FUN3D v14.0 Training
Grid Adaptation

Gabriel Nastac
gabriel.c.nastac@nasa.gov
Computational AeroSciences Branch
NASA Langley Research Center

Public Community Questions: fun3d-users@lists.nasa.gov
Private/Proprietary Questions: fun3d-support@lists.nasa.gov

Spring, 2023

Overview

- Prerequisites
- Grid Adaptation
  - With CAD
  - Without CAD (hybrid frozen surface and BL approach)
- Solution Strategies
- Example Inputs
- pyrefine
- Goal-oriented Adaptation
- Examples
  - Supersonic Diamond Airfoil
  - Subsonic NACA 0012 Airfoil
  - Transonic ONERA M6 Wing
  - Hypersonic Cylinder
  - Hypersonic CEV Capsule
- Where Things Can Go Wrong
- References

Prerequisites

- Users should be familiar with the flow solver and problems of interest using fixed grids
- Users should go over the grid adaptation section in the manual
  - https://fun3d.larc.nasa.gov/papers/FUN3D_INTG_Manual-14.0.pdf#section.8
- NASA refine, our refinement package, is built by default with the newest release
- For adaptation with CAD, you must link Engineering Sketch Pad (ESP) and the OpenCASCADE CAD kernel
  - https://fun3d.larc.nasa.gov/papers/FUN3D_INTG_Manual-14.0.pdf#subsubsection.A.10.16
  - Hybrid frozen surface and boundary layer approach does not need to link these
- You will also need TetGen (open source: http://tetgen.org/) or AFLR3 (https://www.simcenter.msstate.edu/index.php) for bootstrap for CAD adaptation
- The ESP website (https://acdl.mit.edu/ESP/) has pre-built binaries for both ESP and OpenCASCADE, although you can build from source if desired (ESP is open source)
  - ESP does have a GUI; will not be covered
  - ESP training exists on the website: https://acdl.mit.edu/ESP/Training/
- **Clean CAD is assumed; CAD cleanup will not be covered**
  - **Most commercial CAD packages do not use the OpenCASCADE CAD kernel used in ESP**
  - **Expect issues and CAD cleanup to successfully import complex geometries not made with ESP**

Grid Adaptation Overview

- **CAD-to-Solution (C2S) (also called Sketch-to-Solution (S2S)) adaptation with geometry**:
  - User provides **clean CAD** and solver settings
  - Process runs CFD through automated grid adaptation with refinement toward grid convergence
  - **Restricted to pure tetrahedra at this time, heating/shear prediction noisy, but about the right magnitude once grid spacing is adequate**
  - **Not possible to explicitly enforce surface spacing; will need more points than hybrid approach to reach target wall-normal spacing**
  - Manual has a simple ONERA OM6 wing example
- **Hybrid adaptation without geometry**:
  - Fixed surface and thin boundary-layer grid (non-tetrahedral elements) utilizing similar adaptation process – **refine only adapts tetrahedra**
  - **Enforces desired surface and wall-normal spacing**
  - **Yields improved heating/shear prediction versus pure tetrahedra**
- Regarding refinement packages
  - NASA refine is technically refine 3
  - **refine 1 and 2 are internal to FUN3D; use similar methodology but are being deprecated in favor of 3**

Challenges with Grid Adaptation

- **Grid adaptation poses additional challenges, some things to consider:**
  - Aerodynamic quantities typically converge on fixed grids without needing to finely resolve shocks, though problem-dependent
  - **If you can make an accurate fixed grid for your problem trivially, do so**
  - Accurate surface gradient prediction (heating and skin friction) will typically require finely resolving shocks and boundary layers
  - Grid adaptation can make your problem more expensive to run
  - refine is not currently GPU-enabled. If you run flow solver on GPUs, refine will be the vast majority of run time on GPU nodes. Refinement should be done on CPU nodes for efficiency if needed
  - **Grid adaptation is not fully deterministic due to tolerances; final solutions will still be the same, but grids will not be identical generally**
  - You may not necessarily be able to start a simulation from scratch on a highly refined grid
  - Highly anisotropic elements stress solver numerics
  - **Adaptation is mainly applied to steady-state problems**
    - **Unsteady adaptation is still an active research area and not covered today**

Grid Adaptation Process

- Overall process:
  - Create initial grid (with or without CAD)
    - With CAD: bootstrap (generates minimal grid), and optionally adapt (generates grid targeting point count)
    - Without CAD: Mesher of choice, use non-tetrahedral elements in BL. Ensure adequate surface spacing.
  - Run flow solver
    - Output solution and optional scalar interpolant field (currently only one supported)
      - {project}_volume.solb contains primitive variables (e.g., rho,u,v,w,p,turb1)
      - Mach is used implicitly by default for perfect gas path
      - Can choose any variable; Mach and temperature are most common
      - Goal-oriented is possible; covered briefly later
  - Adapt
    - Compute multiscale (or goal-oriented) metric from flow solver
    - Generate new grid using adaptation mechanics
    - Interpolate previous flow solution onto new grid
  - Repeat until you are done

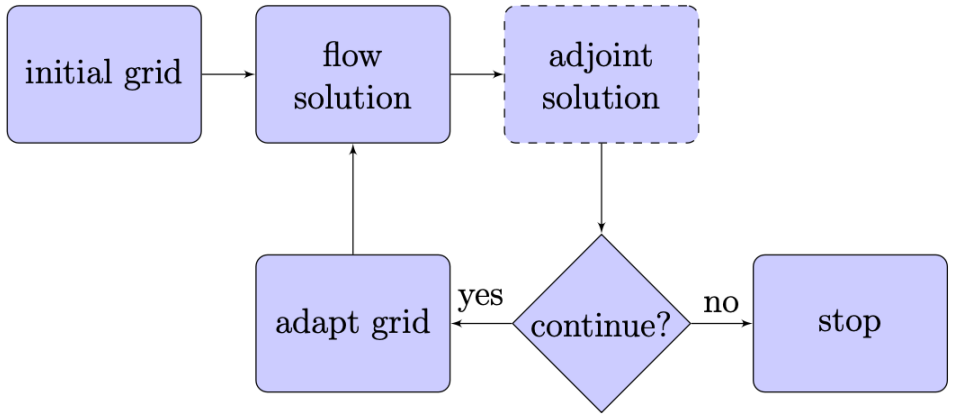
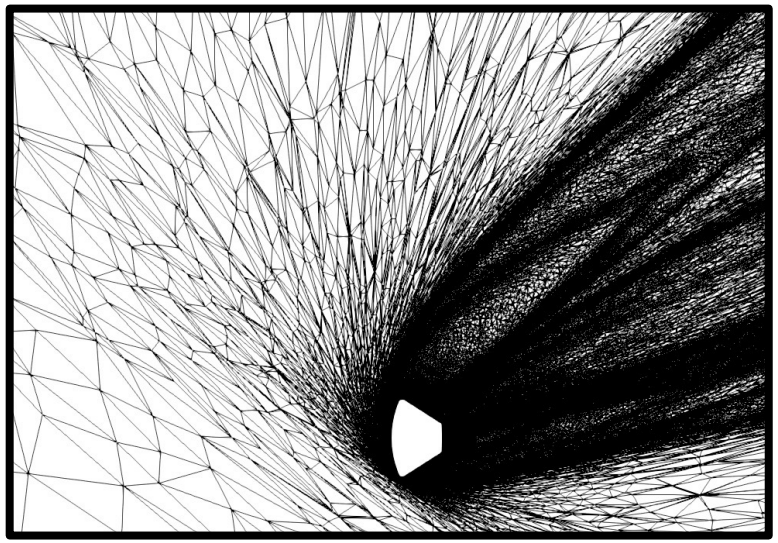


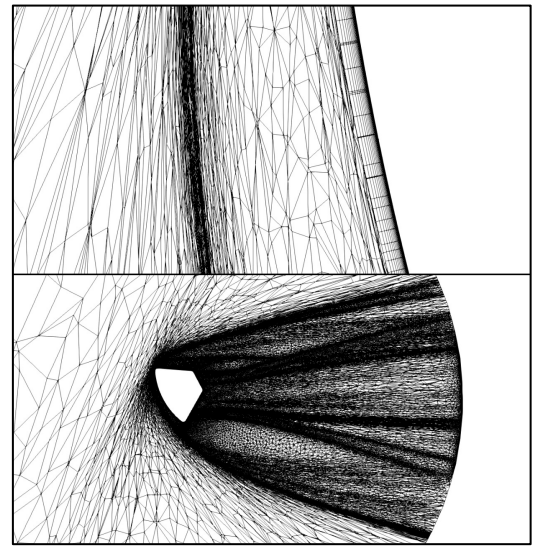
Figure 5: Solution-based grid adaptation process.

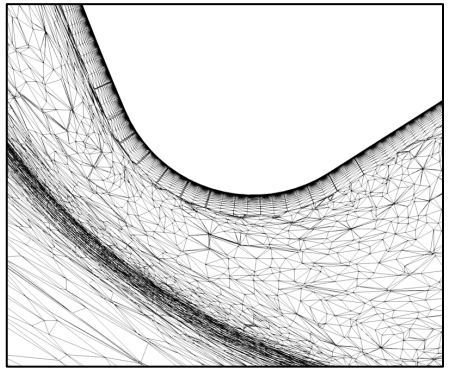


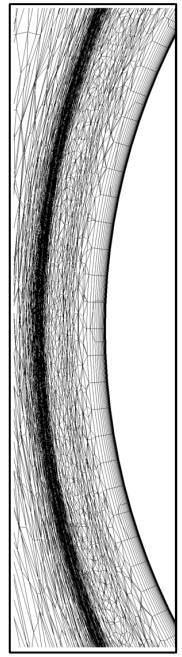
Hypersonic 3D CEV example covered later

Grid Adaptation Process (cont.)

