FUN3D v14.0 Training
Stabilized Finite Elements

W. Kyle Anderson[1],
Stephen L. Wood[1],
Kevin E. Jacobson[2],
Emmett Padway[2]

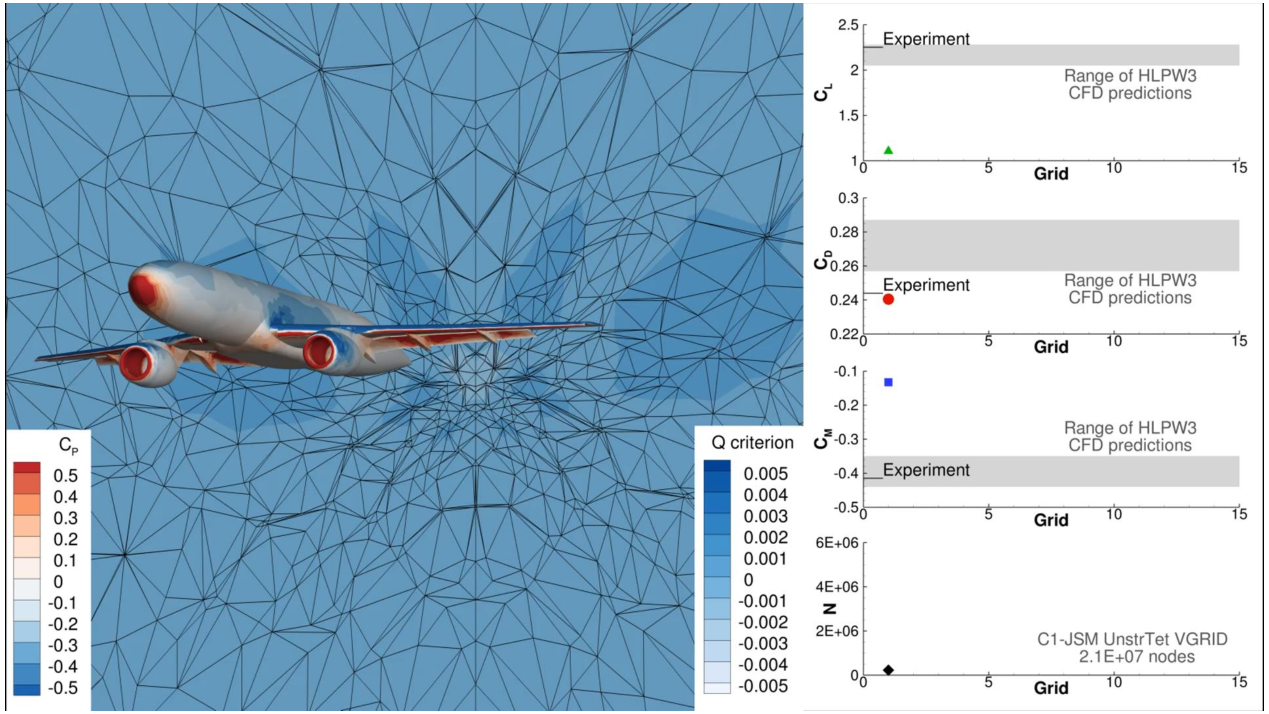
1: Computational AeroSciences Branch
2: Aeroelasticity Branch
NASA Langley Research Center

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

Spring, 2023

Overview

- Why add a Stabilized Finite-Element (SFE) Solver?
- Training scope
- Compilation optimization flags for SFE
- Shared components for SFE and Finite-Volume (FV) in FUN3D
- Steady analysis with SFE
  - Supported modes
  - Nonlinear iteration
  - Input files
  - Output files
- Trouble shooting
- Tutorial cases
  - BSCW
  - ONERA M6 with goal-oriented adaptation



Why add a Stabilized Finite-Element Solver?

