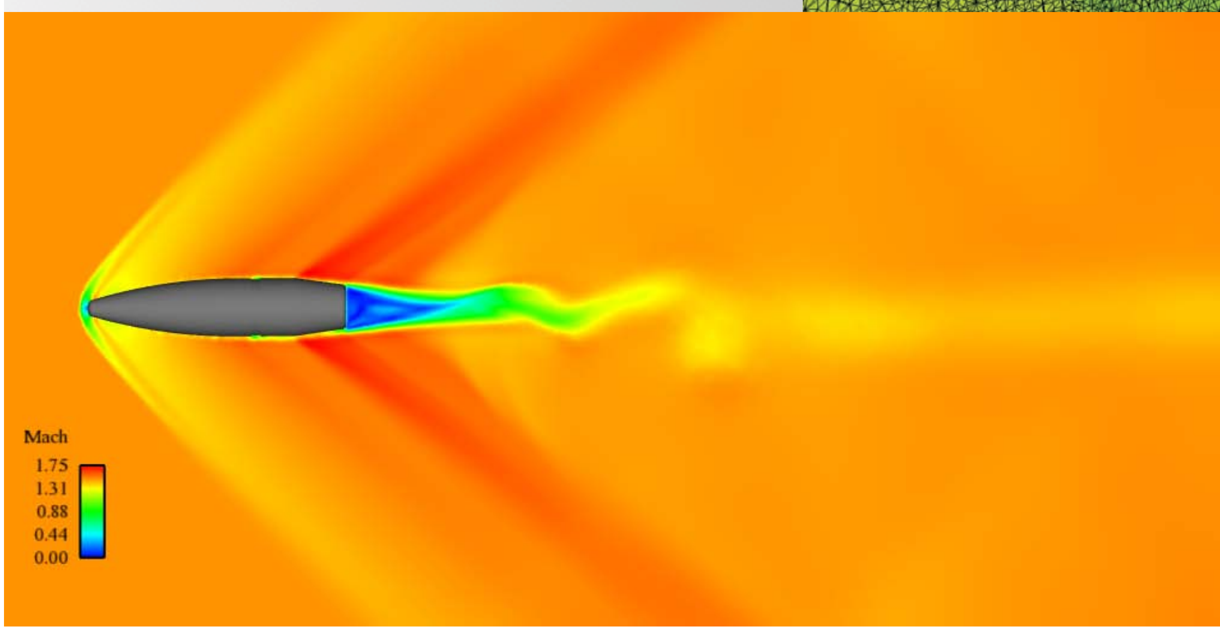
Block 1: Tetrahedral elements

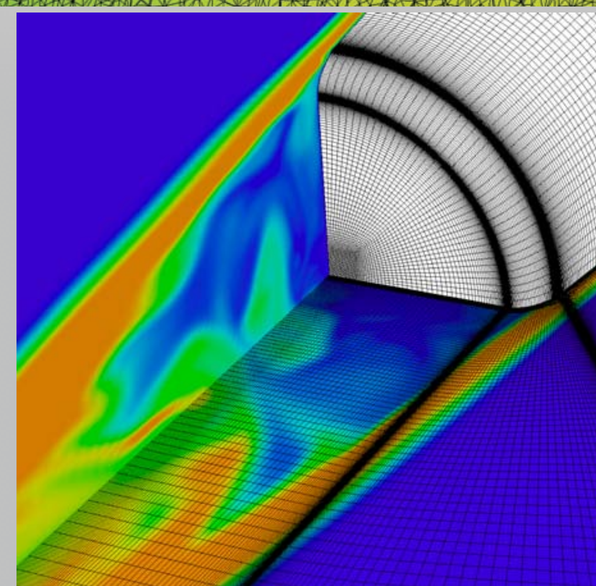
Block 2: Hexagonal elements

Projectile: finer mesh solution

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



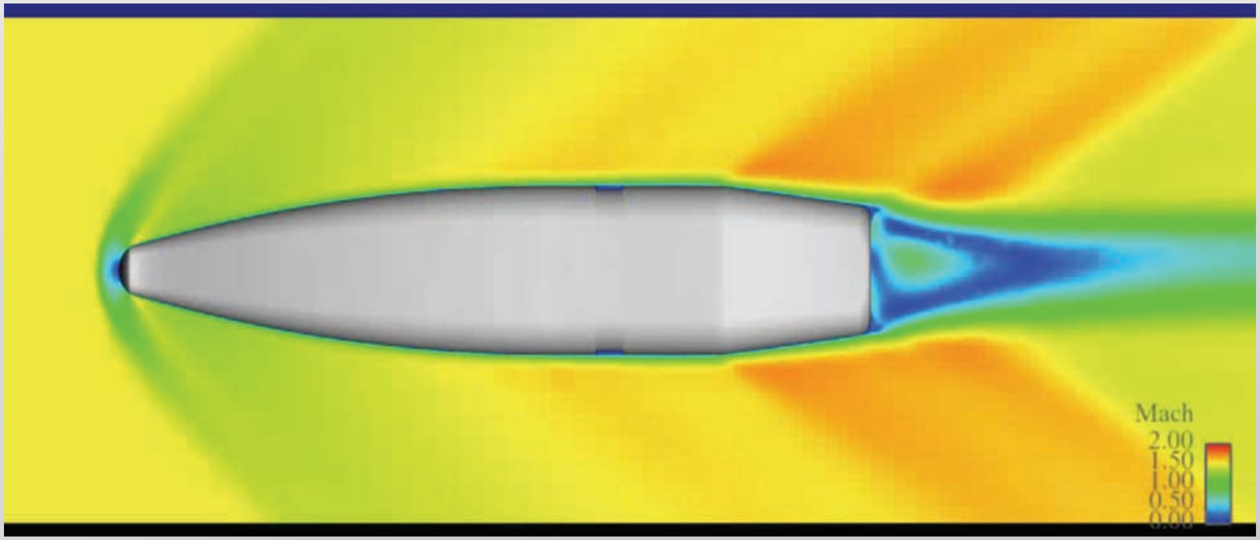




