FUN3D v14.0 Training
Thermochemical Nonequilibrium Simulations

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

- Relevant known issues for 14.0
- Brief overview of generic gas path
- Recent improvements
- Solution strategies for blunt body and engine flows
- Hypersonic flow over a cylinder example
- Grid adaptation with and without CAD
- Hypersonic flow over a capsule examples
- References

Relevant Known Issues for 14.0

- Regression: heating is not smooth across patch boundaries
  - BC precedence was changed in generic gas path to match perfect gas path
  - Heating is only an output unless you use radiative equilibrium wall
  - Workaround: surface patches of interest should be lumped (“bc” or “family”)
  - For CPU runs, setting `heating_from_integral` to false (not default) is another workaround
    - Not implemented for GPU in 14.0 and should not be used
  - **14.0.1** fixes these and regression tests were added
- Volume/sampling output for equilibrium air gas models does not work
  - **14.0.1** fixes this
- GPU STVD inviscid flux scheme has a bug for coupled “kw-sst” turbulence model
  - **14.0.1** fixes this

Brief Overview of Generic Gas Path

- Provided and built by default with the standard FUN3D release
- See “A Tutorial for the Generic Gas Path in FUN3D”, NASA-TM-2014-218658 and past 2018 session for more information
- Path solves the thermochemical nonequilibrium flow equations
- Lineage of the path is NASA LAURA and uses LAURA inputs:
  - tdata (gas model, species input as **mass fractions**)
    - Elemental basis currently required for all species (e.g., CO_2 requires C and O_2 or O)
  - species_thermo_data (NASA9 thermodynamic data)
  - species_transp_data_0 (high order collision integral fits for transport properties)
  - kinetic_data (chemical mechanism, **CGS units**)
    - Bimolecular and termolecular reactions do **not** have the same units and thus have different conversion factors between MKS/CGS unit systems
- Provided thermodynamic, transport, and kinetic data files in installation directory used if they don’t exist in case directory
- Order of equations/residuals is species, momentum, energies, and turbulence

Primary Input Differences from Perfect Gas Path

- Gas model needed at a minimum (tdata file)
- Inputs are mostly dimensional, check manual to make sure
- Default CFLs are set for transonic flows
- Default wall boundary conditions are not valid and need to be set explicitly

```
&reference_physical_properties
  dim_input_type        = 'dimensional-SI'
  velocity              = 5000.0 ! m/s
  density               = 0.001  ! kg/m^3
  temperature           = 200.0  ! K
  gridlength_conversion = 1.0    ! Conversion factor to meters
/

&boundary_conditions
  wall_temp_flag(1)       = .true.
  wall_temperature(1)     = 500.0 ! K
  wall_catalysis_model(1) = 'non-catalytic'
/
```

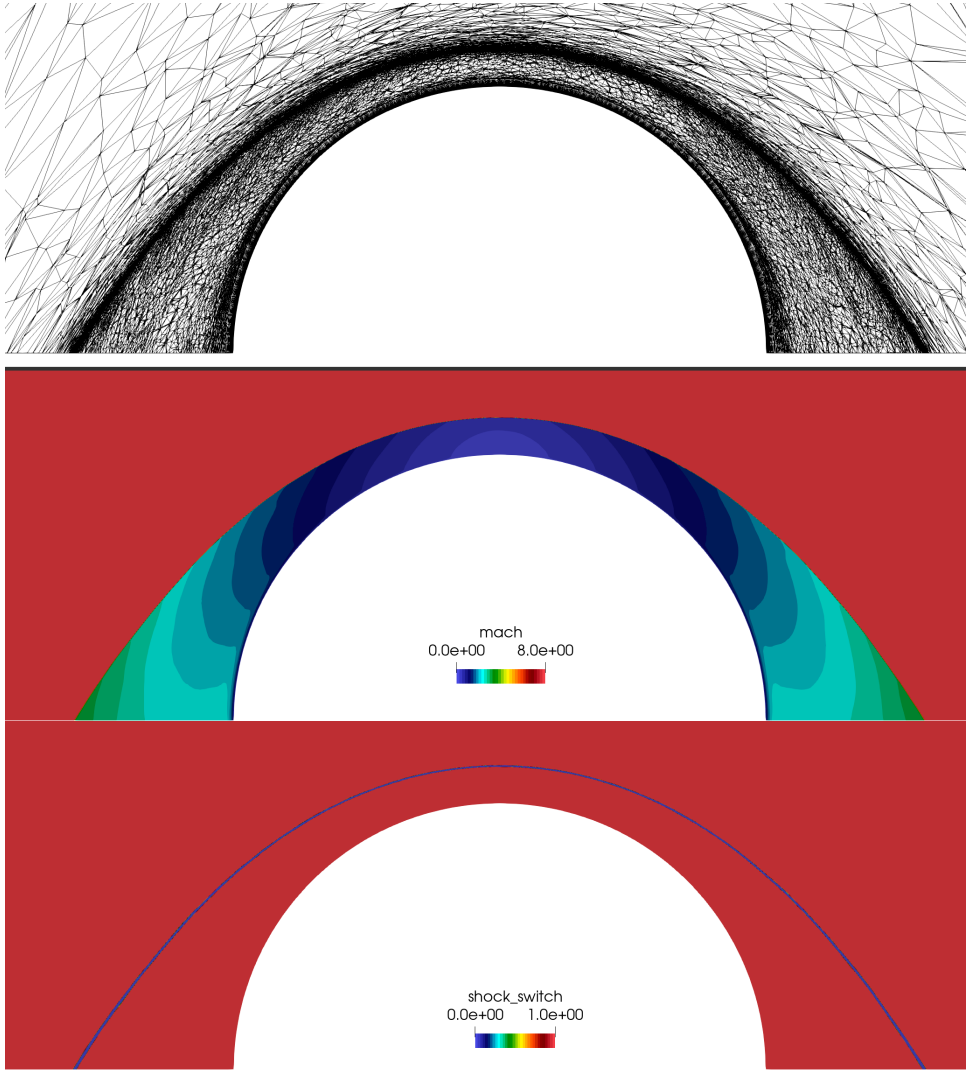
```
tdata file 1:
one
N2 0.767
O2 0.233
NO
N
O

tdata file 2:
perfect_gas
```

Turbulence Models

- Generic gas path was originally developed with fully coupled turbulence models
  - Differ from decoupled turbulence models for perfect gas and incompressible gas paths
- Coupled turbulence models (use meanflow CFL input):
  - ‘sa-catris’ (des option with `&spalart des` flag)
  - ‘kw-sst’
- **Experimental decoupled turbulence models, new in 14.0**
  - Same models as perfect gas and incompressible gas paths
  - Requires cflturb inputs (change defaults as they are set for transonic flows)
  - Requires setting **`use_decoupled_turbulence_gen`** to .true. (and should be false otherwise)
  - ‘sa’, ‘sa-neg’, ‘des’, ‘des-neg’, ‘sst’, ’sst-v’ currently supported
- View variables with turb1/turb2/mu_t_ratio/mu_t outputs
  - Extra eigenvalue dissipation for coupled models can impact TKE for two-equation models
- SA-based models are generally more robust and recommended to start with