- Smaller stencil for linearizations needed for adjoints and strong solvers
- Improved accuracy on tetrahedral meshes
- A path to higher order CFD
  - Everything available in FUN3D version 14.0 is 2nd order (P1 linear elements)
- Lower dissipation compared to the FV solver

Training Scope

- What this training will cover:
  - Compiling FUN3D with the SFE
  - How SFE is different than the standard FV solver
  - Case set up for SFE steady-state analysis
- What will not be covered:
  - Static aeroelastic analysis with SFE
  - Linearized Frequency Domain (LFD) analysis with SFE
- What you should already be familiar with:
  - Basic steady analysis with the FUN3D FV solver

Recommended Configure Options

```
../configure --enable-maxinlining=yes [other options]
```

Max inlining reduces execution time at the expense of compilation time by allowing additional optimizations across functions

Exemplar Intel compiler flags

```
FCFLAGS="-O3 -qopenmp -xCORE-AVX512"
CXXFLAGS="-O3 -qopenmp -xCORE-AVX512 -DL1P=1 -DL2P=1"
```

Optional if you want to do hybrid MPI+OpenMP

- Hybrid mode can be beneficial for the preconditioner in SFE

SFE makes effective use of vectorization

- These are exemplar Skylake optimization flags to enable vectorization

Shared components for SFE and FV in FUN3D

- FUN3D uses much of the same code to drive SFE as the FV solver
- Same nodet_mpi executable and *most* of the non-flow solver portions of FUN3D are compatible with SFE
  - Mesh partitioning
  - Mesh motion
    - SFE does not currently support overset simulations
  - Solution sampling
    - SFE solution is stored at the nodes like the FV solver
    - Derived quantities -- e.g., **`vort_mag`** -- are post processed using least-squares reconstruction (same as FV)

Steady analysis with SFE

Supported modes:

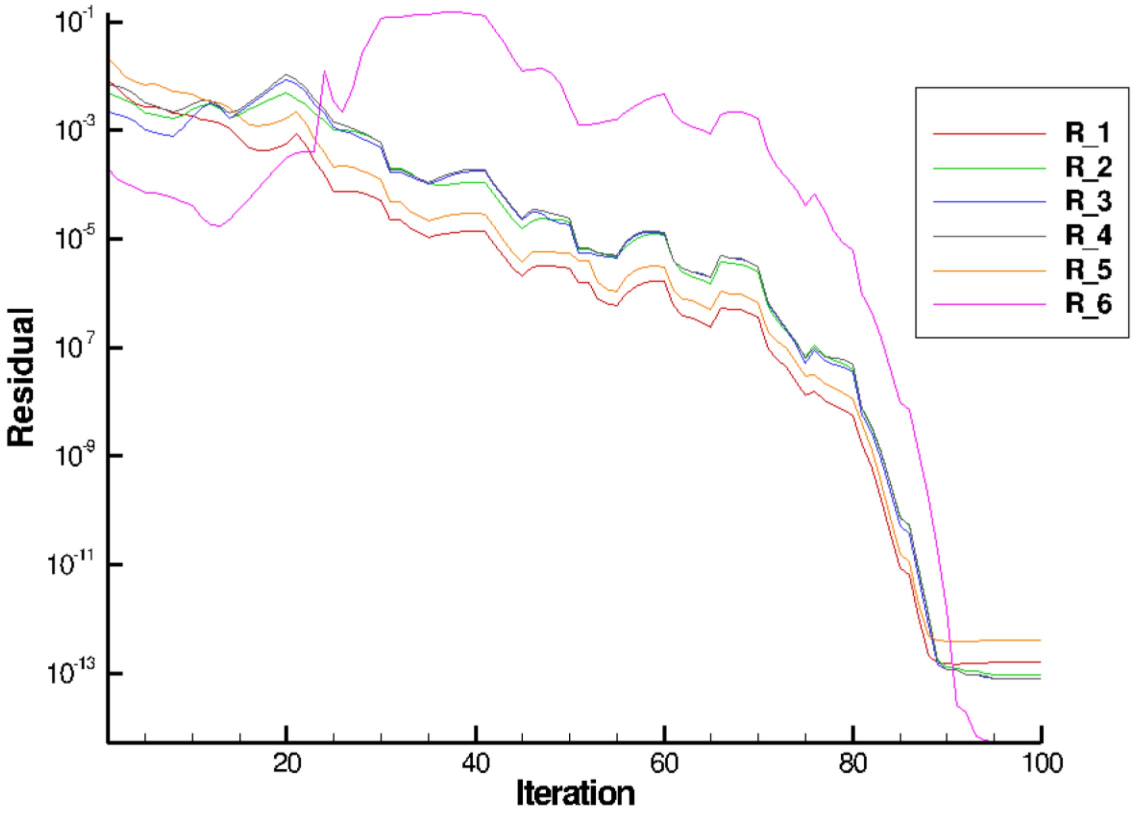
- CPU
- Compressible flow (perfect gas)
- Inviscid, Laminar, RANS (SA negative)
- Mixed-element grids
- 2D or 3D mode