- Primary input into refinement is number of points through complexity
  - ~2*complexity for pure tetrahedra
  - ~(2-3)*complexity for mixed element grids – **refine will only adapt tetrahedra**
    - Due to fixed surface/BL grids, as points → ∞ and exceed BL points, this ratio will tend to 2
    - Latest refine now includes mixed elements in complexity formula; did not consider this before (only tetrahedra were counted)
    - Points in prism BL ~ prisms/2
- refine uses solb/meshb file formats
- Refinement cycle includes a CFD solution and grid update
- No specific process/iteration strategy set in stone
  - Default approach for f3d wrapper, our grid adaptation Ruby driver, is doubling complexity (`schedule_complexity_growth`) after a few cycles at fixed complexity
  - Hybrid grid strategy is generally to fix the complexity as we will always have our target surface spacing
- **Recommended to simulate only the domain of interest to reduce cost**
  - **Don't simulate the wake if it isn't needed**
- **Hybrid approach**
  - **Ensure BL height is below shock standoff distance**
  - **Aspect ratio cells O(1-10) are ideal at interface**
  - **Quality of surface and BL grid matters for hybrid approach. Ideally, thickness and number of layers should be as uniform as possible. Any gaps in BL grid can cause issues with adaptation.**







Solution Strategies – FUN3D Finite-Volume (FV)

- FUN3D SFE covered next week
- Overall, use similar options as you would for a fixed grid
  - May need reduced CFL over high quality fixed grids
  - e.g., for supersonic flow, use "`dldfss`" inviscid flux construction and CFL ~ O(10)
- Use fewer iterations than you would for a fixed grid for an adaptation cycle
  - Problem dependent, typically a few hundred to potentially a few thousand iterations
  - Ideally solution is as steady as possible before adaptation
- For multiscale adaptation, Mach or temperature are recommended scalars
  - Temperature as the metric is recommended for hypersonic flows for smoother heat transfer (unable to capture adiabatic wall boundary layers for C2S)
  - Mach will better capture boundary layers but captures shocks worse for chemically reacting flows
- For perfect gas path, HANIM can aid in robustness
  - Not required
  - For some problems such as those involving limiters, computational performance may improve reducing max CFL (`hanim_max_cfl`) to a certain value (e.g., 10^2-10^4; default is 10^{11})
- Monitor surface spacing
  - Once again, can't currently enforce surface spacing for C2S
  - Boundary output: "`yplus`", "`slen`", "`re_cell`"
- SST models require y^+ ~ 1 and are typically unstable for large y^+
  - Ensure y^+ is adequate
  - Recommended to use SST only for hybrid grids with adequate wall resolution
  - SA-based models are recommended overall
- First order inviscid fluxes (`first_order_iterations`) are generally recommended for first few adaptation cycles
  - Especially for supersonic/hypersonic cases
- For high hypersonic cases, you may want/need to restart from scratch for a few cycles
  - Set `import_from` to blank ("") in case_specifics for iteration < X
- Edge-based viscous terms (EBV) are recommended though still experimental
  - See https://fun3d.larc.nasa.gov/session6_2023.pdf#page=8 for more details, some caveats
  - Speeds up perfect gas path by ~1.4x
  - Speeds up generic gas path by ~2x

Solution Strategies – refine

- "`-h`" or "`--help`" on refine commands will output available options, e.g.:
  - "`ref –h`"
  - "`ref loop –h`"
- Sweeps (-s <var>)
  - Default is 30, 10 is recommended for hybrid approach, can go lower to improve speed but grid quality will degrade slightly
- Buffer (--buffer)
  - Coarsens the metric as it approaches the outflow plane
  - Multiscale adaptation by default will adapt shocks and wakes all the way to outflow which will increase cost and can impact stability
- Fixed Point (--fixed-point)
  - Used for time-accurate refinement; interpolant scalar must be sampled across many time steps instead of just the final time step
  - See manual for more information
  - refine is too slow to use every few CFD steps; fixed point should use enough samples to capture lowest frequency of interest
- Aspect Ratio (--aspect-ratio <var>)
  - Limits aspect ratio; advisable for unsteady problems. This will limit the capability to resolve boundary layers for C2S; **not recommended generally**
- Gradation (--gradation <var>) – Stretching Ratio
  - Smooths metric field; setting to a value such as 10 can help with shock capturing (reduces anisotropy at shocks)
  - Can help flow solver stability as shocks shift during adaptation
  - Can reduce anisotropy in BL leading to larger wall spacing (requiring more points to reach target spacing)
  - **Default is recommended**
- Spalding (--spalding <var>)
  - Initializes (**does not enforce**) wall spacing for refinement. Spacing should be set to desired target spacing (e.g., $y^+ = 1$)
  - Recommended for viscous problems

Solution Strategies – refine (cont.)

- Axisymmetric Grids
  - Simply append "`--axi`" for most refine commands (bootstrap, adapt, translate, loop (ref_cl)) following 2D approach
  - **Quarter symmetry recommended as axisymmetric is a pseudo axisymmetric approach currently, YMMV**
  - Hybrid approach is in theory possible for 2D/axisymmetric grids but much more complicated as FUN3D grids are 3D; not supported
- Interpolation Error control (--norm-power <var>)
  - Default is 2
  - **4 is recommended to better capture boundary layers for C2S – see Ref. 3 for details**
- Interpolation metric (--interpolant <var>)
  - Default is "mach" for perfect gas path, "`ref loop -h`" shows other built-in options
  - Can use interpolant solb file containing any scalar output from FUN3D, required for generic gas path
- Without geometry, refine assumes planar faces
  - Recommended to use a box instead of a sphere for farfield for hybrid approach
  - A sphere farfield would not gain curvature with improved refinement since we don't have geometry
- Refine performance can be comparable to flow solver and is also not on GPUs currently
  - Will generally be vast majority of simulation cost on a GPU node with many GPUs for large grids
    - May not be a problem if you have few GPUs per node
  - Maximum points thus far O(100-200M)
  - Can adjust sweeps to reduce runtime
  - Manual approach enables you to split jobs across different node types if needed (e.g., CFD on GPU nodes, refine on CPU nodes)
    - Job schedulers have dependency support (e.g., wait until one job is done to run next)
    - Python package for PBS: https://github.com/nasa/pbs4py
      - Example: https://github.com/nasa/pyrefine/tree/main/examples/onera_m6/steady_sa_gpu
  - Refine has performance issues when using hybrid approach with grids consisting of many hexahedra/pyramids
    - Use prism/tetrahedra grids for hybrid approach (which enables EBV to be used)

Grid Adaptation - Approaches

- Three approaches will be covered on the same setup
  - Hybrid approach using explicit commands to understand process
  - Hybrid approach using f3d Ruby script
  - C2S approach using f3d Ruby script
- These are the CEV examples from the Thermochemical Nonequilibrium training:
  - https://fun3d.larc.nasa.gov/training-9.html#agenda
  - https://fun3d.larc.nasa.gov/session6_2023.pdf
- Overall process also applies to perfect gas path
- Perfect gas path examples will be covered later
- Multiscale adaptation will be the focus here (not goal-oriented adaptation)

Grid Adaptation – Hybrid Manual Approach

Files in directory:

```
fun3d.nml.1
fun3d.nml.2
cev01.lb8.ugrid
cev.mapbc
tdata
```

```
# adapted points ~(2-3)*complexity for mixed element grids
target_complexity=1500000
initial_complexity=1000000
ramp_complexity=100000


# clean
rm -rf Flow
mkdir Flow


# initialize
cp cev01.lb8.ugrid cev.lb8.ugrid
ref translate cev.lb8.ugrid cev.meshb
mv cev.lb8.ugrid cev.meshb Flow/
cp fun3d.* tdata cev.mapbc Flow/
cd Flow/
touch output.txt


# initial solution
iter=1
cp fun3d.nml.1 fun3d.nml
time mpiexec refmpi distance cev.lb8.ugrid cev-distance.solb --
fun3d-mapbc cev.mapbc
time mpiexec nodet_mpi > screen.out
cp cev_tec_boundary.szplt "cev${iter}_tec_boundary.szplt"
cp cev_slice.szplt "cev${iter}_slice.szplt"
cp cev_hist.dat "cev${iter}_hist.dat"
cp screen.out "screen${iter}.out"
cp cev.grid_info "cev${iter}.grid_info"
date >> output.txt
echo $iter >> output.txt
```

```
# iterate
cp fun3d.nml.2 fun3d.nml
complexity=$initial_complexity
for iter in {2..15}
do
# fun3d outputs interpolant and volume solb files
  time mpiexec refmpi loop cev cev1 $complexity -s 10 --interpolant
cev_sampling_geom1.solb > ref.out
# refine outputs cev1.lb8.ugrid cev1-restart.solb cev1.meshb
  mv cev1.lb8.ugrid cev.lb8.ugrid
  mv cev1-restart.solb cev-restart.solb
  mv cev1.meshb cev.meshb
  time mpiexec refmpi distance cev.lb8.ugrid cev-distance.solb --
fun3d-mapbc cev.mapbc
  time mpiexec nodet_mpi > screen.out
  cp cev_tec_boundary.szplt "cev${iter}_tec_boundary.szplt"
  cp cev_slice.szplt "cev${iter}_slice.szplt"
  cp cev_hist.dat "cev${iter}_hist.dat"
  cp screen.out "screen${iter}.out"
  cp cev.grid_info "cev${iter}.grid_info"
  cp ref.out "ref${iter}.out"
  date >> output.txt
  echo $iter >> output.txt
  echo $complexity >> output.txt
  if [ $complexity -lt $target_complexity ]
  then
    let "complexity+=ramp_complexity"
  fi
done
```

Grid Adaptation – Hybrid f3d Approach

- `f3d` Ruby script simplifies process but hides details
- `f3d` should already be in your path if the bin folder is
- `f3d --help`
- `f3d watch` will tail flow and refinement output
- `f3d iterate` output will show commands being used
- Here, we are fixing complexity for 15 cycles
- Relative tolerance (which checks drag coefficient) must be low to prevent f3d from stopping
- *_cl enables command line options to be set
- **Refine distance function must be used for pure tetrahedral grids; default FUN3D distance function at the moment cannot handle heavily skewed anisotropic elements near surface. Mixed-element grids are generally ok with FUN3D function**
- You can set various namelist options as a function of iteration or mesh complexity
- **Recommended to use first order iterations for initial cycles to initialize flow field**

```
Files in directory:

fun3d.nml
cev01.lb8.ugrid
cev01.mapbc
tdata
case_specifics
```

```
case_specifics file:

root_project 'cev'
schedule_initial_complexity 1_500_000
schedule_max_complexity 1_500_000
schedule_complexity_per_core 500
schedule_subiteration_limit 15
schedule_relative_tol 1E-8
ref_cl("-s 10 --interpolant #{project}_sampling_geom1.solb")
def iteration_steps
  refdist
  if (iteration < 3) then
    flo({'first_order_iterations' => '2000',
         'steps' => 2000})
  else
    flo
  end
  ref_loop_or_exit
end
```

```
Run script:
f3d clean
ref translate cev01.lb8.ugrid cev01.meshb
time f3d iterate > output.txt
```