HLLE++ Inviscid Flux Update

- HLLE++, first implemented in NASA OVERFLOW, has been adapted and incorporated
- HLLE++ is an adaptation of the Roe scheme
  - Removes need for an entropy fix
  - Greatly reduces susceptibility to carbuncles
  - More numerically dissipative algorithm inside shocks
  - Incorporates an improved and stronger shock switch for stability with grid adaptation
- **Simply set inviscid flux construction to ”hlle++”**
- **“hvanalbada” is recommended flux limiter**
  - **h limiters are not valid for axisymmetric grids (only consider planar symmetry)**
  - **minmod_gg is only valid for stvd, e.g., use minmod for axisymmetric grids**
  - **Quarter symmetry recommended if you want axisymmetric problem**
- Shock sensor and shock switch coefficients have hidden inputs if you wish to change them (see **Ref. 3** for details), but defaults have been tested across variety of cases and flow regimes
  - `shock_sensor_alpha, shock_switch_alpha/beta` in `&inviscid_flux_method`
  - Can visualize with `shock_sensor` and `shock_switch`
- Hypersonic (Mach > 5) cases use a stronger shock switch
  - Needed to cleanly capture shocks with grid adaptation
  - For high quality shock-aligned structured grids or low grid resolution, you may want to turn this off if region is large with `adaptive_shock_sensor`



Hypersonic ($M_\infty = 8$) flow over a full 3D sphere
Centerline slice of grid, Mach, and shock switch

Viscous Flux Updates

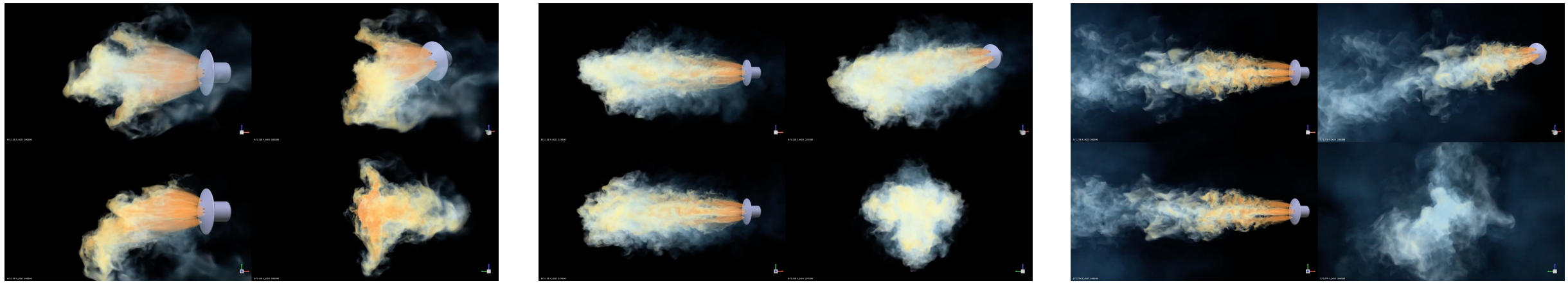
- Default viscous fluxes in FUN3D are computed with a cell-based viscous (CBV) method using Green-Gauss element-based gradients, equivalent to a Galerkin-based approach for tetrahedra
  - Utilizes compact nearest neighbor stencil
  - Jacobian includes full nearest neighbor stencil for viscous terms (14~ neighbors for tetrahedra, 26~ for hexahedra)
  - Involves loop over primal cells
  - For generic gas path, viscous Jacobians are about half of run time
- Edge-based viscous (EBV) approach has been developed in recent years and has been extended to generic gas
  - **Still considered experimental – not default**
  - See papers for more details (**Ref 4.**)
  - Effectively mathematically equivalent to CBV approach, same compact nearest neighbor stencil
  - Involves loop over edges of a virtual tetrahedralized grid
    - Can potentially impact stability as midpoint quantities are edge-averaged instead of cell-averaged
  - **Overall 2x+ speedup of whole run time for generic gas path** (perfect gas path sees 1.4x~)
- **Simply set ebv to true in &code_run_control**
- **Issues still present**:
  - Requires valid tetrahedralization of mixed-element grid
    - Current algorithm works on vast majority of grids but not all grids
    - Improved tetrahedralization algorithm is in development
  - Nonplanar surface quads can cause issues; common in pure hexahedra grids
    - This is being worked on
- **Recommended grid topologies for EBV are triangles on surface with prismatic boundary layers and tetrahedra currently (adapted grids).**

GPU Updates

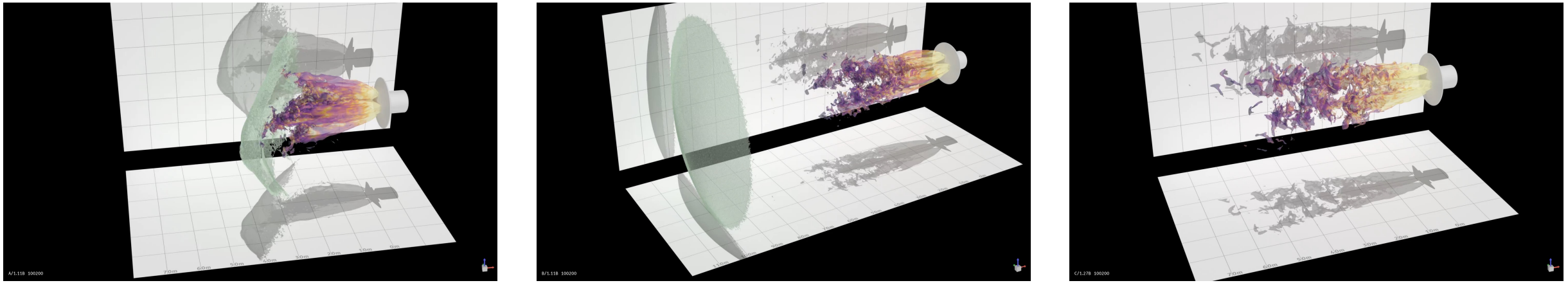
- Generic gas path is GPU-enabled in FUN3D Library for Universal Device Acceleration (FLUDA), see manual and **Ref. 1** for more details
- FLUDA is a primarily single-source, multi-architecture implementation capable of running performantly on CPUs and NVIDIA, AMD, and Intel GPUs and is the future version of the finite-volume path of FUN3D (see **Ref. 2**)
  - **Performance across architectures will generally scale with memory bandwidth (MBW); CBV option will see larger speedups on GPU.**
  - **NVIDIA 80GB PCIe A100 (1935 GB/s MBW per GPU) ≈ 600 x86 AMD EPYC 7742 Rome cores (409.6 GB/s MBW per dual-socket node)**
  - **Memory and runtime will scale roughly as number of equations squared, e.g., 5-species air versus PG ~ $9^2/5^2$ ~ 3.24x more memory and runtime**
  - **Current FLUDA release supports up to 16 species (library currently supports up to 40 species on GPU, but memory limitations above exist)**
  - **If you strongly want something not on the list below, please let us know**
  - **Use a decent portion of GPU memory (30%+) for maximum efficiency and for benchmarks**
- Supported options:
  - Gas Models: perfect gases, thermochemical nonequilibrium gases
  - Temperature Models: one and two-temperature models
  - Fluxes: Roe, STVD, HLLE++
  - Limiters: hvanalbada, hvanleer, hvenkat, hminmod, minmod gg (for STVD)
  - Coupled Turbulence modeling: SA-Catris with DES option, KW-SST
  - Experimental Decoupled Turbulence modeling: SA/SA-neg models with DES, SST, SST-V
  - Boundary Conditions: 3000 (tangency/slip wall), 4000 (no-slip wall, including adiabatic wall, constant temperature wall, radiative equilibrium wall, specified surface velocities), 5000 (farfield), 5026 (extrapolate), 5051 (back pressure), 6661/2/3 (symmetry), 7021 (subsonic inflow), 7100 (fixed inflow)
  - Catalytic Wall Models: All but equilibrium-catalytic
  - Time integration: Backward Euler for steady flows, all unsteady BDF schemes, local time-stepping
  - Time-averaging statistics
  - Specified rigid grid motion
  - Aeroelastic analysis using internal modal solver with modal mesh deformation
  - Asynchronous native volume outputs (zero runtime data output on parallel file systems)

Mars Retropropulsion GPU Simulations at Scale on DOE Summit

$M_\infty = 2.4$ $M_\infty = 1.4$ $M_\infty = 0.8$



Y_{H2O} 0.00 0.45



Shock visualization with isosurfaces of $Y_{H2O} = 0.40$, colored by vorticity magnitude
(each Cartesian grid line represents 10 m)
10-species DES, ~7 billion element grids running performantly on thousands of NVIDIA V100s (5,000~16,000) in a few days for several second simulation times. See **Ref. 5**.