Input files:

- **`fun3d.nml`** – options not specific to SFE
  - Mesh information, flow conditions, sampling options, number of steps, …
- **`sfe.cfg`** – options that only affect SFE
  - Shock smoothers, nonlinear controller parameters, …
  - The syntax is namelist like but is C++, so indexing starts at 0 for array inputs
- **`sfe_restart.cfg`** – automatically generated inputs for smooth restarting of SFE
  - Overrides **`sfe.cfg`** if this file is read

SFE steady convergence

- The Newton-based nonlinear solver in SFE will show behavior closer to HANIM than the point implicit solver typically used for the FV solver
  - Each iteration is slower, but fewer iteration are required
  - SFE typically converges in less than 300 steps rather than 10,000s

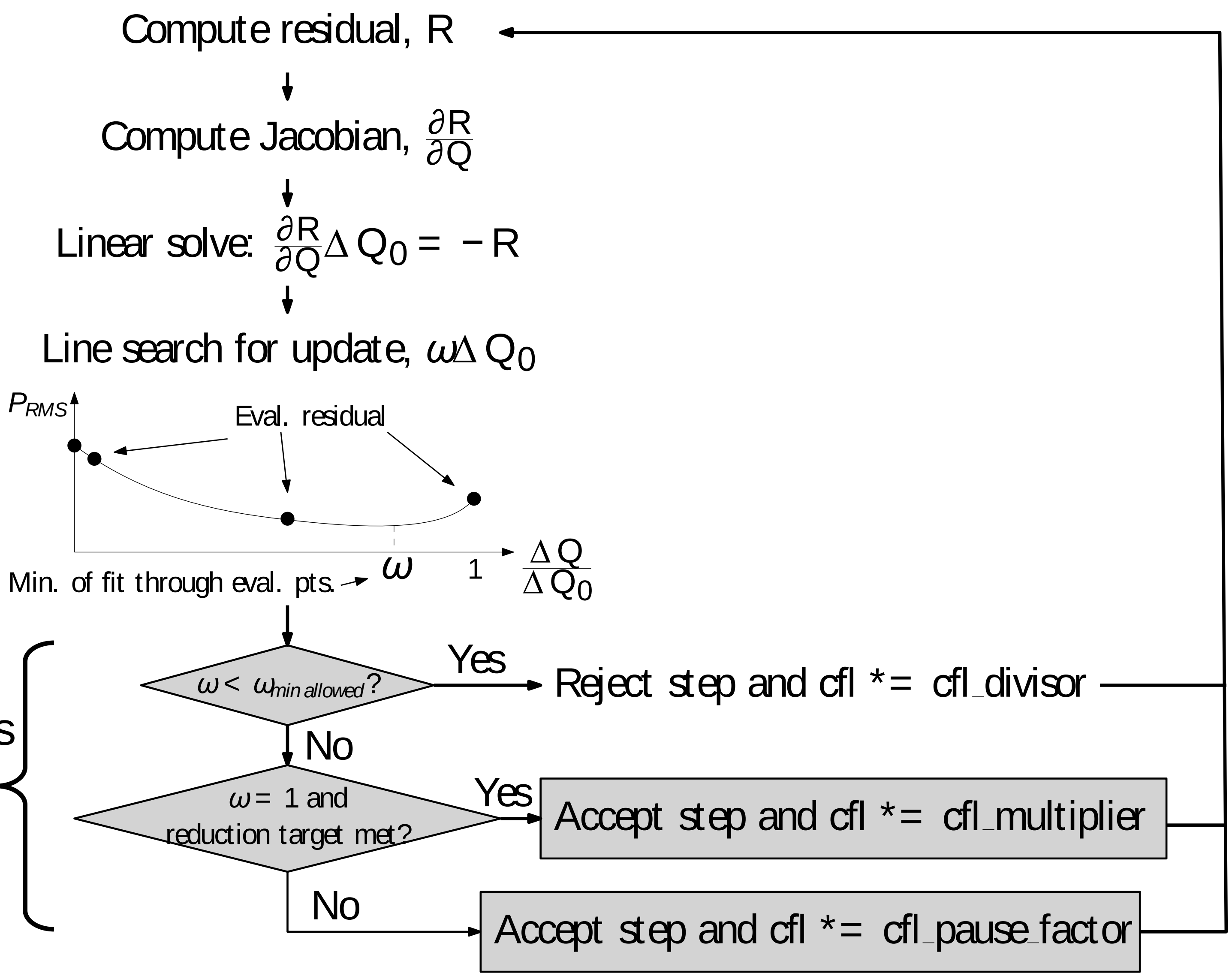


The SFE nonlinear iteration

Most of the compute time is spent here

L2 norm of residual

The actual controller decision conditions are more complicated to account for things like realizability constraints



fun3d.nml inputs for SFE

- To use SFE, you must set the **`flow_solver`** in the fun3d.nml:

```
&governing_equations
    flow_solver = 'sfe'
/
```

- Options for the FV discretization or algorithm settings do <u>not</u> affect SFE, e.g.

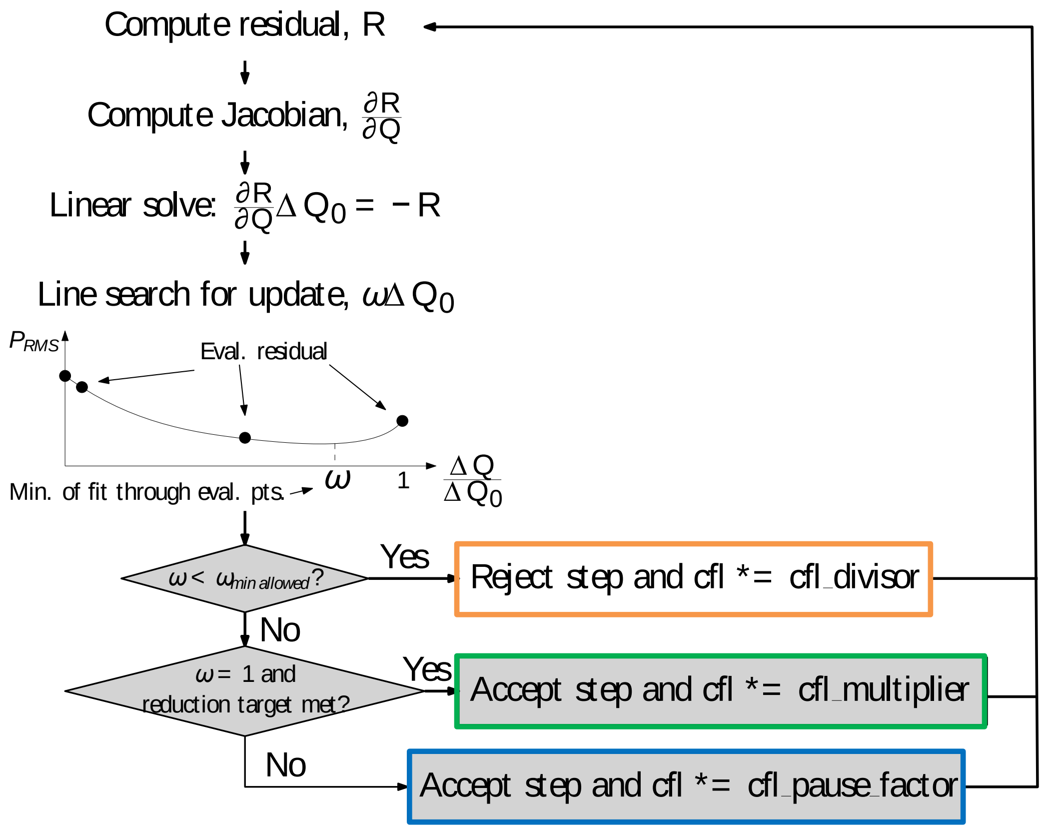
```
&inviscid_flux_method, &linear_solver_parameters, ...
```