Grid Adaptation – C2S f3d Approach

- ESP supports IGES and STEP files
- You do not have to do the Boolean of a farfield with ESP
  - **Once again, ESP utilizes OpenCASCADE CAD kernel which is not used in majority of CAD packages and thus tolerance issues are common**
- ESP generates EGADS file which contains geometry representation
- meshb files now include geometry to adapt the surface
- 4-norm, which better captures boundary layers, is recommended
- Spalding option initializes near wall surface grid with that spacing
  - Not enforced and ultimately depends on metric
- Here, cycling is used, ramping from 200k points to 6.4m points, keeping complexity fixed for 5 cycles unless default drag tolerance is hit

```
Files in directory:

fun3d.nml
tdata
cev.iges
cev.csm
case_specifics
```

```
Run script:

f3d clean
serveCSM -batch cev.csm # newer ESP version uses serveESP
ref bootstrap cev.egads
cp cev-vol.mapbc cev01.mapbc
mpiexec refmpi adapt cev-vol.meshb --spalding 2E-6 100000 \
            --fun3d-mapbc cev-vol.mapbc -x cev01.meshb \
            -x cev01.lb8.ugrid > initialize.txt
time f3d iterate > output.txt
```

```
root_project 'cev'
schedule_initial_complexity 100_000
schedule_max_complexity 3_200_000
schedule_complexity_per_core 500
schedule_subiteration_limit 5
ref_cl("--norm-power 4 --interpolant #{project}_sampling_geom1.solb")
def iteration_steps
  refdist
  mesh_complexity = schedule_current()
  if (mesh_complexity < 400_000) then
    flo({'first_order_iterations' => '1000'})
  else
    flo
  end
  ref_loop_or_exit
end
```

```
cev.csm

box -10 -15 -15 20 30 30
select face
attribute bc_name $5000_farfield
select face 2
attribute bc_name $5051_outflow
import cev.iges # cev iges file
select face
attribute bc_name $4000_cev
subtract # cev from box
scale 0.073025 # scale
dump cev.egads
```

Adaptation Scripting with pyrefine

- pyrefine is a Python module for driving refine-based grid adaptations
  - Available online: https://github.com/nasa/pyrefine
  - pyrefine documentation: https://nasa.github.io/pyrefine
  - **Note: pyrefine does not require FUN3D to be compiled with `--enable-python`**
- The adaptation driver breaks the process into 3 parts using composition.
  - Several options are provided for each part of the process, or you can write a custom one.
  - Some provided options:
    1. refine: multiscale, multiscale fixed point (unsteady), goal-oriented
    2. Complexity controller: fixed schedule growth, C_D convergence-based growth, angle of attack sweep
    3. Simulation: Finite Volume, two-stage Finite Volume for unsteady, SFE, SFE adjoint
  - To launch PBS jobs for each part of the adaptation, the scripts use **pbs4py**, another NASA Python module
    - https://github.com/nasa/pbs4py
    - Each part can have a separate pbs launcher
      - Useful for requesting GPU nodes for FUN3D GPU runs and CPU nodes for refinement
- **There is an examples directory in pyrefine that demonstrates many of the capabilities**
- pyrefine also includes post-processing scripts and interactive tools for monitoring adaptations
  - These tools do not require the adaptation to be run with pyrefine
  - https://github.com/nasa/pyrefine/tree/main/pyrefine/monitoring

Goal-oriented Adaptation

- Goal-oriented adaptation aims to improve the prediction of some output functional such as lift or drag, see **Ref. 1**
  - By targeting a specific output of interest, the adaptation process can converge faster than the multiscale-based process,
    - For example, computing drag on a supersonic vehicle does not require resolving shocks far away from the body, but the multiscale metric will target all the shocks indiscriminately. The goal-oriented metric will target which parts of the domain affect the output functional's accuracy.
- The goal-oriented metric requires an adjoint solution from either the Finite Volume or SFE solvers
  - There are two versions of goal-oriented metrics in the latest refine
    1. `ref loop --opt-goal`: constructs the metric based on the Hessian of the inviscid fluxes and adjoint gradients
    2. `ref loop --cons-visc {mach} {reynolds} {temperature}`: includes the viscous flux terms
  - Both versions of goal-oriented metrics are still research areas for FUN3D
    - Recent work in refine has primarily been based on multiscale metrics
    - Much of the recent goal-oriented adaptation research has been done with SFE due to the deep convergence characteristics of its solver for the primal problem and the preconditioned SLAT linear solver for the adjoint problem
      - **Stephen Wood will cover SFE goal-oriented adaptation in more detail next week**

Goal-oriented Adaptation (cont.)

- Goal-oriented refine expected input files:
  - `{project_rootname}_flow.solb` – the flow state to interpolate to the next mesh
  - `{project_rootname}_volume.solb` – the flow primal and adjoint states
  - Example command for a target complexity of 5000.0:
    - `mpiexec refmpi loop box03 box04 5000.0 --opt-goal`
- When using strong BCs for viscous walls, the adjoint state at the strong BC DOFs isn't coupled to the rest of the field. Refine can mask the strong BCs to avoid the discontinuities affecting the adaptation process:
  - `--mask --fun3d-mapbc {project_rootname}.mapbc`
- pyrefine can help automate this process for SFE-based goal-oriented adaptation as shown in this driver script code snippet:

https://github.com/nasa/pyrefine/blob/main/examples/onera_m6/steady_sa_sfe_goal/adapt.py

```
adapt_driver = AdaptationDriver(project, pbs)
adapt_driver.refine = RefineGoalOriented(project)
adapt_driver.refine.mask_strong_bc = True
adapt_driver.simulation = SimulationFun3dSFEAdjoint(project, fwd_omp_threads=2, adj_omp_threads=20)
adapt_driver.controller.save_all = True

adapt_driver.set_iterations(1, 20)
adapt_driver.run()
```

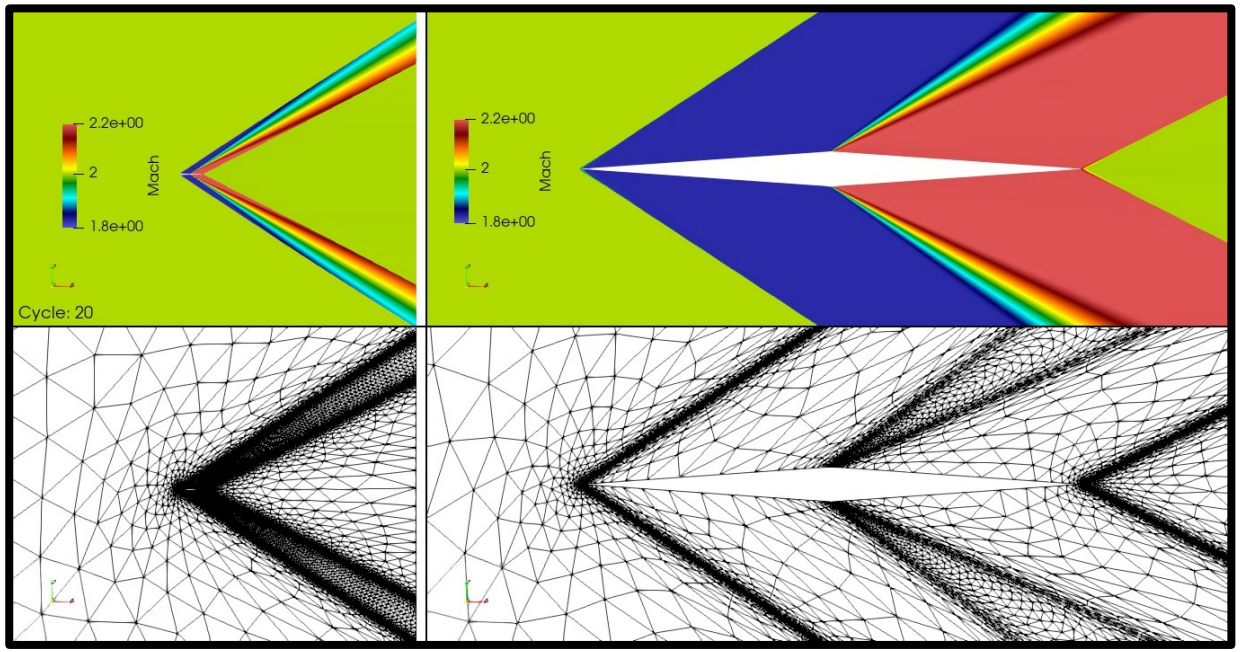
Examples

- Examples
  - Supersonic Diamond Airfoil
    - 2D, inviscid
    - C2S (CAD-to-Solution)
  - Subsonic NACA 0012 Airfoil
    - 2D, turbulent
    - C2S
  - Transonic ONERA M6 Wing
    - 3D, turbulent
    - C2S
  - Hypersonic Cylinder
    - 2D, 5-species air (generic gas path)
    - C2S
    - **Covered in training session 6 (inputs and structured grid)**
  - Hypersonic CEV Capsule
    - 3D, 5-species air (generic gas path)
    - C2S and Hybrid Approach (fixed surface/BL grid)
    - **Covered in training session 6**

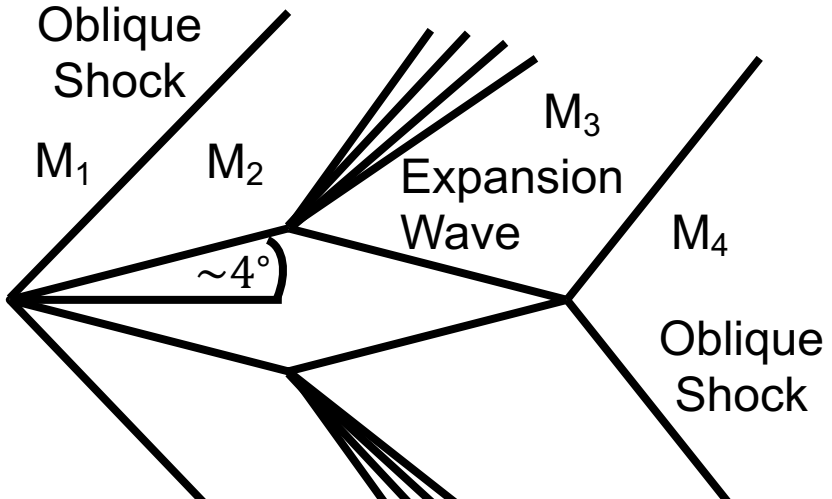
Supersonic Diamond Airfoil

- See **Ref. 2** for more details
- 2D mode used
- Inviscid
- Mach 2 freestream
- HANIM
- O(1) CPUhr

```
diamond.csm

box -10 -10 0 20 20 0

select edge
attribute bc_name $5000_farfield

UDPRIM $$/diamond thick 0.069756474

select edge
attribute bc_name $3000_diamond

subtract

dump diamond.egads

end
```