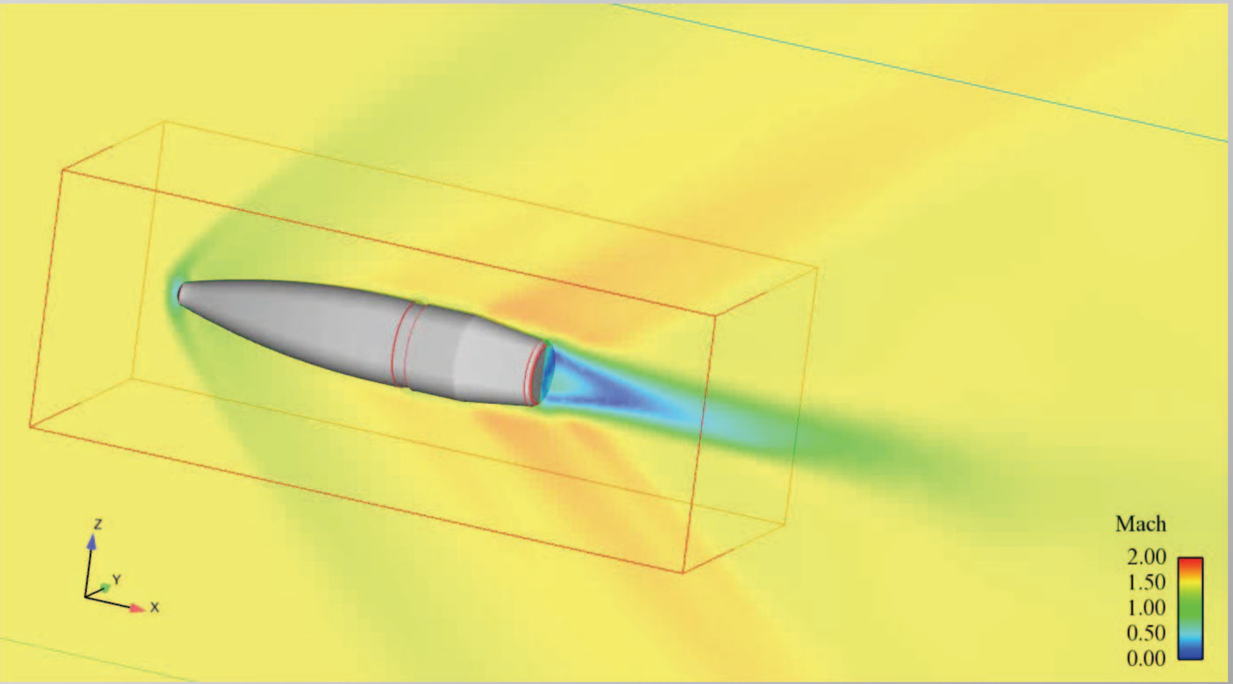
Projectile: pitching oscillation

5 M Cells
Second order solution
5 degrees of pitching oscillations

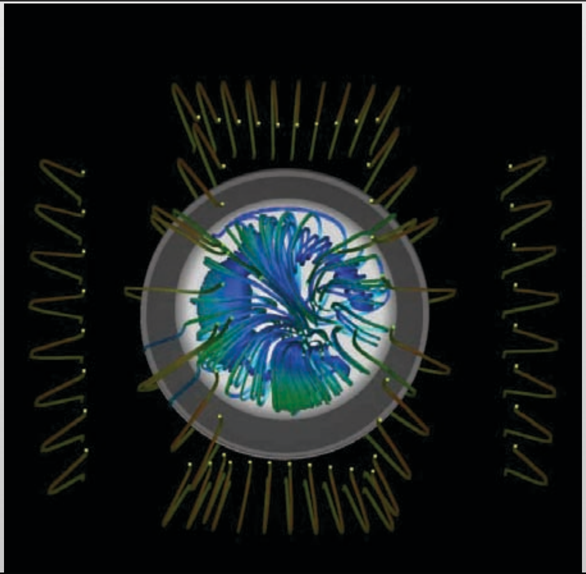
Projectile mesh

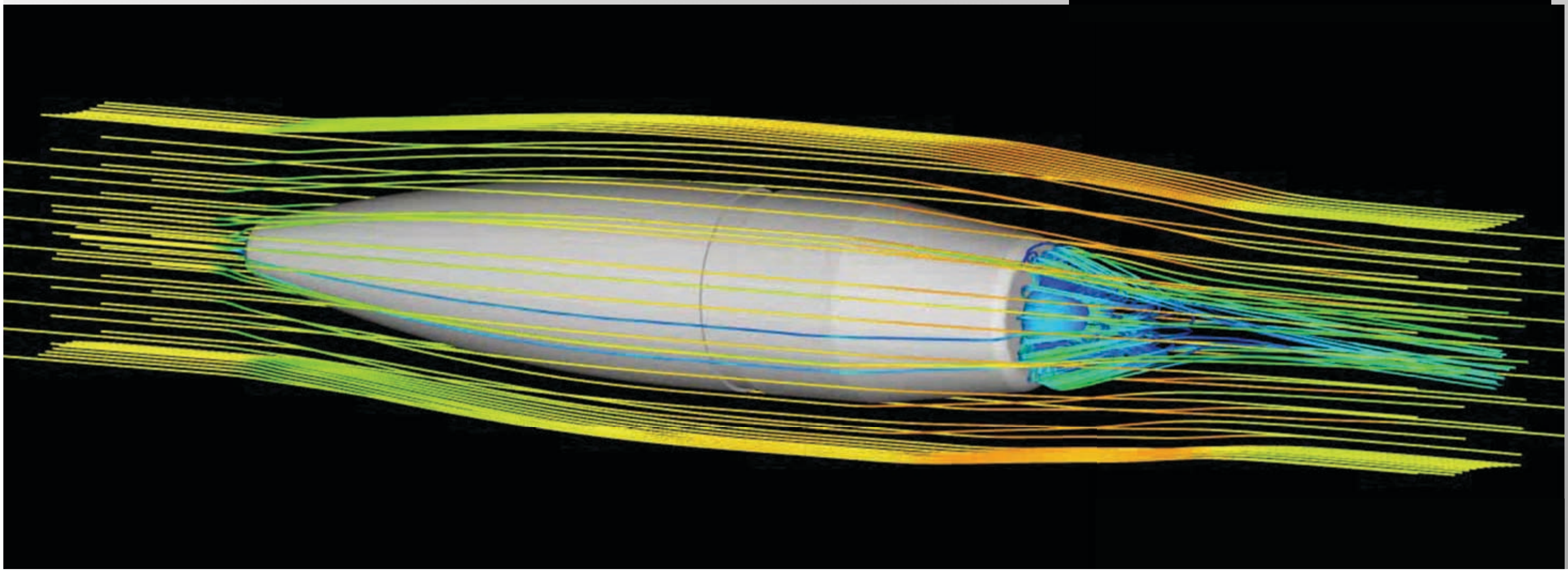


Background mesh



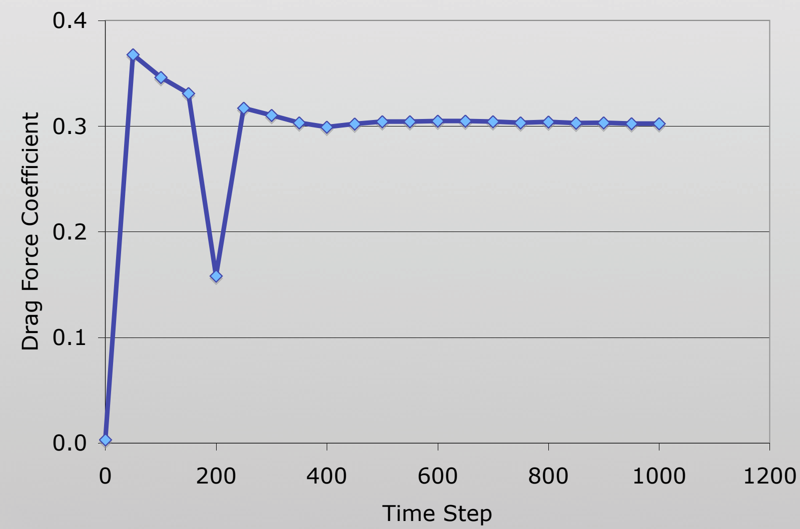