- Typical options used in SFE runs:
  - **`&project, &raw_grid,`** sampling namelists
  - **`&code_run_control – steps, restart_read`**
  - **`&governing_equations – viscous_terms`**
  - **`&reference_physical_properties – mach_number, reynolds_number, temperature, angle_of_attack`**

sfe.cfg :: nonlinear controller parameters

- `cfl_init = 1.0` – initial CFL number
- `cfl_min = 0.1` – minimum allowed CFL number
- `cfl_max = 1.0e6` – maximum allowed CFL number
- `cfl_divisor = 0.1`
- `cfl_multiplier = 1.25`
- `cfl_pause_factor = 0.8`

- More aggressive options for easier cases:
  – `cfl_divisor = 0.1`
  – `cfl_multiplier = 2.0`
  – `cfl_pause_factor = 1.0`



sfe.cfg :: smoothers 1 of 4

Smoothers locally add dissipation by augmenting the local viscosity to capture shocks, expansions, etc.

- Set **`smoothing=.true.`** for cases where the flow may go supersonic
- SFE has several smoother options that have different sensors used to detect where to add dissipation

Multiple smoothers can be turned on in the same simulation

- Set **`number_of_smoothers = 2`**
- Use the indices of **`smoother_type, smoother_clip,`** etc. to control the individual smoothers. <u>Reminder: array inputs start at 0 in sfe.cfg.</u>

Controlling the smoothers:

- **`smoother_clip(i) = 2.0`** – threshold used for the i^{th} smoother to selectively apply smoother in the field
  - lower values -> increased area of application and higher magnitude (local effect)
- **`smoother_coef(i) = 1.0`** – i^{th} smoother's scaling coefficient (global effect)
  - higher values -> more smoothing applied