Value	Theory	CFD
M_1	2.00	2.00
M_2	1.86	1.86
M_3	2.15	2.15
M_4	2.00	2.00

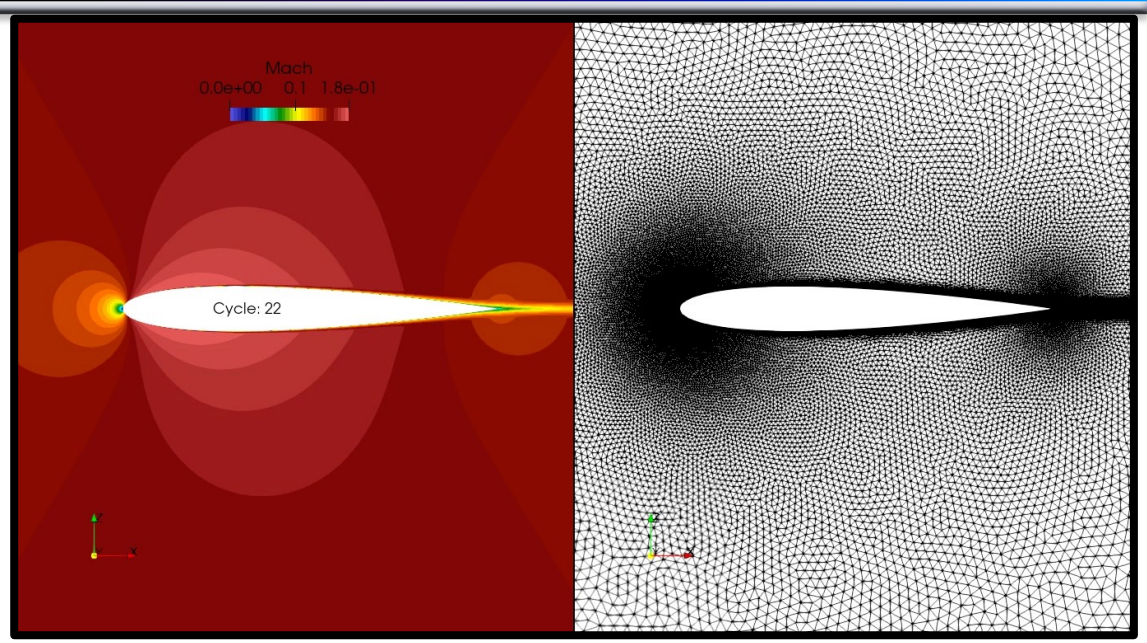



```
case_specifics

root_project 'diamond'
schedule_initial_complexity 4000
schedule_max_complexity 64_000
schedule_complexity_per_core 500
schedule_subiteration_limit 5
ref_cl('--gradation 10')
```

Subsonic NACA 0012 Airfoil

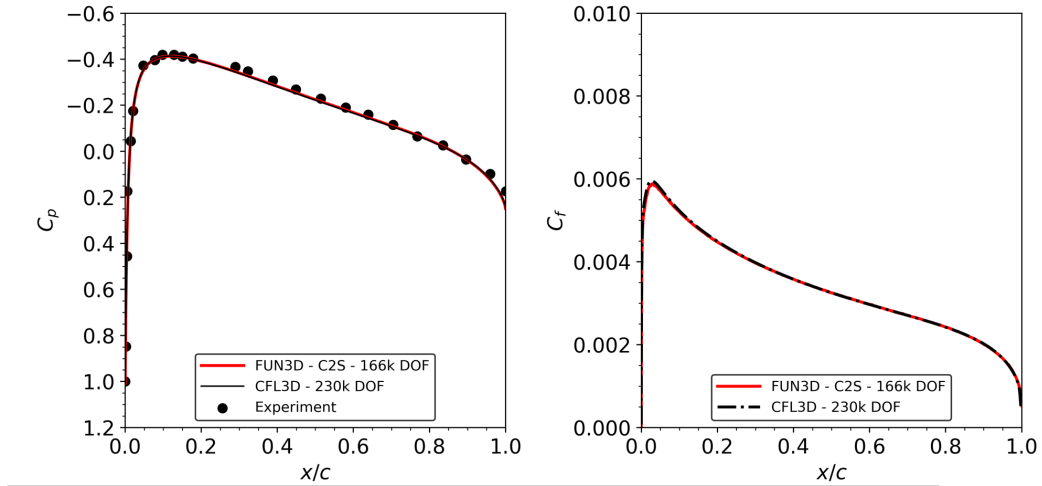
- https://turbmodels.larc.nasa.gov/naca0012_val.html
- 2D mode used
- Turbulent flow ($Re_c = 6\times10^6$)
  - "sa-neg" turbulence model
- Mach 0.15 freestream
- HANIM
- **Skin friction smooth**
- Wake adapted all the way to exit (bottom right picture)
- O(5) CPUhr

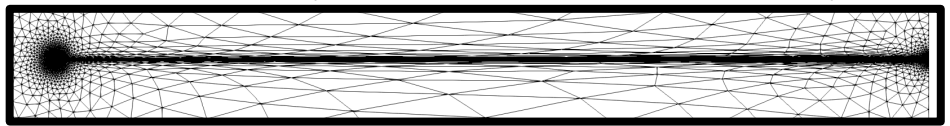
```
case_specifics

root_project 'naca'
schedule_initial_complexity 4000
schedule_max_complexity 128_000
schedule_complexity_per_core 500
schedule_subiteration_limit 5
ref_cl('--norm-power 4')
```



Solver	C_D
FUN3D C2S	0.00816
FUN3D	0.00812
CFL3D	0.00819

```
naca.csm

box -500 -500 0 1000 1000 0

select edge
attribute bc_name $5000_farfield

UDPRIM naca thickness 0.12 camber 0.00 sharpte 1

select edge
attribute bc_name $4000_airfoil

subtract

dump naca.egads

end
```





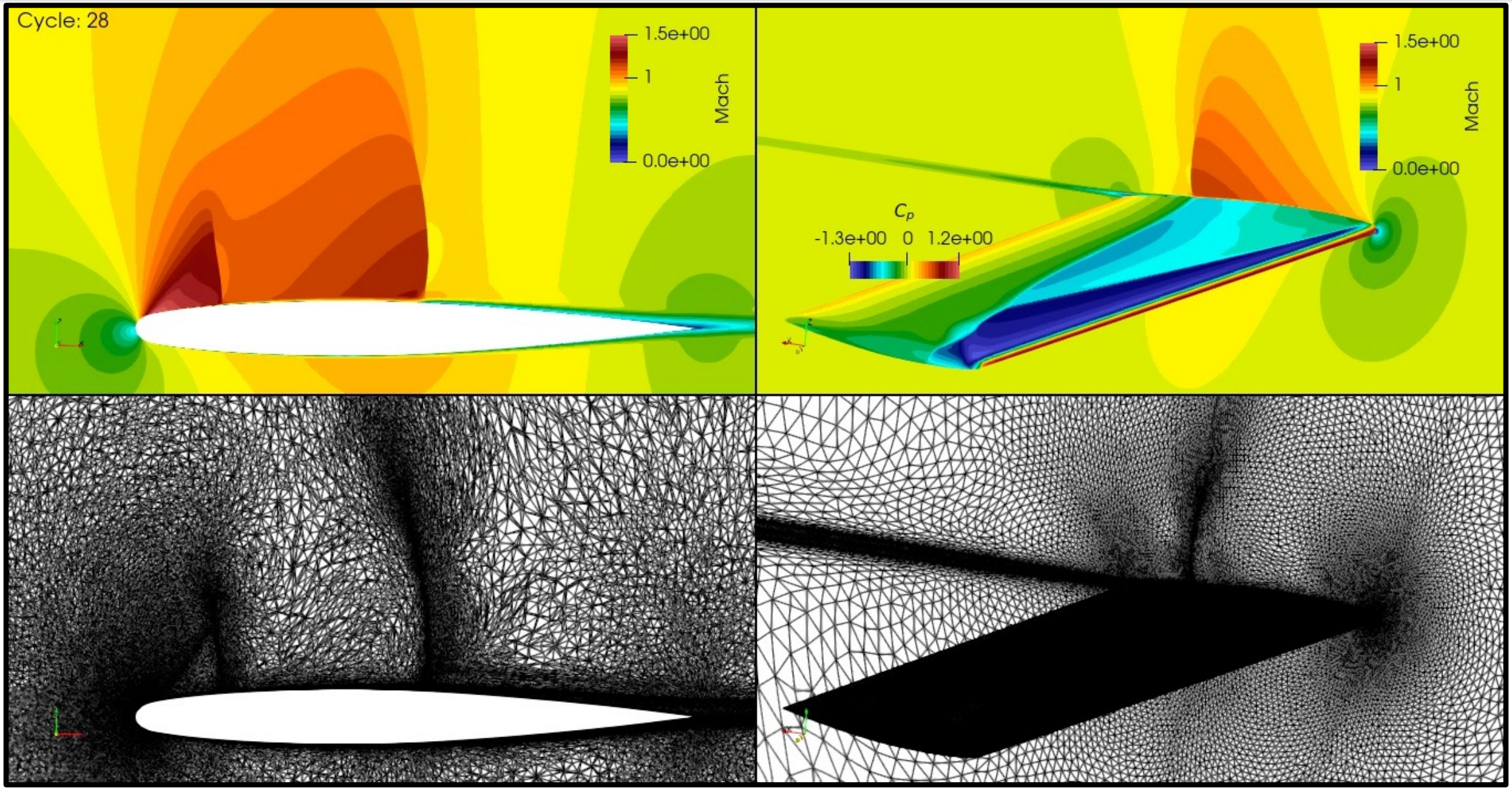
Transonic ONERA M6 Wing Setup

- https://turbmodels.larc.nasa.gov/onerawingnumerics_val.html
- See **Ref. 3** for more details
- 3D STEP file of ONERA M6 wing with a sharp trailing edge (om6ste)
- Turbulent flow ($Re_{c_r} = 14.6 \times 10^6$)
  - "sa-neg" turbulence model
- Transonic Conditions ($M_\infty = 0.84$)
- HANIM
- Ramping from 200k to 12.8M points
- O(1k-2k) CPUhr for 12.8M points
  - Reduce final complexity to reduce cost

```
om6ste.csm

despmtr radius 100.0
sphere 0 0 0 radius # outer boundary
select face
attribute bc_name $5000_farfield
import AileM6_with_sharp_TE.stp # wing step file
select face
attribute bc_name $4000_wing
scale 1.0/810.491484086 # scale to unit root chord
subtract # wing from sphere
box -2*radius 0 -2*radius 4*radius 2*radius 4*radius
select face
attribute bc_name $6662_ysymm
intersect # half domain
dump om6ste.egads
```