Projectile: pitching oscillation

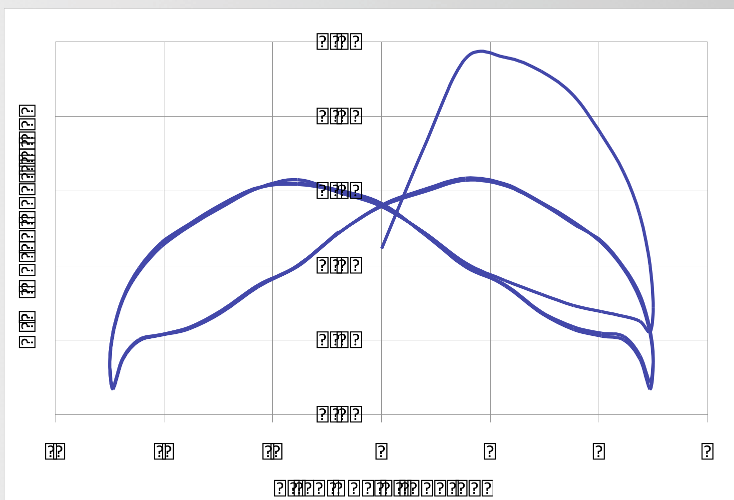


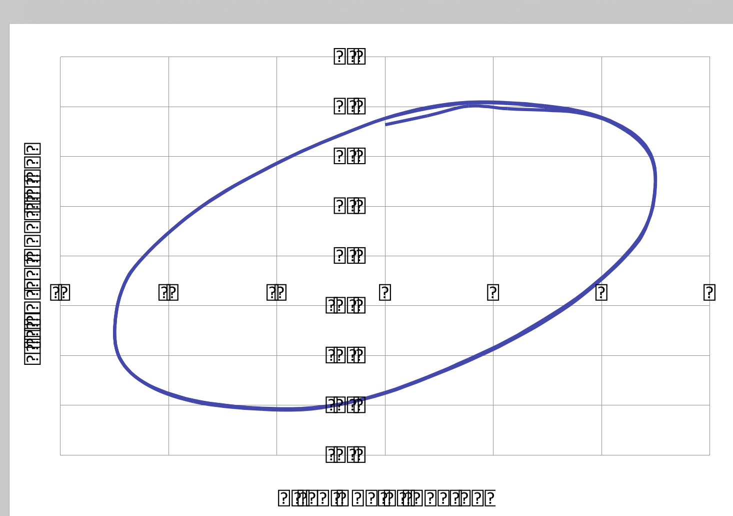


Force coefficients

Initial solution at zero AoA

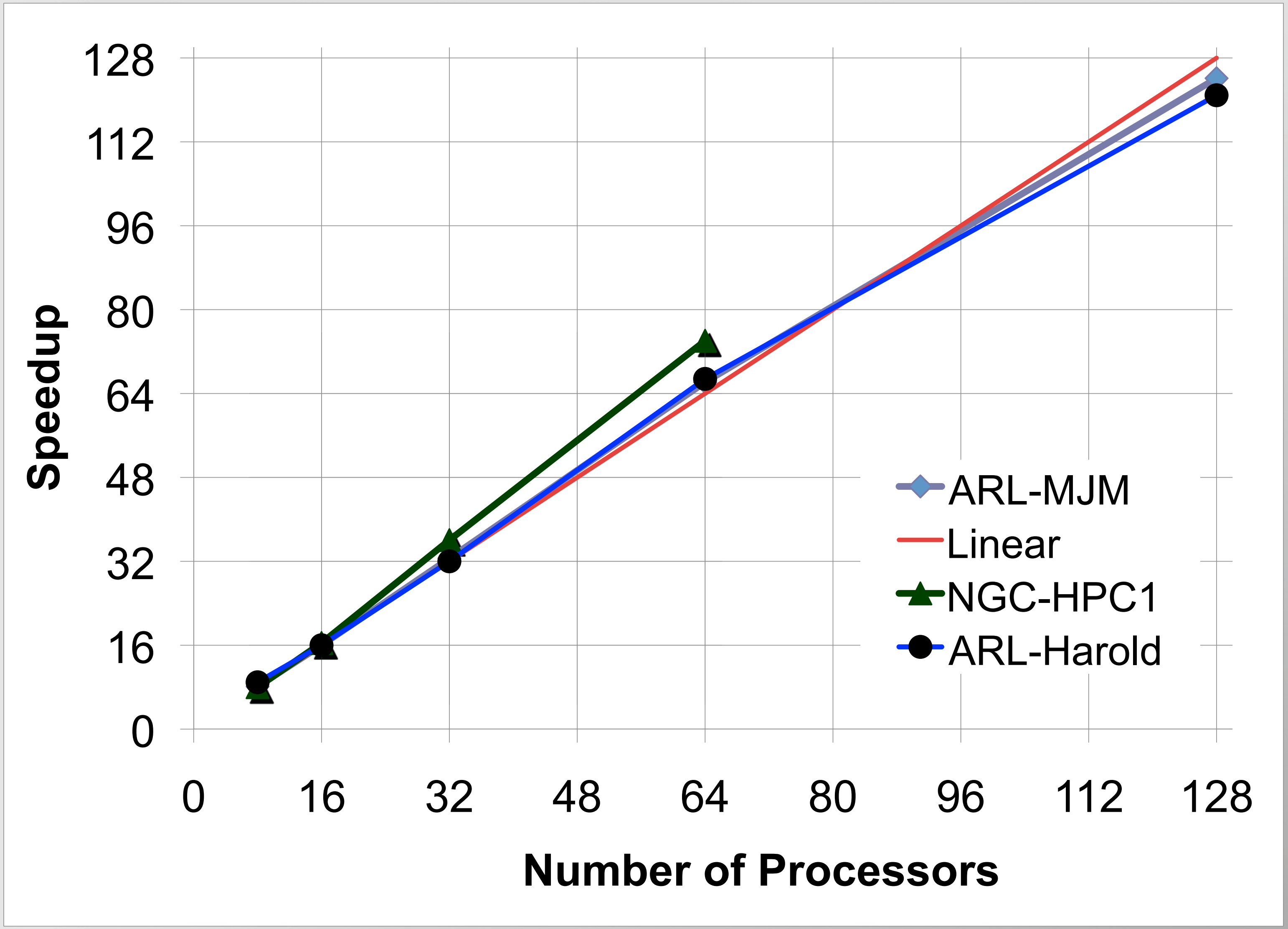




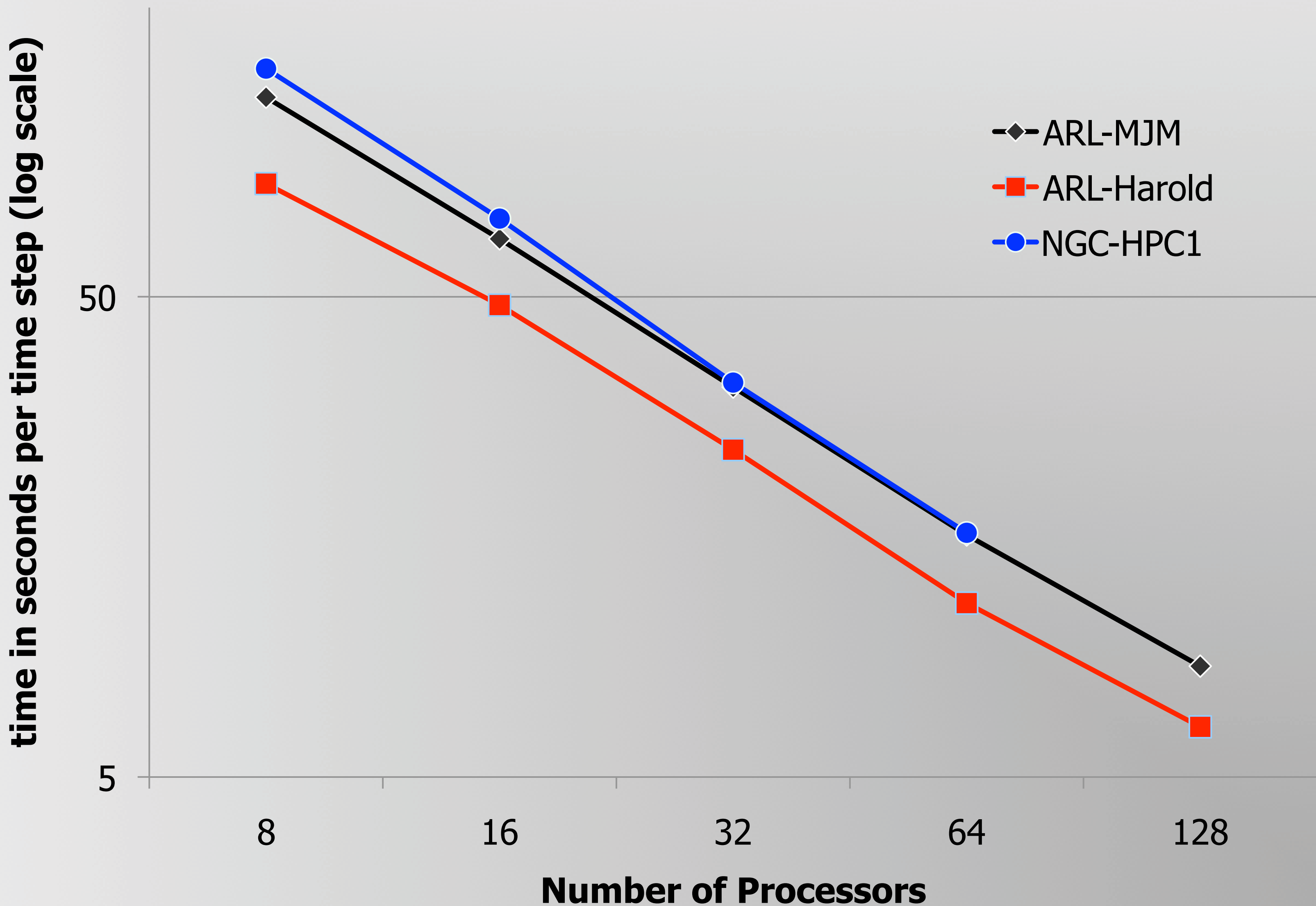


5 M elements mesh for five degrees of pitching oscillations

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 off 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 1014 sec

Conclusion

- Overset capability using SUGGAR/DiRTlib was implemented into CaMEL Aero flow solver.
- DiRTlib runs smooth for parallel communication and update between the overset mesh blocks
- Since SUGGAR runs sequential and all other processors wait for it every time step, SUGGAR is the bottleneck in parallel computing for the moving mesh problems (not valid for prescribed motion).
- Libsuggar implementation requires special attention if solver has mixed C and Fortran programming.

Future Work

- Implement SUGGAR/DiRTlib into incompressible CaMEL version.
- Run bigger mesh and for more complex motions.
- Move into the parallel version of SUGGAR, Suggar++, for better parallel efficiency.