sfe.cfg :: smoothers 2 of 4

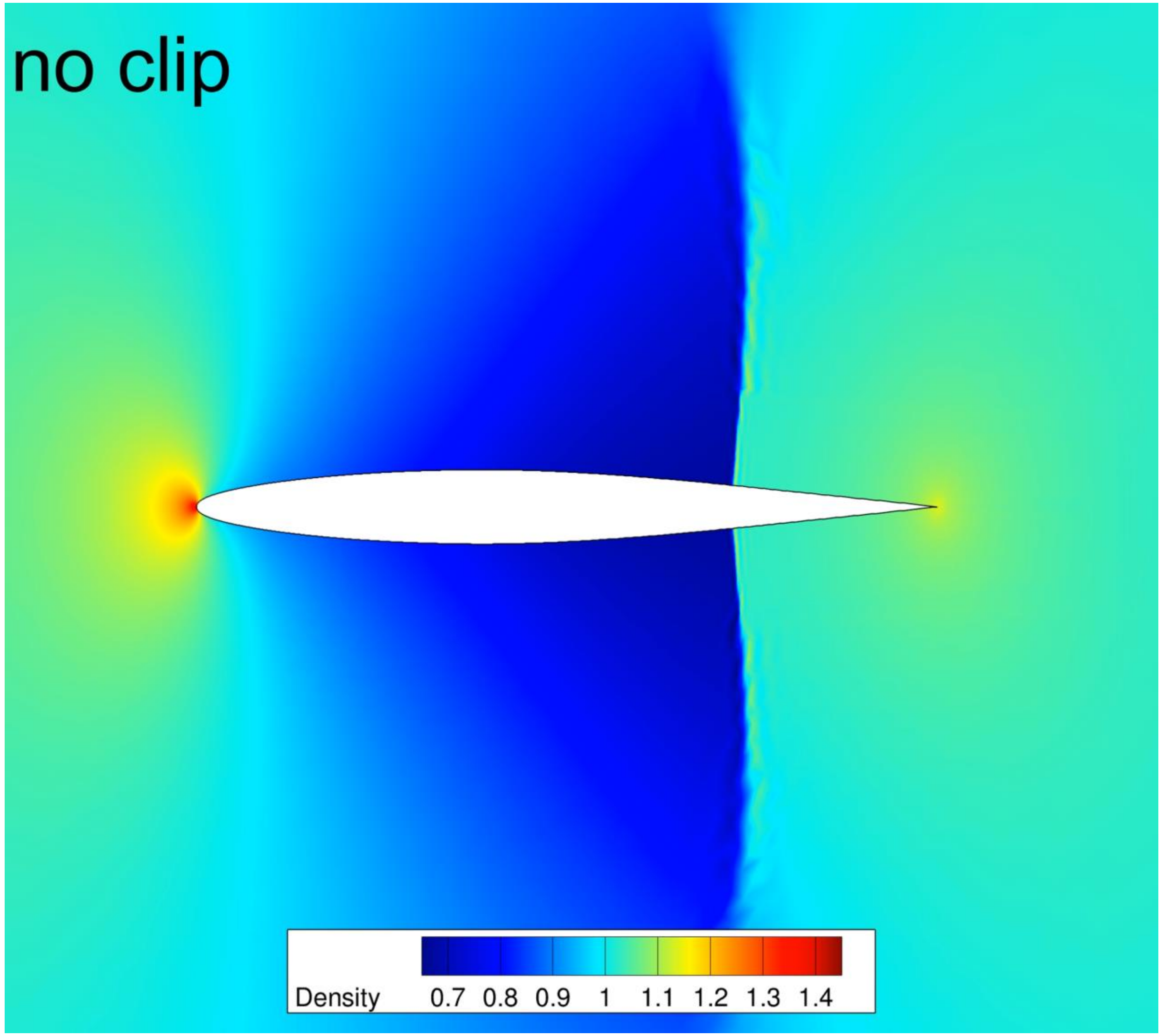
`smoother_type(i) = metric_pressure`
the default smoother which applies smoothing at shocks and expansions

- **`smoother_clip(i) = 2.0`**

  Typically use between 1.0 – 4.0,

  but we have used as low as 0.5

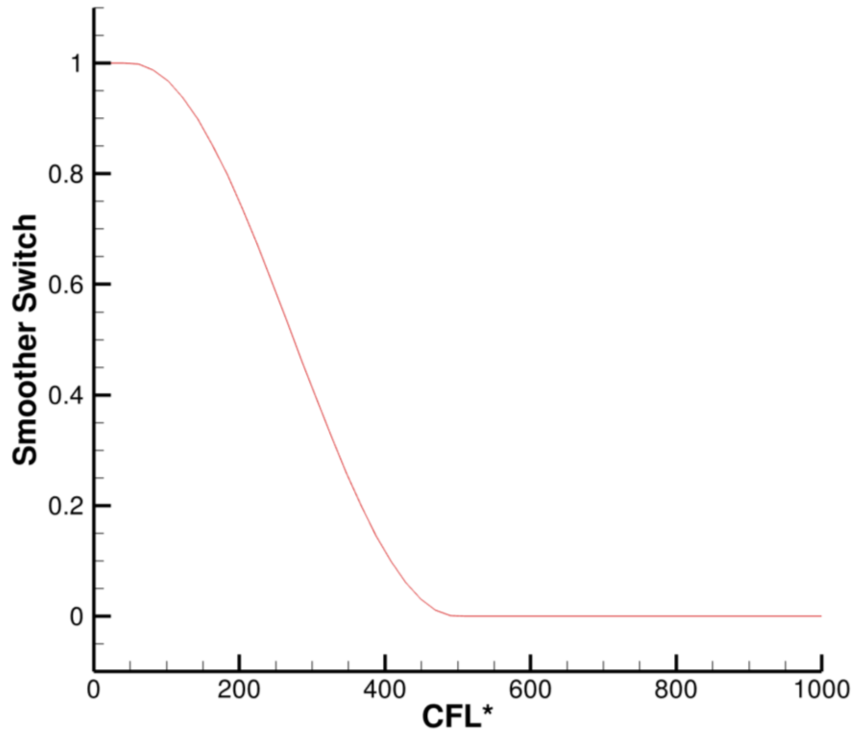
- **`smoother_coef(i) = 1.0`**
  Typically leave at 1.0