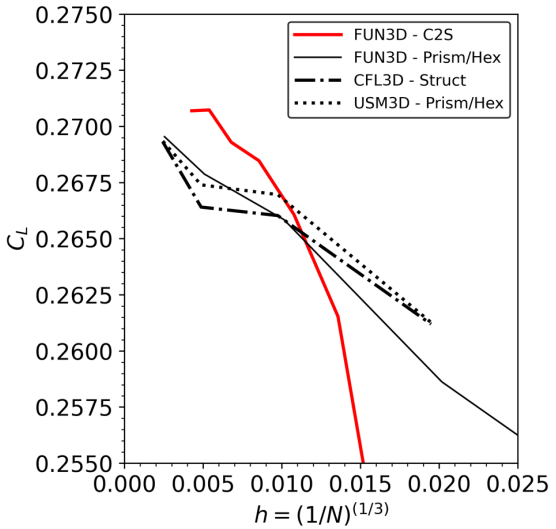
```
case_specifics

root_project 'om6ste'
schedule_initial_complexity 100_000
schedule_max_complexity 6_400_000
schedule_complexity_per_core 500
schedule_subiteration_limit 5
ref_cl('--norm-power 4 --buffer')
```

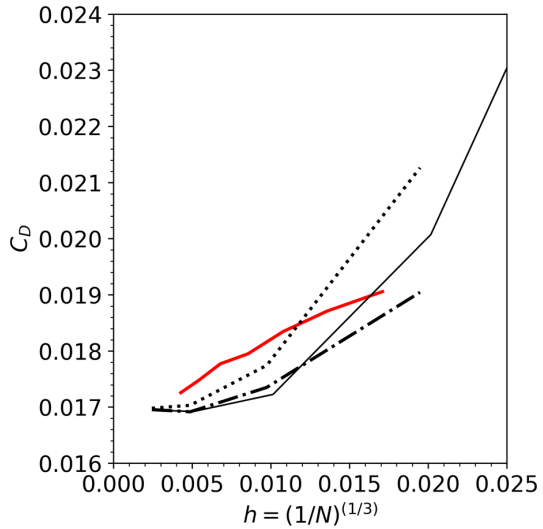
Transonic ONERA M6 Wing Results

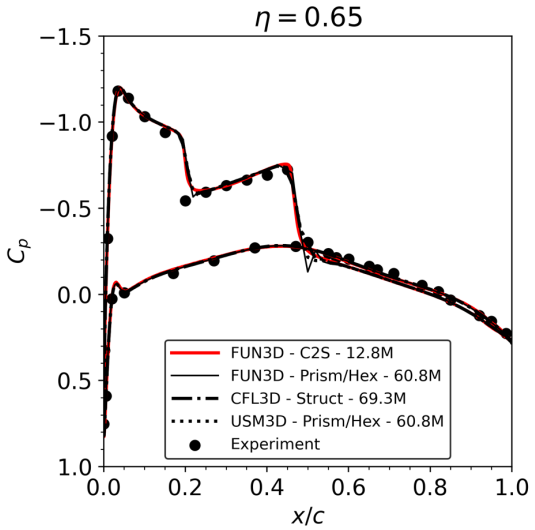


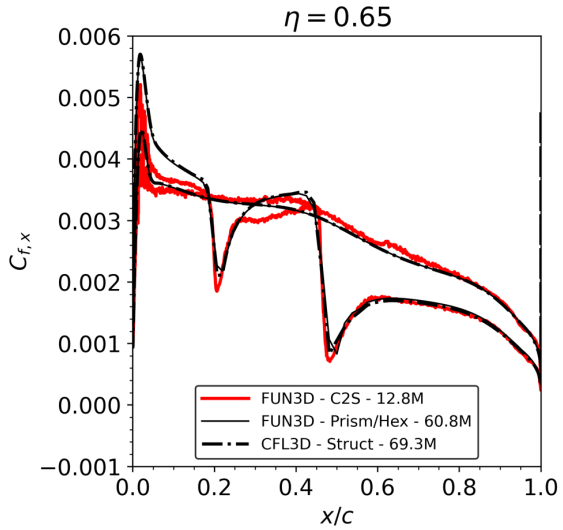
Transonic ONERA M6 Wing Results (cont.)

- Surface pressures agree well with experiment and other solvers on fixed grids
- Skin friction comparable, but noisier
  - $y^+ \sim 1$ comparison needed
- Surface $y^+ \sim 5 \rightarrow$ more points needed to reduce to 1
- Boundary layer growth determined by adaptation
- Surface grid anisotropy large
- Overall results reasonable considering completely automated grid