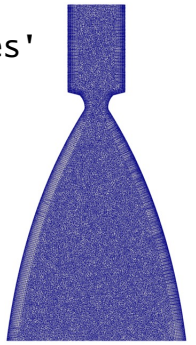
High-Speed Flows over Blunt Bodies

- Inviscid flux construction recommended to be “hlle++” with “hvanalbada” flux limiter
- Use the right gas model for the flow condition; e.g., perfect gas versus one-temperature 5-species air versus two-temperature 11-species air
- Outflow planes recommended to be extrapolate (5026) or back pressure (5051) with small static pressure
  - If simulating forebody of vehicle and there is a boundary layer exiting a domain, 5051 is better due to characteristics (for subsonic flow, extrapolate is not correct)
- Simple CFL schedule with steady-state time stepping should generally work
  - Max local CFL is problem-dependent
  - 2.0-10.0 is recommended
  - Low hypersonic flows using simpler models (5-species air, one-temperature) with simple geometry can use larger CFLs
  - High hypersonic flows using more complex models (many species, two temperature) with complex geometry will require lower CFLs
- High hypersonic flows should use first order iterations and may need a larger than default CFL ramp
- Limiter buzzing is typical, can freeze limiters
- If heating prediction is goal, heat transfer on surface should be periodically monitored for convergence
  - Viscous force convergence is a surrogate; 14.0.1 will have option to output integrated heat transfer to screen
- Monitor surface cell Reynolds number (”re_cell”) which should be below 1 as conservative metric for converged heat transfer spacing (problem-dependent, can be higher), check ”yplus” for turbulent problems

Flows involving Engines

- The perfect gas path strategy is generally followed (see 2018 boundary condition tutorial)
- Simulating the subsonic plena of engines is the most common approach to modeling engines
- **Start with a standalone engine for simplicity before running a full vehicle**
- Inflow gas conditions are typically obtained from equilibrium analysis (e.g., CEA or Cantera)
- Total temperature and pressure input in SI units
- Flow initialization volume set in subsonic portion of engine using subsonic outward velocity
  - Chemically reacting flows may need outer freestream inert volume (e.g., pure N2 rather than air) depending on initial transients and reactions
- CFL schedule with steady-state time stepping
- First order iterations should be used
- Reducing update % of variables may be needed (max_frac_target)
  - Max and min temperature limits can also be set in &update_limits
- Max local CFL of 5 ramped after 2000 iterations with 4000 first order iterations is good first phase attempt
- Combine plenum and engine walls into a component to compute thrust and mass flow rate
- **{project}_stream_info.dat** file will show thrust and mass flow rate of components in various units, need to set grid_units properly
- Monitor mass flow rate and thrust over time using the **fm component files**

```
&flow_initialization
  number_of_volumes    = 1
  mks_units            = .true. ! set rho,u with SI/MKS units instead
  type_of_volume(1)    = 'cylinder'
  point1(1:3,1)        = A, 0.0, 0.0
  point2(1:3,1)        = B, 0.0, 0.0
  radius(1)            = R
  rho(1)               = X  ! kg/m^3 ~p0/(R_gas*T0)
  u(1)                 = Y  ! m/s ~ Mach 0.1-0.3
  temperature(1,1)     = Z  ! ~T0 [K]
  mass_fraction(2,1)   = 1.0
/
&boundary_conditions
  grid_units               = 'm'  ! 'cm','mm','feet','inches'
  plenum_t0(1)             = X    ! K
  plenum_p0(1)             = Y    ! Pa
  plenum_id(1)             = 2
  farfield_turbulence(1)   = .false.
  boundary_turbulence(1,1) = 0.01 ! turb variable vector
/
&component_parameters
  allow_flow_through_forces = .true.
  number_of_components      = 1
  gas_properties_bc(1)      = 1 ! Not used/needed for mdot/forces
  component_count(1)        = -1
  component_input(1)        = '1,3' ! plenum, and engine walls
  component_name(1)         = 'engine'
/
&update_limits
  max_frac_target = 0.2 ! Limits update to 20% rather than 50% default
/
```



Flows involving Engines

- Inviscid flux construction recommended to be “hlle++” or “roe” with “hvanalbada” flux limiter
- Engine walls recommended to be adiabatic non-catalytic to start with
- SA-based models are more robust and recommended to start with
  - Decoupled models may need to specify boundary turbulence variables as freestream is used by default and may not be appropriate
- Non-reacting pseudo species approaches are common
  - Perfect gas approach (shown to the right) with constant specific heat ratio (1.4) and arbitrary molecular weight
  - Collision integral fit pairs (species_transp_data_0 file) need approximated (e.g., N2 for air)
  - Alternatively, can use weighted average of actual species which will have temperature-dependent thermodynamics
  - Empty local kinetic data file is equivalent to frozen chemistry
- If a species does not exist, you can add them with some effort
  - NASA9 thermodynamic data: https://cearun.grc.nasa.gov/ThermoBuild/
  - If collision integral fits are not readily available, you can generate them from kinetic theory which is valid for temperatures observed in combustion
    - See https://doi.org/10.1063/1.1732130 and CHEMKIN transport theory for approach on computing $\pi\sigma^2\Omega^{(1,1)*}$ and $\pi\sigma^2\Omega^{(2,2)}$
    - See utils/collision_integral_lib.py (14.0.1)

```
species_thermo_data:
Fs
&species_properties
mol_wt = 28.96
/
3
0.00000000E+00  0.00000000E+00  0.35000000E+01  0.00000000E+00
0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
0.00000000E+00  0.00000000E+00   200.000  1000.000
0.00000000E+00  0.00000000E+00  0.35000000E+01  0.00000000E+00
0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
0.00000000E+00  0.00000000E+00  1000.000  6000.000
0.00000000E+00  0.00000000E+00  0.35000000E+01  0.00000000E+00
0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
0.00000000E+00  0.00000000E+00  6000.000 20000.000

tdata:
one
Fs 1.0
```

High-Enthalpy Cylinder (HEC) Example Inputs

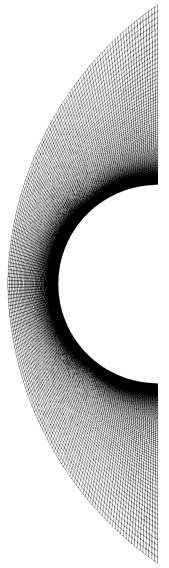
- High-enthalpy hypersonic (Mach = 8.7) flow around a cylinder
  - Experimental surface pressure and heat transfer data available
- 5-species air, one-temperature gas model
- Mass fractions obtained through precursor tunnel simulation
- $u_\infty = 4776\frac{m}{s}, \rho_\infty = 0.003255\frac{kg}{m^3}, T_\infty = 694\ K$
- Wall is modeled as cold (300 K) and non-catalytic
- Outflow is modeled with back pressure (5051) with a static pressure of 10.0 Pa (vacuum), can alternatively use extrapolate (5026) here
- Structured hex grid (not shock-fit) with 65k points
  - Surface properties are grid converged (tested up to 16x finer grid) but shock front is not as grid is not shock adapted
- Run as 3D with two y-symmetry planes
- 5000 iterations with max CFL 10 (use 5 for two-temperature)
- Runtime is a little over a minute on an NVIDIA V100
  - Don't benchmark the code on 65k points, saturate GPU memory if possible
  - Should not use EBV due to 2D grid

```
tdata:

one
N2 0.7356
O2 0.1340
NO 0.0509
N
O  0.0795
```