sfe.cfg :: smoothers 3 of 4

`smoother_type(i) = ramped` – Spatially uniform smoothing for domain that ramps away as CFL* (the maximum CFL) increases.

- Starting ramping down at CFL=50. Completely off at CFL=500
- Useful for helping cases that have difficulty starting
- **`smoother_coef(i) = 1.0`** – Sometimes need to increase to 10 for tough cases



sfe.cfg :: smoothers 4 of 4

Recommendations for smoothers:

- If running low subsonic, you can leave on the default, **`smoothing = .false.`**
- For anything above low subsonic, **`smoothing = .true.`**
  - Defaults to one active smoother with **`smoother_type(0) = metric_pressure`**
- For high-speed cases that have a hard time starting, add a second smoother with ramped dissipation (the metric_pressure will be the 0^{th} smoother) :

```
smoothing = .true.
number_of_smoothers = 2
smoother_type(1) = ramped
smoother_coef(1) = 1.0 ! If it still struggles to start, try 10.0
```

sfe.cfg :: linear solver parameters 1 of 3

`max_matvec = 600` – total number of linear search directions

`krylov_dimension = 300` – number of orthogonal search directions

- each new Krylov vector takes more time and more memory to store

`level_of_fill = 2` – size of the fill in for the Incomplete LU preconditioner

- More fill will typically yield a more accurate approximate inverse but requires more memory

Linear solver tolerances

`relative_linear_residual_tolerance = 1e-8`

`absolute_linear_residual_tolerance = 1e-15`

Note: the default parameters are recommended for steady simulations. For adjoint and LFD these parameters often need to be adjusted.

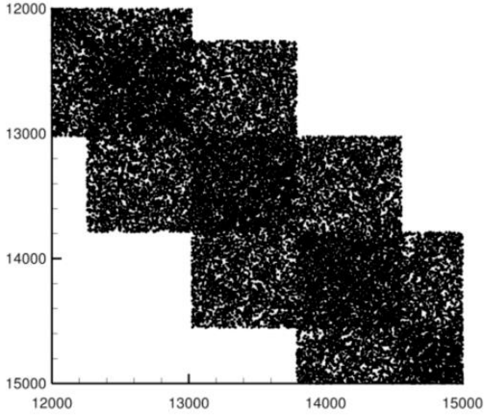
sfe.cfg :: linear solver parameters 2 of 3

Reordering

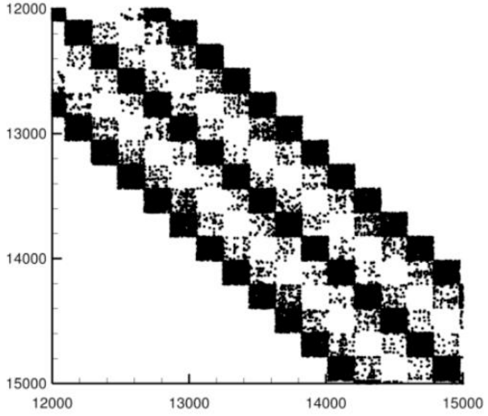
- **`reorder = k-ordering`** – default node reordering algorithm
  - **`reorder = cmk`** – Cuthill McKee algorithm
- **`reverse = .false.`** – Reverse the ordering algorithm