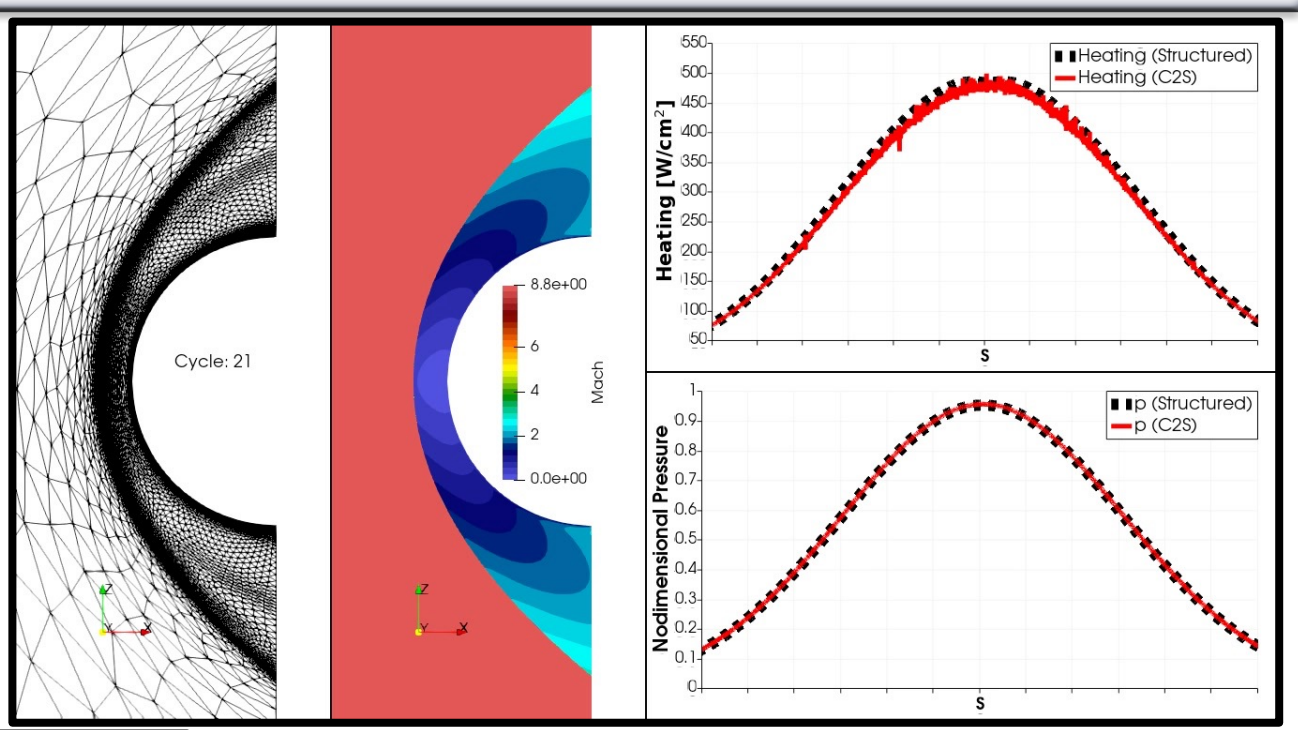


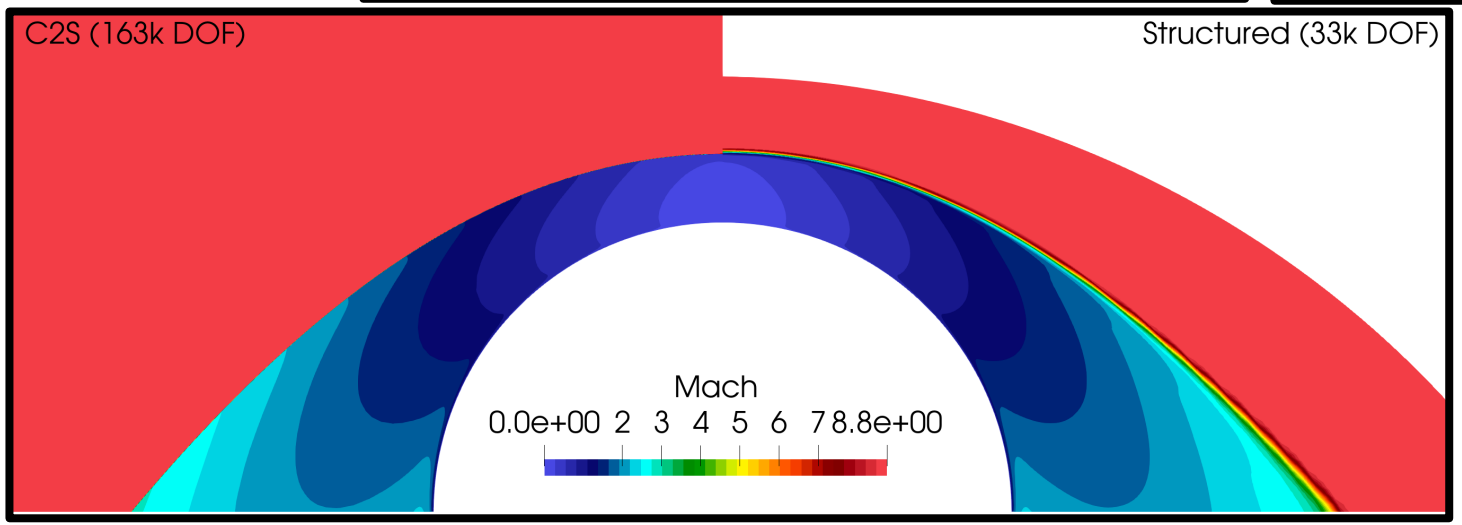


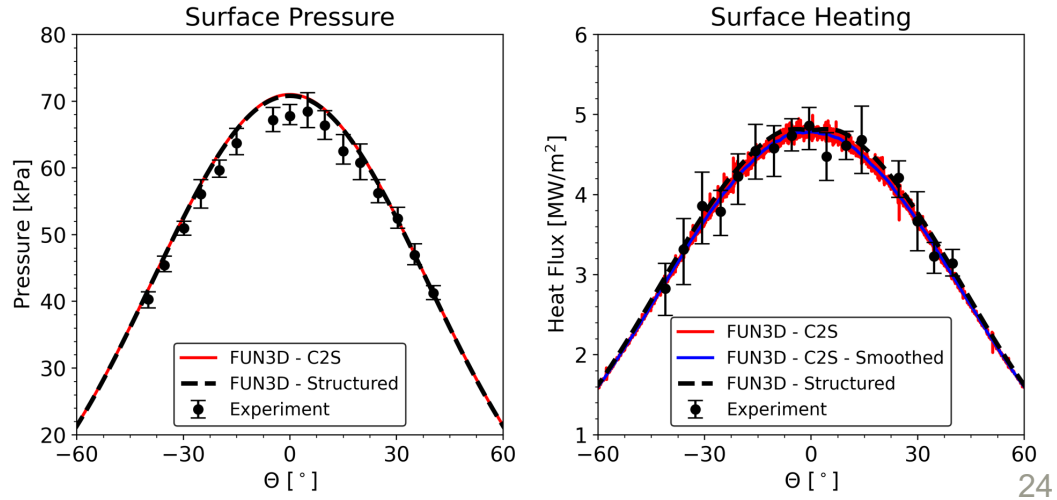
Hypersonic Cylinder Example

- **Example covered in training session 6**
  - **Structured grid inputs in session 6 files**
- 3D mode used (**2D mode not fully supported for generic gas**)
  - 1 cell prism extrusion - 2 y-symmetry planes
- High-enthalpy hypersonic (Mach = 8.7) flow around a cylinder (see **Ref. 5**)
- 5-species air, one-temperature gas model
- Laminar flow
- Gradation (stretching ratio) helps to stably capture shock
  - Gradation limits anisotropy in BL ~ O(10)
  - **Default gradation recommended for 3D problems for C2S**
- **Heat transfer noisy but generally within 1-2% of structured grid result**
- Structured grid has 5x less points (~33k vs ~163k)
  - AR in BL for C2S O(10-100) depending on gradation, AR in BL for Structured grid is O(3000)
  - Hybrid approach (frozen surface/BL) enables arbitrary AR to reduce DOF
- O(40) CPUhr

```
ref_cl("--norm-power 4 --gradation 10 --interpolant
#{project}_sampling_geom1.solb")
```

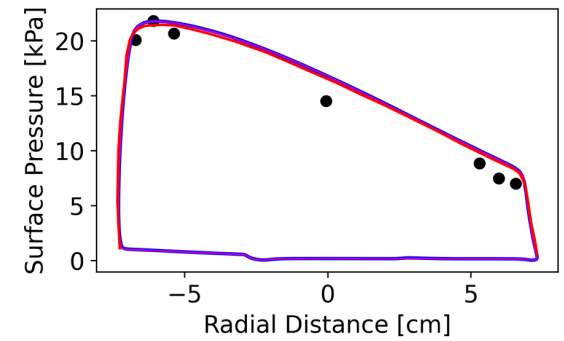


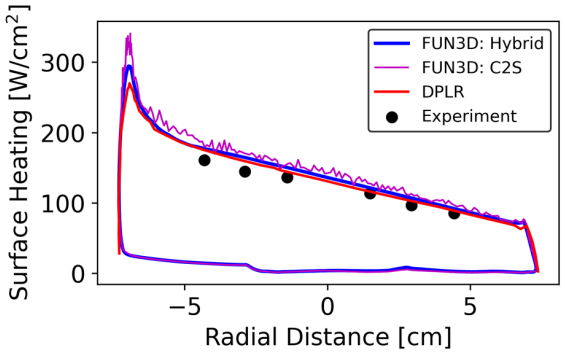


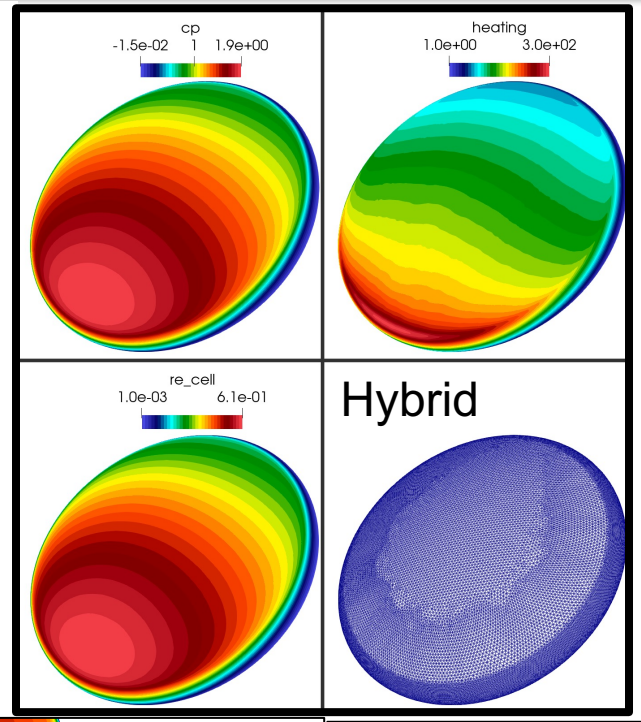


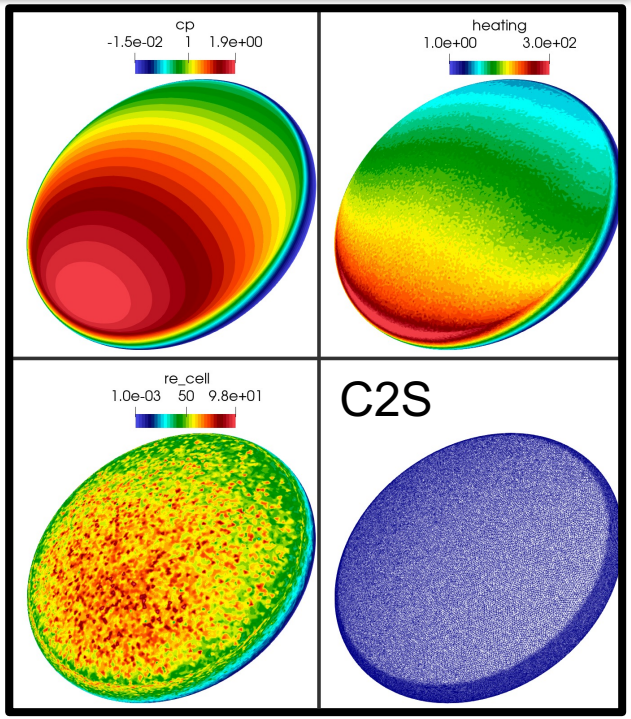
Hypersonic CEV Example

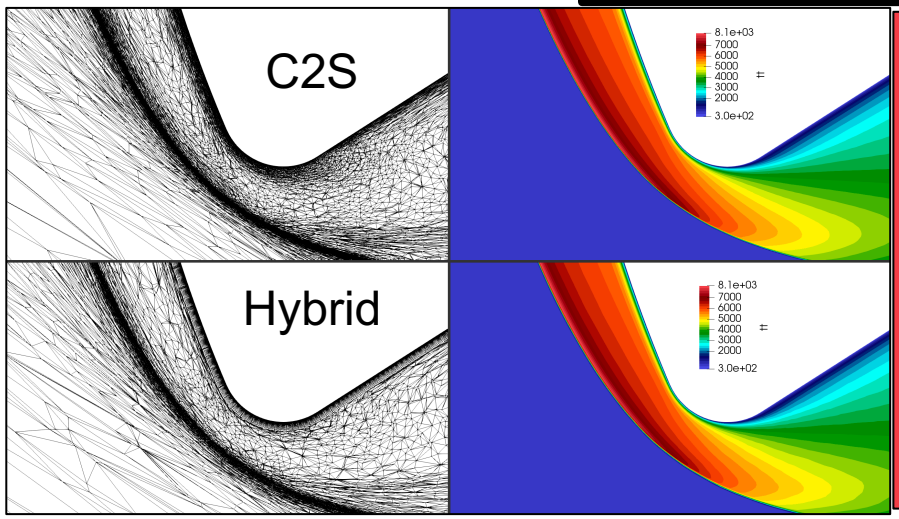
- **Covered in training session 6**
- High-enthalpy hypersonic flow (Mach ~9) over CEV
  - $\alpha = 28°$
- Experimental and DPLR data available on centerline
- 5-species air, one-temperature gas model
- Laminar flow
- Cold non-catalytic wall
- See **Refs. 4/5**
- Runtime O(1k-2k) CPUhrs
  - C2S (~6.4m points)
    - Can reduce target complexity to make it cheaper
  - Hybrid (~4m points)
- Note C2S spacing O(100) times coarser on the surface versus hybrid