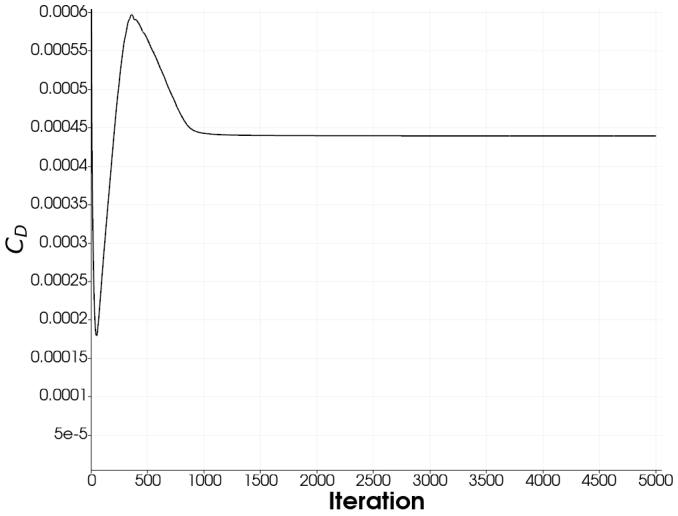
```
hec.mapbc

5
1              5000          inflow
2              6662          left
3              5051          outflow
4              6662          right
5              4000          wall
```

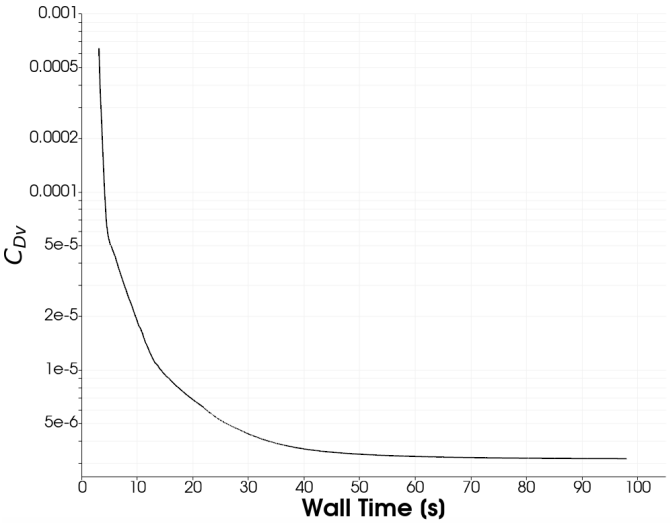
fun3d.nml:

```
&project
  project_rootname = "hec"
  case_title       = "high enthalpy cylinder
flow in HEG: run 627"
/

&raw_grid
  grid_format    = "aflr3"
  data_format    = "stream"
  patch_lumping = "family"
/

&governing_equations
  eqn_type      = 'generic'
  viscous_terms = 'laminar'
/

&reference_physical_properties
  dim_input_type = 'dimensional-SI'
  velocity       = 4776.0    ! m/s
  density        = 0.003255 ! kg/m^3
  temperature    = 694.0     ! K
/

&inviscid_flux_method
  flux_construction = 'hlle++'
  flux_limiter      = 'hvanalbada'
/

&global
  boundary_animation_freq = -1
  time_timestep_loop      = .true.
  estimate_remaining_time = .true.
  plt_tecplot_output      = .true.
/

&nonlinear_solver_parameters
  schedule_cfl = 0.1 10.0
/

&code_run_control
  steps              = 5000
  restart_write_freq = 5000
  restart_read       = "off"
/

&boundary_conditions
  static_pressure(3)      = 10.0 ! Pa
  wall_temp_flag(5)       = .true.
  wall_temperature(5)     = 300.0 ! K
  wall_catalysis_model(5) = 'non-catalytic'
/

&sampling_parameters
  number_of_geometries  = 1
  sampling_frequency(1) = -1
  type_of_geometry(1)   = 'plane'
  plane_center(1:3,1)   = 0.0 0.0 0.0
  plane_normal(1:3,1)   = 0 1.0 0
/

&sampling_output_variables
  mach           = .true.
  tt             = .true.
  mass_fr_i(1:5) = 5*T
  shock_sensor   = .true.
  shock_switch   = .true.
/

&boundary_output_variables
  cp             = .true.
  heating        = .true.
  mass_fr_i(1:5) = 5*T
/

&gpu_support
  print_crude_dev_mem = .true.
  use_cuda            = .true.
/
```

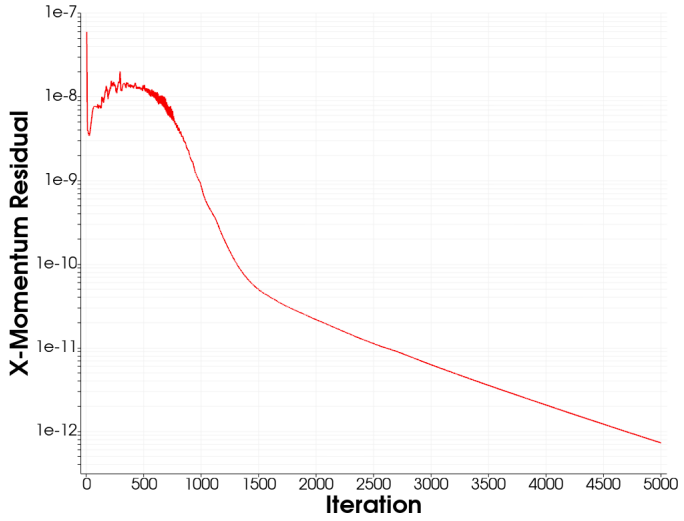


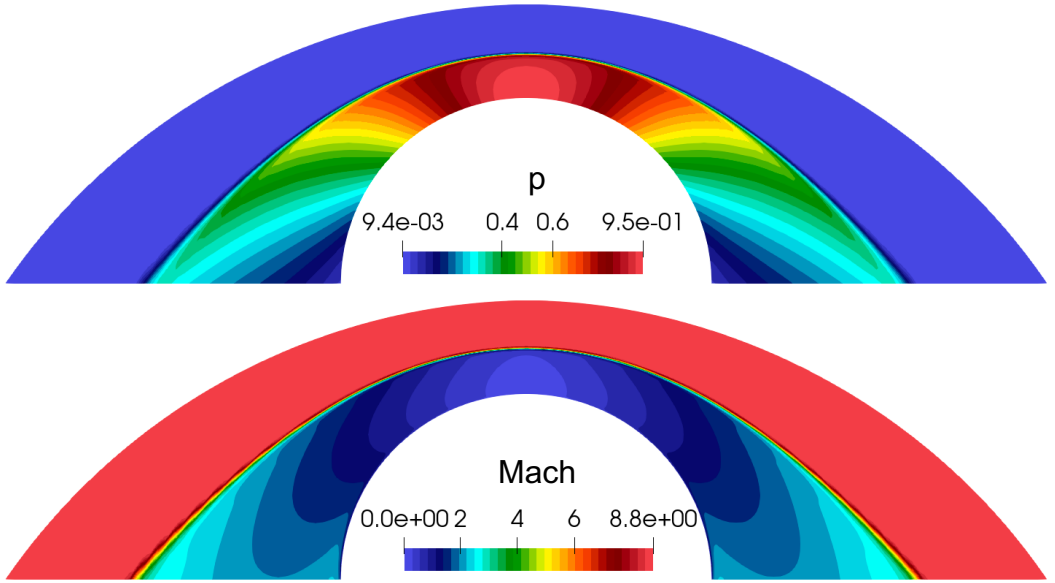
High-Enthalpy Cylinder (HEC) Example Outputs