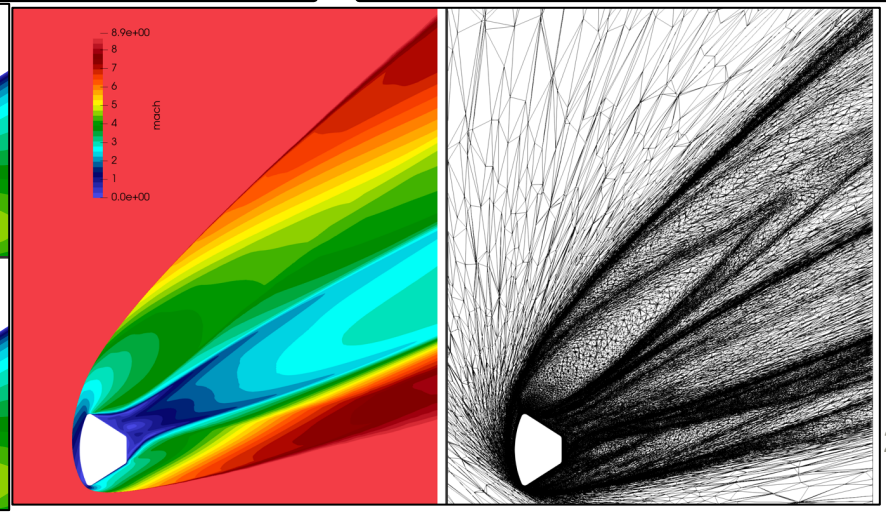




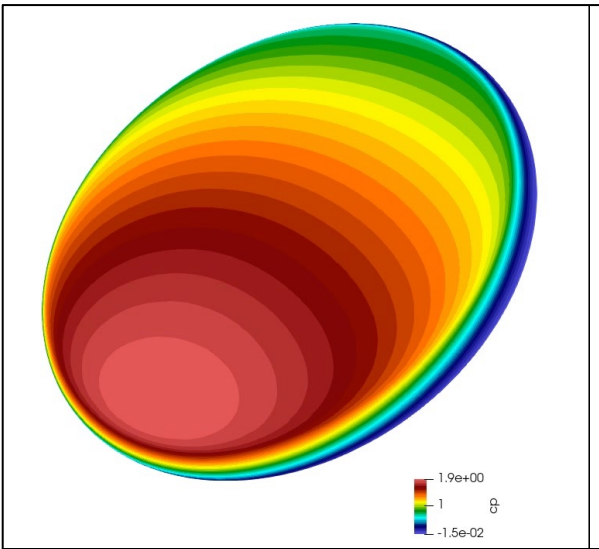


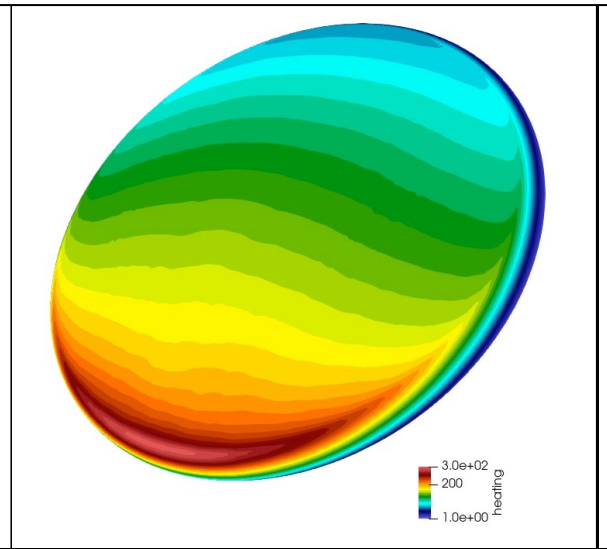


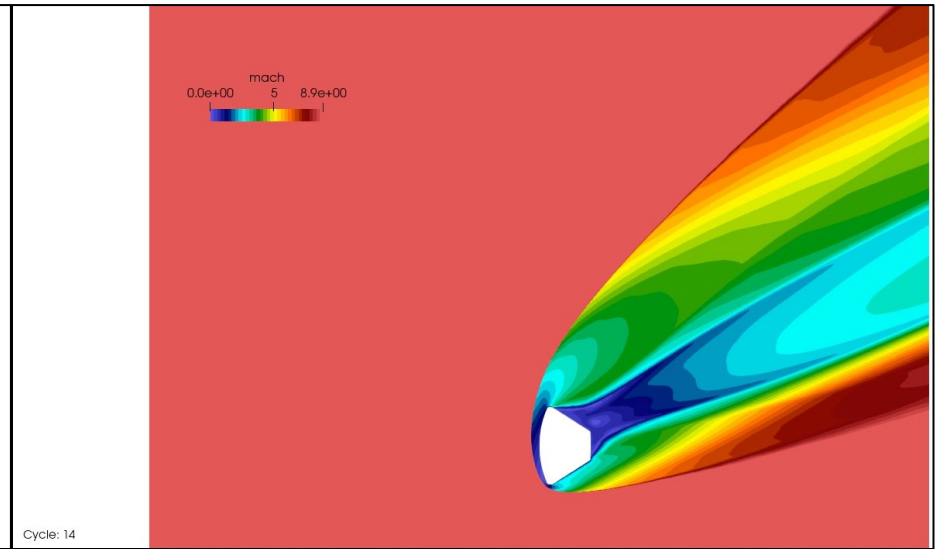


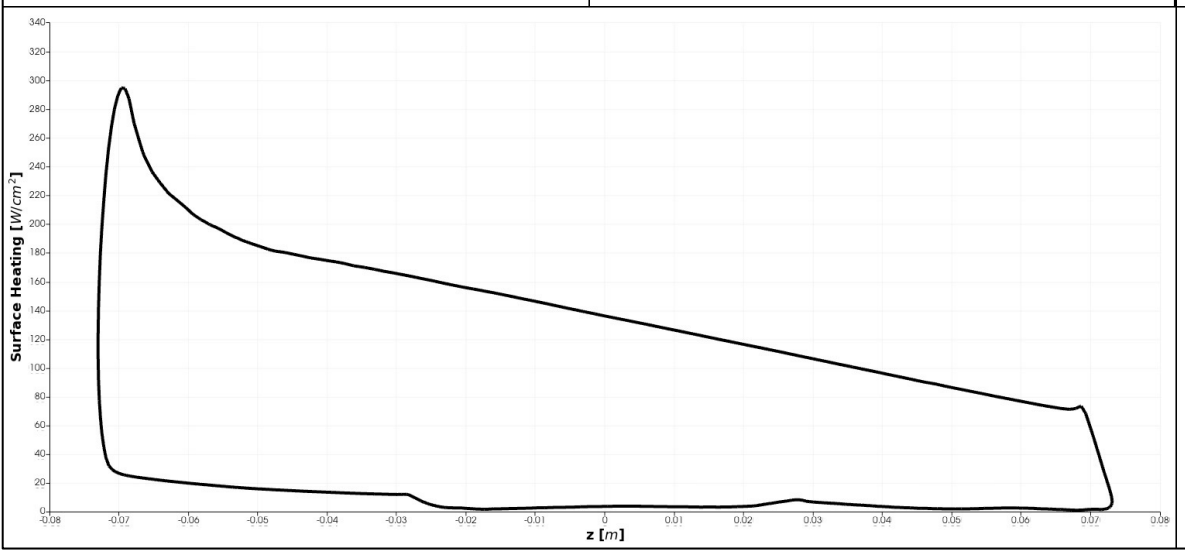


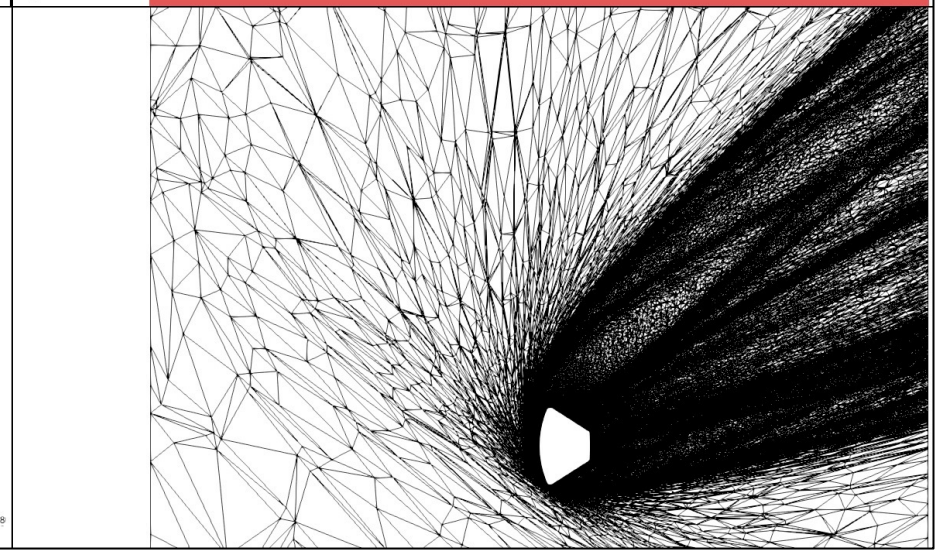
Hypersonic CEV Hybrid Results

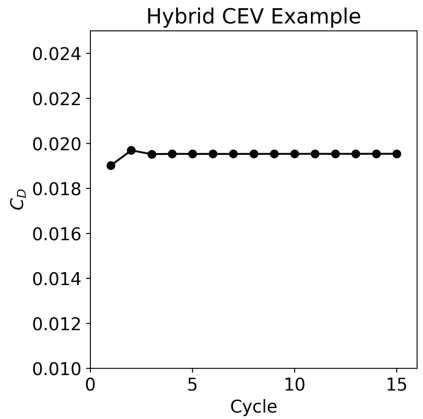




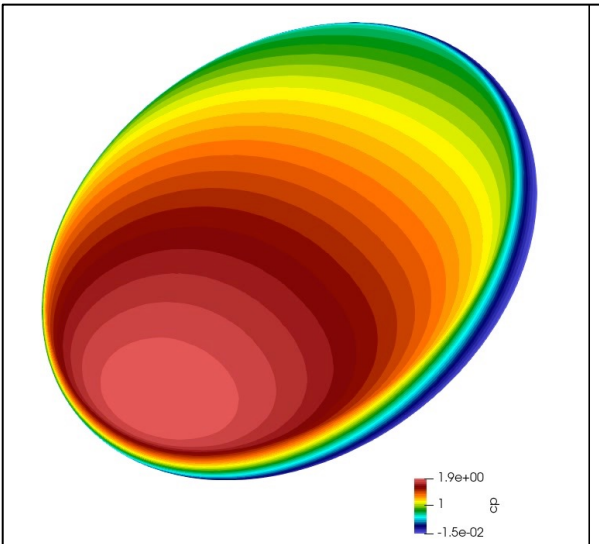


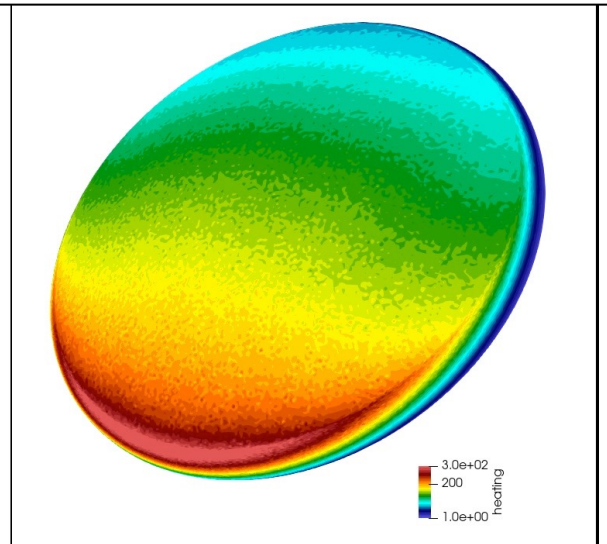


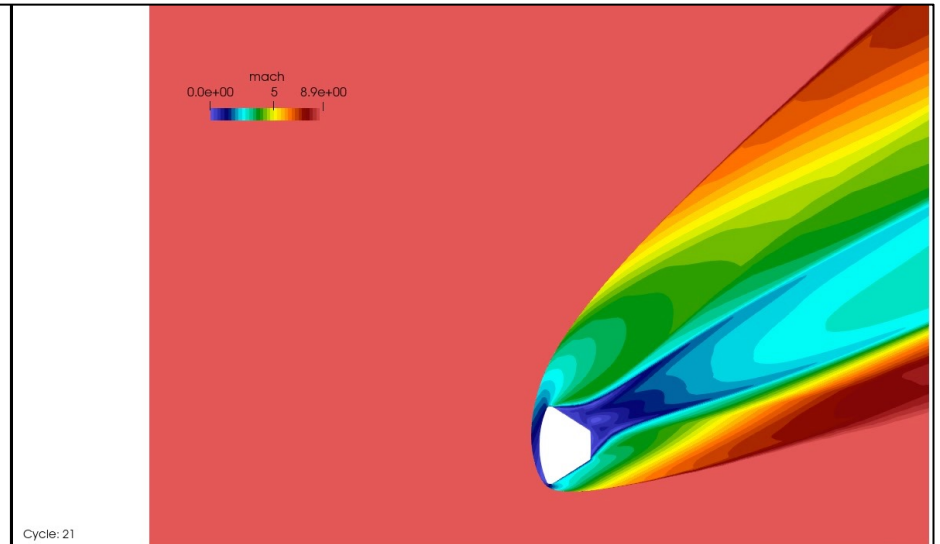


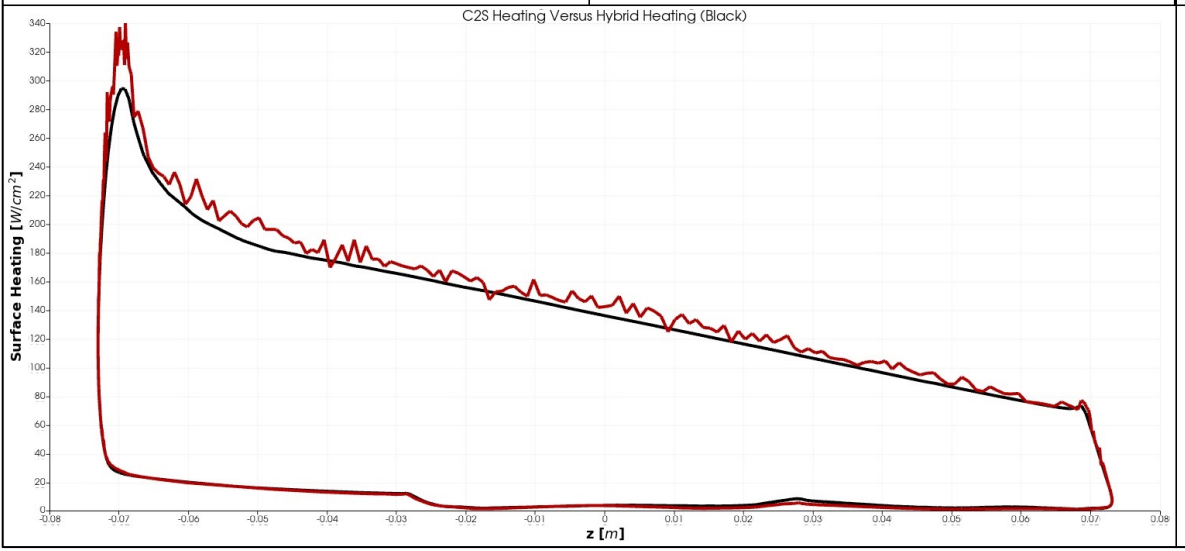


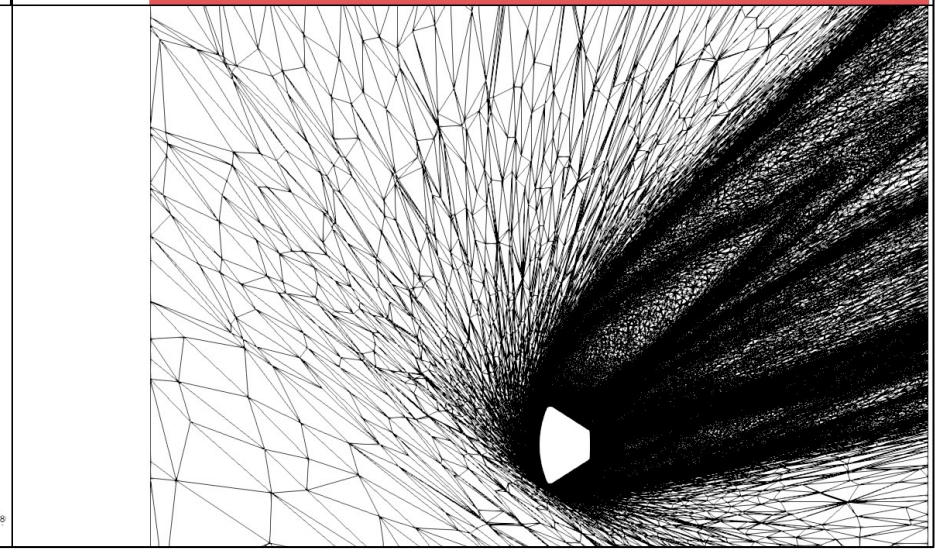
Hypersonic CEV C2S Results

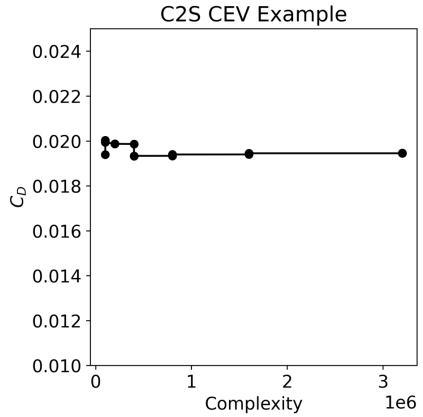












Where Things Can Go Wrong

- Interpolation of a bad solution can cause divergence
  - Generally observed during initial startup for high hypersonic flows
  - Workaround: restart from scratch for another cycle or two
    - Set import_from to blank (“”) in case_specifics for iteration < X
- Clipping of solution occurred (e.g., min density on surface, this can be reduced → see manual)
  - Same as above, you may want to try restarting from scratch on that grid. Often this is unrecoverable without doing so.
  - Generally, doesn’t impact solver stability or overall solution dramatically, but heating/shear will not be correct on these points
  - **Recommended to use as clean and as simple of a geometry as possible**
    - **Adaptation will commonly uncover geometric issues, especially with CAD adaptation**
- Hybrid adapted grids are not smooth and have gaps
  - For hybrid strategy, you will need some minimum number of points or else refine will generate gaps in grid (due to fixed surface/BL, not a problem for CAD approach with pure tetrahedra)