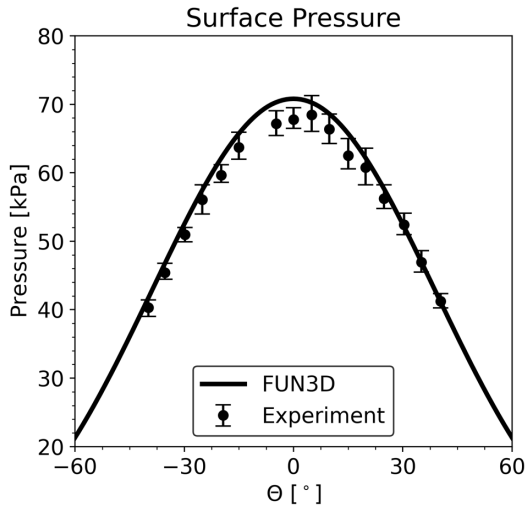


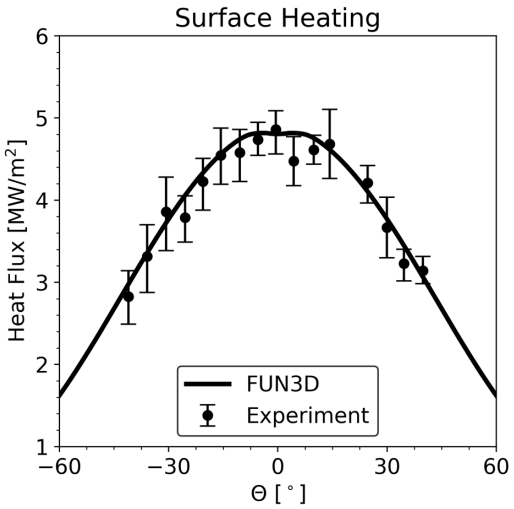


Viscous drag surrogate for heating convergence









Grid Adaptation

- Will be covered in more detail during grid adaptation tutorial, manual section is reasonably detailed
- Need to build FUN3D with ESP and OpenCASCADE (https://acdl.mit.edu/ESP/) for adaptation with CAD
- **Grid adaptation for hypersonic flows 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 a structured grid trivially, do so
  - Accurate heating and shear prediction will typically require finely resolving shocks
  - Grid adaptation can make your problem more expensive to run
  - If you use GPUs, grid adaptation 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
- Grid adaptation can be performed with and without geometry
- **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 using NASA refine 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
  - Utilizes ESP and the OpenCASCADE CAD kernel which CAD packages do not generally use
    - Expect issues and CAD cleanup to successfully import complex geometries not made with ESP
    - ESP and CAD cleanup will not be covered, lots of tutorials for ESP online, ESP has a GUI
  - 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 – **refinement only adapts tetrahedra**
  - Yields improved heating/shear prediction versus pure tetrahedra
  - Enforces desired surface and wall-normal spacing

Grid Adaptation

- Three approaches will be covered on the same setup
  - Hybrid approach using explicit commands
  - Hybrid approach using f3d Ruby script
  - C2S approach using f3d Ruby script
- Multiscale adaptation will be the focus here (no adjoint, or goal-oriented adaptation)
- There are Python wrappers alternatively which will not be covered now
  - https://github.com/nasa/pyrefine
  - pyrefine includes a simple browser-based GUI to see history information across adaptation cycles
- **Overall process:**
  - **Create initial grid**
  - **Run flow solver**
    - **Output solution and scalar metric (currently only one supported)**
      - **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**
  - **Adapt on provided metric**
    - **Output new grid and interpolate previous solution onto new grid**
  - **Repeat until you are done**

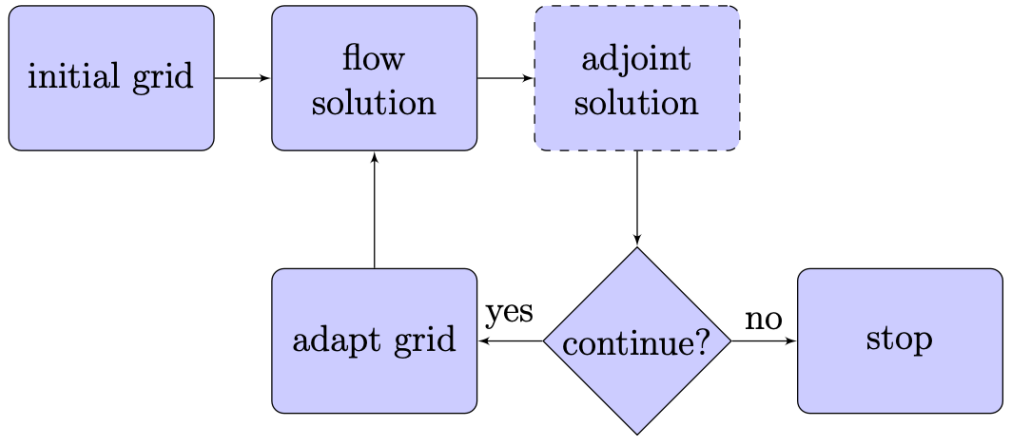
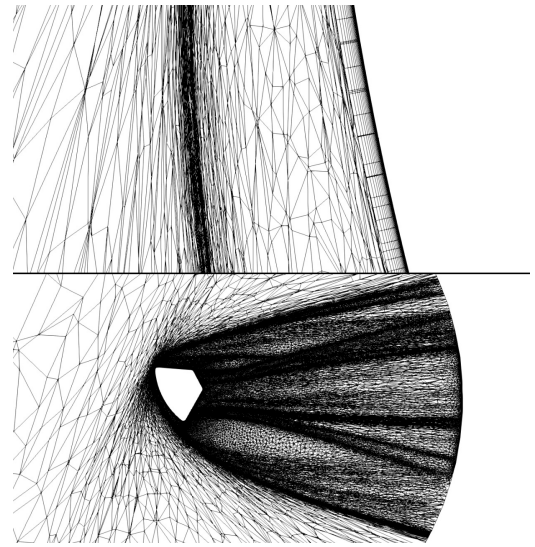
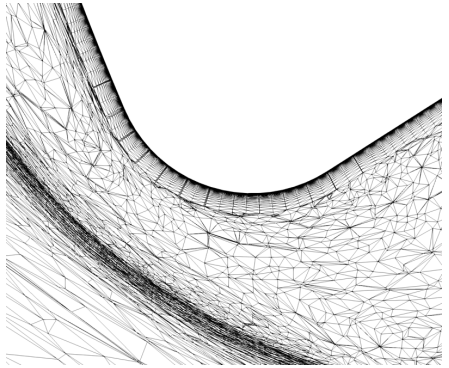


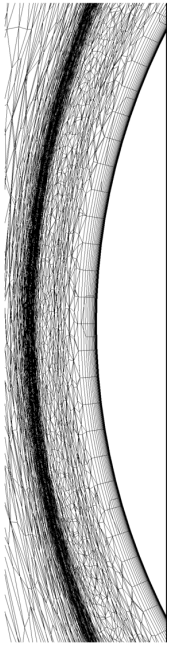
Figure 5: Solution-based grid adaptation process.