- Complexity ratio is much larger than 2 for hybrid adapted grids
  - Increase complexity, as complexity → ∞, grid points tend to 2*complexity
- Shocks are not well captured with adaptation
  - Use more points if you can afford it
  - Reduce problem domain (no wake, symmetry)
- Problem diverges early on
  - Try HANIM if available
  - Reduce CFL (minimum 2.0) and increase schedule and first order iterations
  - If using EBV, try default CBV approach

Summary

- Prerequisites
- Grid Adaptation
  - With CAD
  - Without CAD (hybrid frozen surface and BL approach)
- Solution Strategies
- Example Inputs
- pyrefine
- Goal-oriented Adaptation
- Examples
  - Supersonic Diamond Airfoil
  - Subsonic NACA 0012 Airfoil
  - Transonic ONERA M6 Wing
  - Hypersonic Cylinder
  - Hypersonic CEV Capsule
- Where Things Can Go Wrong
- References

References

1. **Goal-oriented Mesh Adaptation**
   - Paper: https://doi.org/10.1016/j.jcp.2021.110340
2. **Diamond Airfoil Adaptation Example**
   - Paper: https://ntrs.nasa.gov/citations/20160006030
3. **ONERA M6 Wing Anisotropic Mesh Adaptation Verification**
   - Paper: https://ntrs.nasa.gov/citations/20200003084
4. **Multi-architecture FLUDA details, includes capsule/Dream Chaser grid adaptation examples**
   - Paper: https://ntrs.nasa.gov/citations/20220016937
   - Presentation: https://ntrs.nasa.gov/citations/20220017092
5. **Hybrid grid adaptation**
   - Paper: https://ntrs.nasa.gov/citations/20210023222
   - Presentation: https://ntrs.nasa.gov/citations/20210024972

Public Community Questions: fun3d-users@lists.nasa.gov
Private/Proprietary Questions: fun3d-support@lists.nasa.gov