Grid Adaptation (continued)

- Number of points is a function of complexity (input into refinement)
  - ~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 is doubling complexity 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.**
- 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
    - Can reduce sweeps of refine (-s option): default is 30, 10 is recommended for hybrid approach, can go lower but grid quality will degrade slightly
  - 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
  - **Recommended grid approach is triangle surface grid and prism boundary layer; reduce hexahedra/pyramids as they are slower with refine**
    - **Use EBV if using prism/tetrahedra grids**







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
```

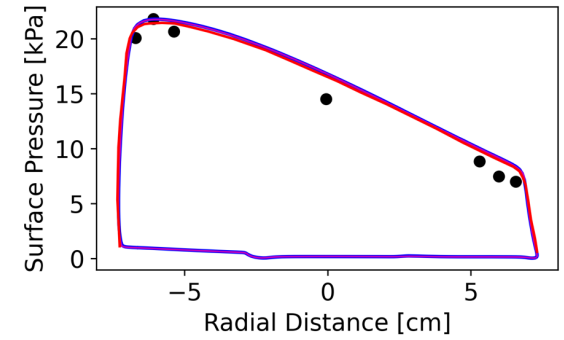
```
Run script:

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
```

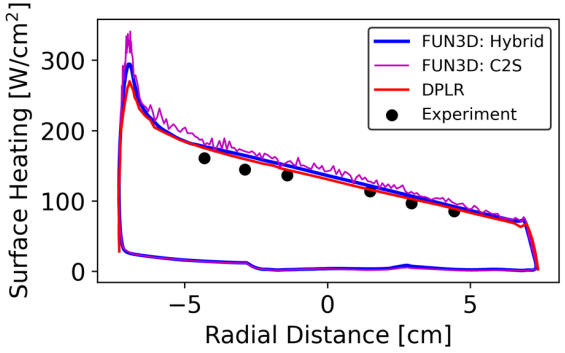
```
root_project 'cev'
schedule_initial_complexity 100_000
schedule_max_complexity 3_200_000
schedule_complexity_per_core 500
schedule_subiteration_limit 5
ref_cl("--norm-power 4 --interpolant #{project}_sampling_geom1.solb")
def iteration_steps
  refdist
  mesh_complexity = schedule_current()
  if (mesh_complexity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

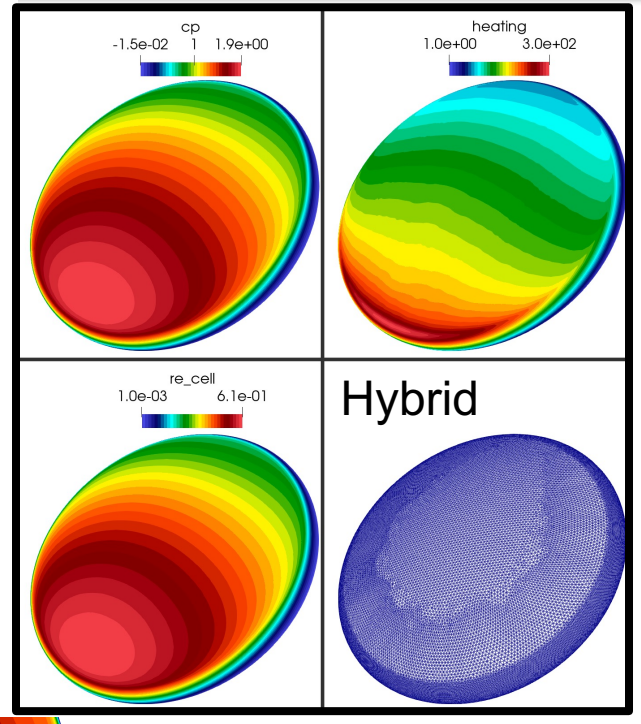
```
cev.csm

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
```

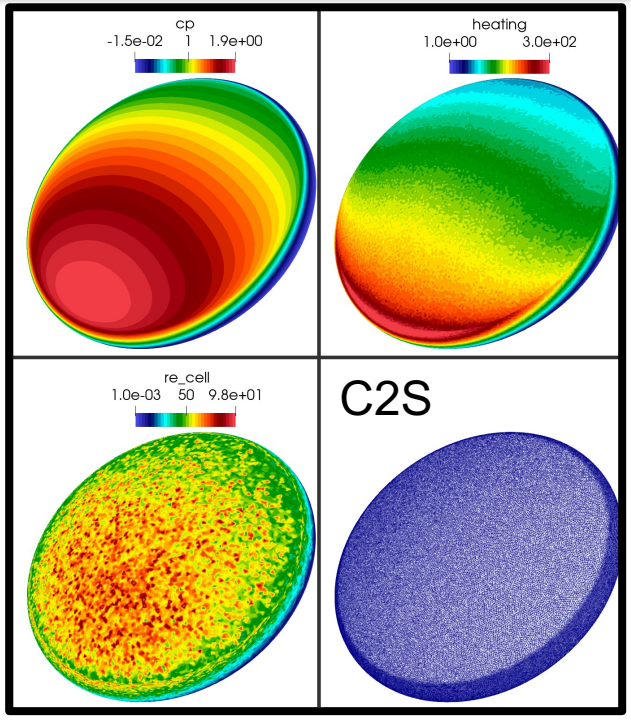
Grid Adaptation – CEV Example

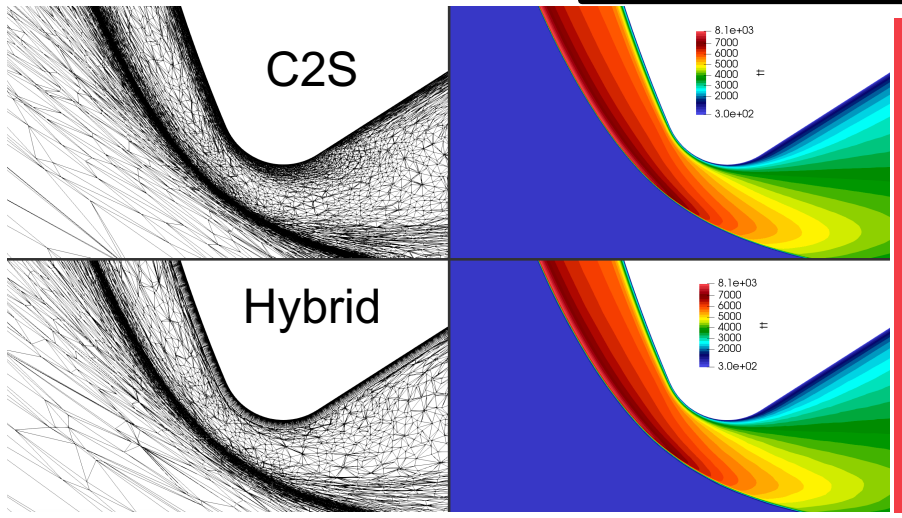
- High-enthalpy hypersonic flow (Mach ~9) over CEV
  - $\alpha = 28°$
- Experimental and DPLR data available on centerline
- 5-species air, one-temperature gas model
- Laminar flow
- Cold non-catalytic wall
- See **Ref. 2**
- Runtime O(1k-2k) CPUhrs
  - C2S (~6.4m points)
    - Can reduce target complexity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

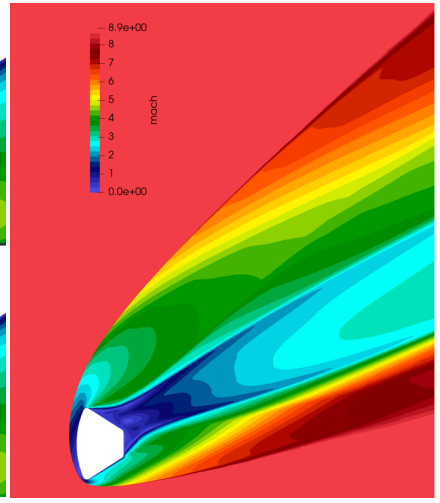


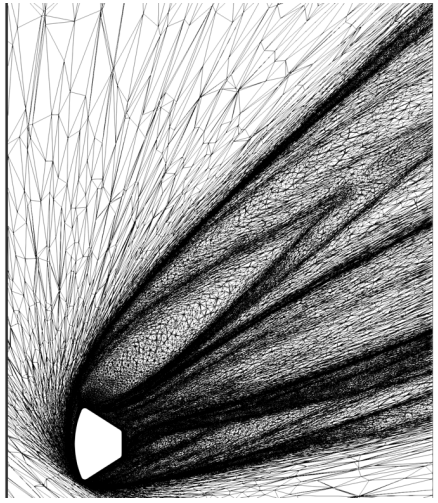




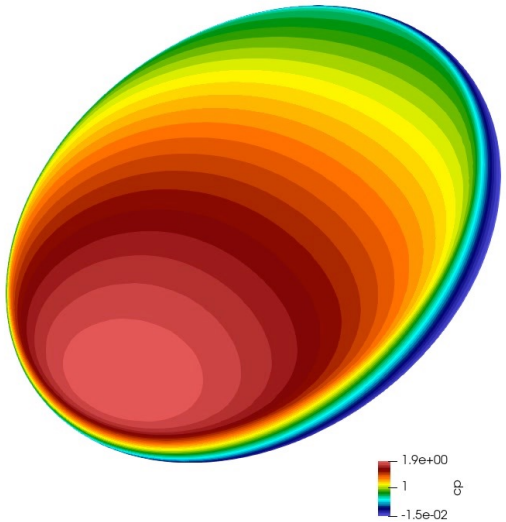


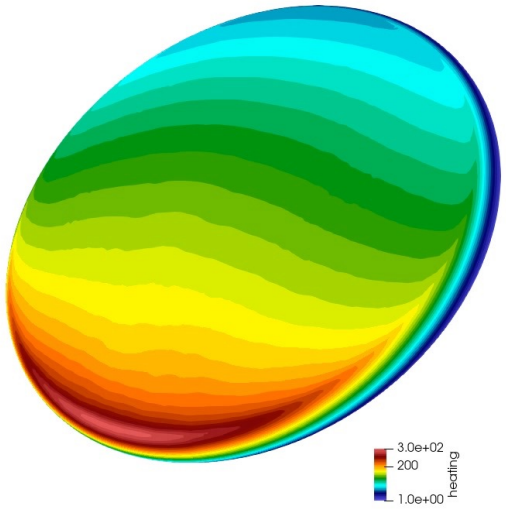


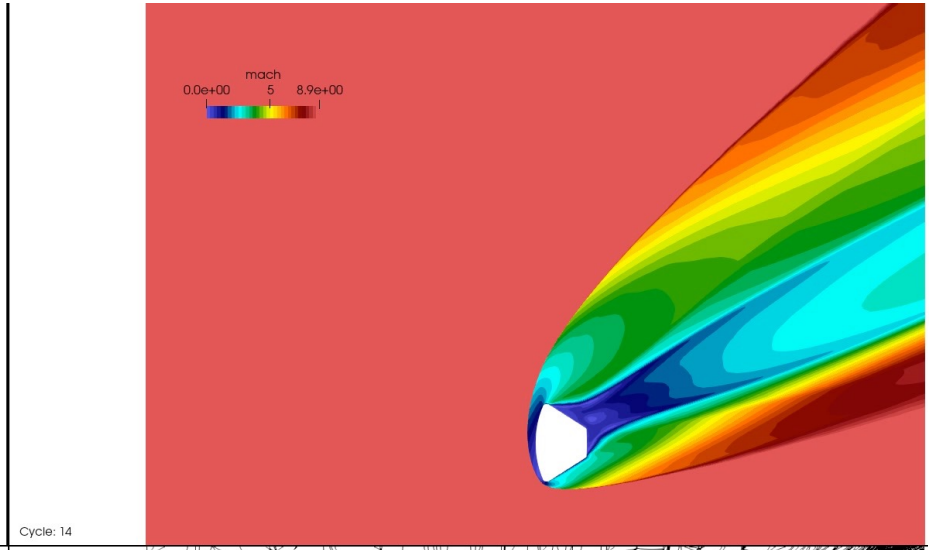


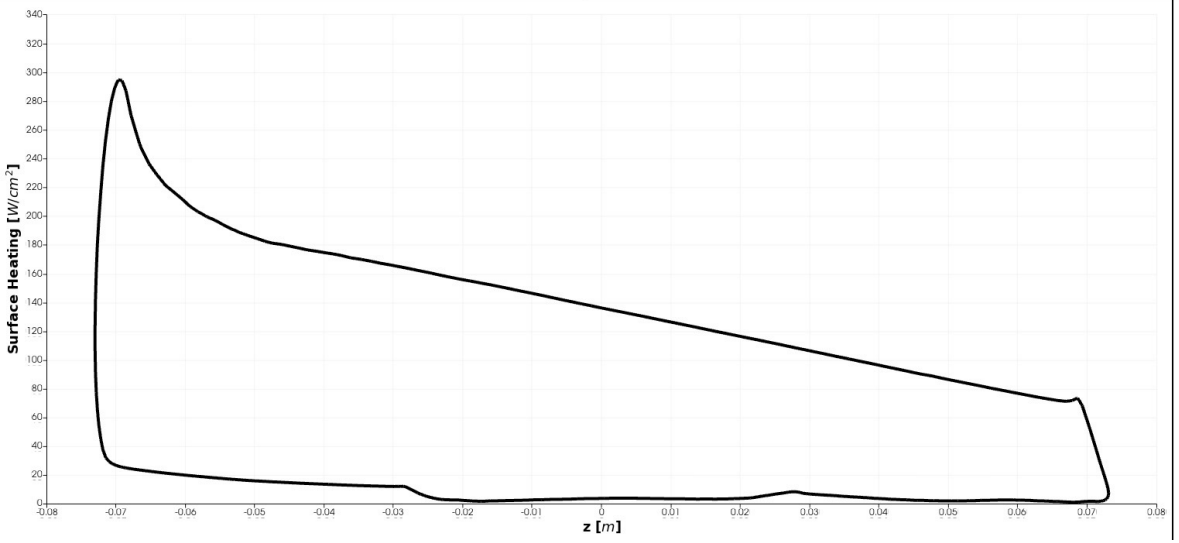


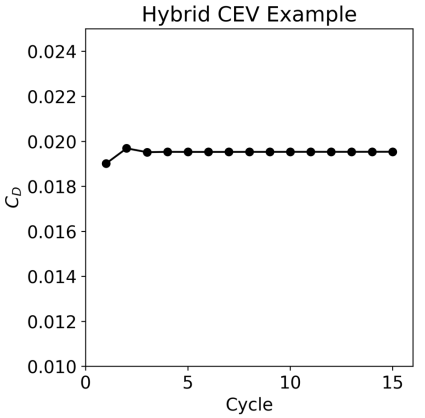
Grid Adaptation – CEV Hybrid Results



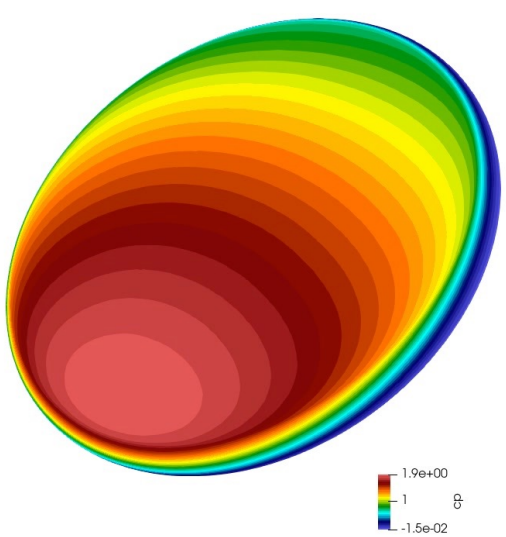


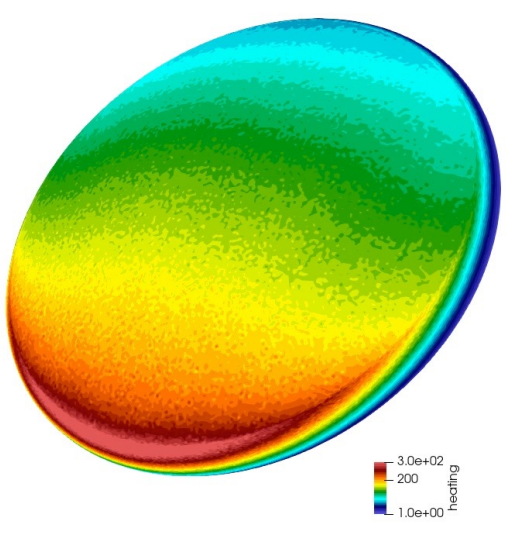


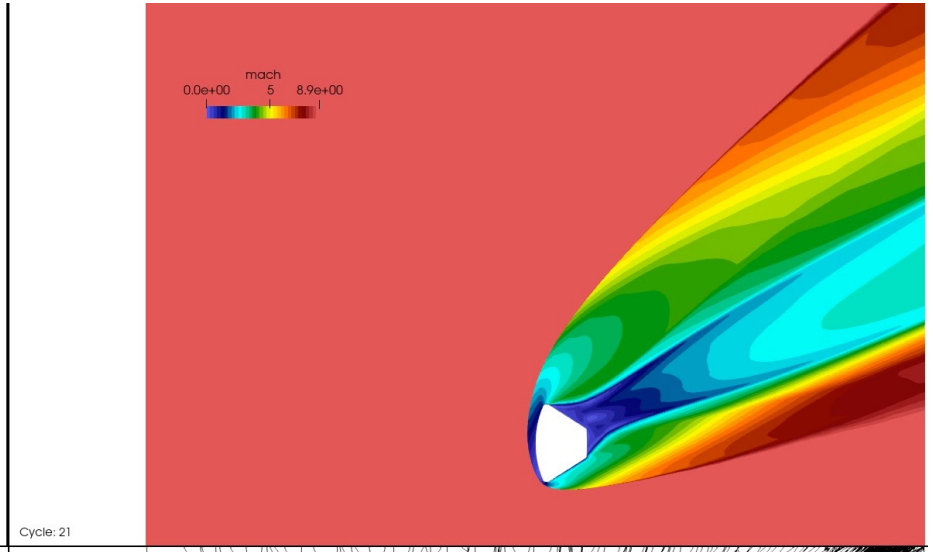


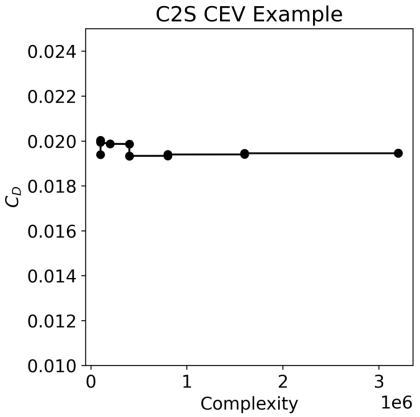


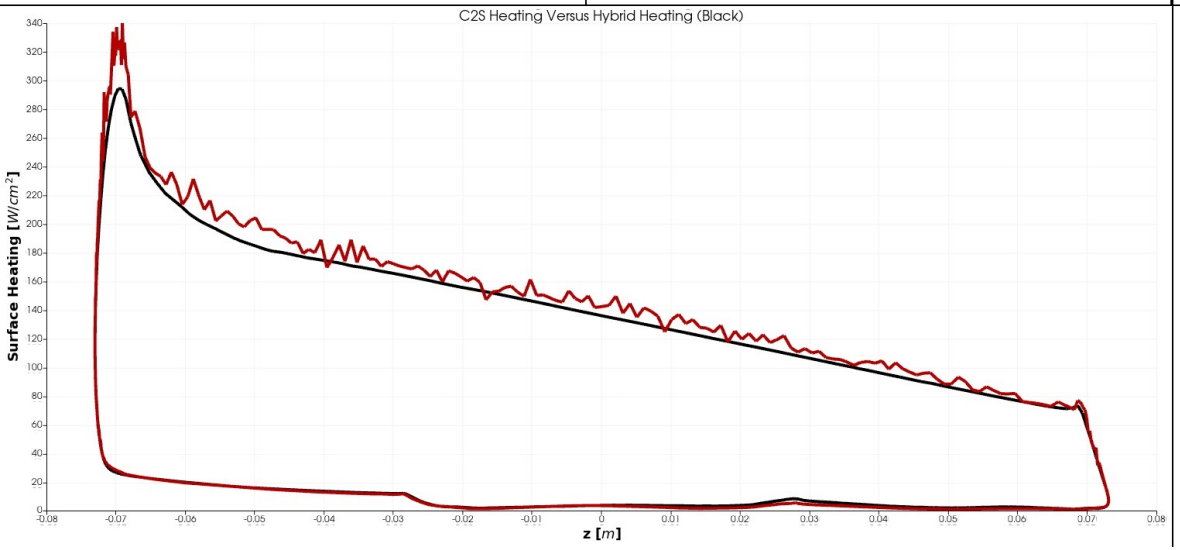
Grid Adaptation – CEV C2S Results

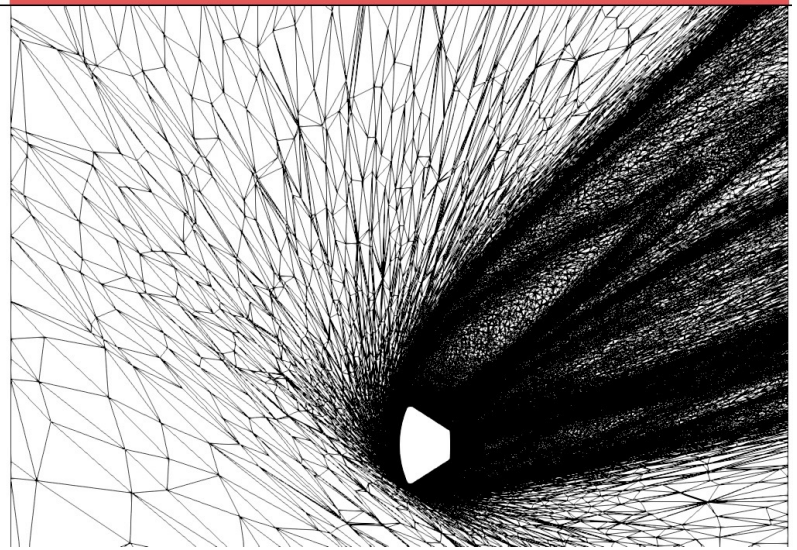












Where Things Can Go Wrong

- 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)
  - 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 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)
  - Reduce shock switch if you can run stably (turn off adaptive_shock_sensor and potentially adjust shock coefficients, see HLLE++ slide and **Ref. 3**)
- Problem diverges early on
  - Reduce CFL (minimum 2.0) and increase schedule and first order iterations
  - If using EBV, try default CBV approach (see viscous flux slide)
- 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

Summary

- Relevant known issues for 14.0
- Brief overview of generic gas path
- Recent improvements
- Solution strategies for blunt body and engine flows
- Hypersonic flow over a cylinder example
- Grid adaptation with and without CAD
- Hypersonic flow over a capsule examples
- References

References

1. GPU Implementation

- Paper: https://ntrs.nasa.gov/citations/20205010075
- Presentation: https://ntrs.nasa.gov/citations/20205011236

2. 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

3. Hybrid grid adaptation and HLLE++

- Paper: https://ntrs.nasa.gov/citations/20210023222
- Presentation: https://ntrs.nasa.gov/citations/20210024972

4. Edge-based viscous method

- Paper: https://ntrs.nasa.gov/citations/20220005528
- Presentation: https://ntrs.nasa.gov/citations/20220006376

5. Mars Retropropulsion campaign

- Paper: https://ntrs.nasa.gov/citations/20210024958
- Presentation: https://ntrs.nasa.gov/citations/20220002950

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