Q ordering

- Add localized randomization to the ordering as a second step applied after the initial reordering to improve stability of linear solver. Recommended for difficult linear problems (transonic LFD).
- **`q_ordering = 0`** – off (default), **`q_ordering = 1`** – turn on q-ordering
- **`prune_width = 12.0`** – factor that controls the number of rows involved in the local randomization
  - Smaller **`prune_width`** can be more stable but slower due to larger final matrix bandwidth



`prune_width = 1`



`prune_width = 4`

sfe.cfg :: linear solver parameters 3 of 3

Dynamic reordering

- SFE can use a couple of indicators to dynamically adjust q-ordering if difficult linear problems are encountered during a simulation
- **`dynamic_reordering = 2`** – selects the indicator to trigger reordering
  - **`0`** – off
  - **`1`** – reorder before linear solve if growth trigger exceeded
  - **`2`** – reorder if linear solve did not reach residual reduction target and the growth trigger is exceeded
- **`dynamic_reordering_growth_trigger = 1.0e10`** – the threshold of acceptable growth in the L2 norm of the first Krylov vector due to the application of the preconditioner
- **`dynamic_reordering_prune_factor = 0.75`** – multiplicative adjustment to the size of the groups of rows and columns in the matrix that are reordered when
  - typical values are 0.5 and 0.75
- **`dynamic_reordering_min_prune_width = 1.0e-6`** – minimum allowed prune width

sfe.cfg :: other inputs 1 of 2

Residual smoothing – Locally add dissipation based on change in state during line search

- `residual_smoothing = .true.`
  - Turn on/off residual smoothing
- `residual_smoothing_coefficient = 10.0`
  `residual_smoothing_secondary_coefficient = 50.0`
  - Amount of smoothing to apply. Larger values -> more dissipation (magnitude)
- `residual_smoothing_switch_interval = 5`
  - Alternate between using the primary and secondary coefficient every {**`interval`**} steps

Round-off termination - trigger SFE termination when solution is not changing anymore but still above specified **`stopping_tolerance`**

- `round_off_termination = 1`
- `round_off_tolerance = 1e-12`
- Stop SFE taking steps when $\frac{RMS(\Delta q)}{RMS(q)} <$ **`round_off_tolerance`**

sfe.cfg :: other inputs 2 of 2

Weak boundary condition for viscous walls (4000)

- **`weak_bc = 2`** – (default) penalty-based weak boundary condition
  - Velocities at surfaces will be small, but not exactly zero at convergence
- **`weak_bc = 0`** – strong enforcement of no slip condition

SA-QCR2000 Quadratic Constitutive Relation – not read from fun3d.nml

- **`qcr = .true.`**

SFE outputs

- Typical FUN3D steady outputs will be based on SFE's calculations:
  - `{project}_hist.dat` – residual history
  - `{project}.flow` – restart state
- `Standard (screen) output` – minimal amount of output for monitoring a simulation
- `{project}_sfe.out` – detailed version of SFE monitoring output
- `{project}_sfe_hist.dat` – Tecplot file of SFE residuals, forces and moments, CFL number, etc.
- `sfe_restart.cfg` – inputs that can be used to restart a simulation
- `{project}_flow.921`, `{project}_flow.cfl` – details of nonlinear controller and line search
  - Useful information to send to us for user support cases, but otherwise can ignore

SFE standard (screen) output for an iteration 1 of 2

```
Iter    9  CL = -3.192930242e-04  CD =  6.357647136e-01  CMy = -1.212308508e-02
  maxMach =  7.4001062e-01   minTemp =  9.9986614e-01  minDens =  9.9877269e-01
  res =  1.3462e-04  5.0573e-03  5.3034e-06  1.2820e-04  1.0171e-03  2.1076e-05
  linear matvecs =      134   final res =    8.53029e-08   rate =    9.59122e-09
  CFL =  8.00000e-01  line search step size = 0.98985
```

SFE standard (screen) output for an iteration 2 of 2

```
Iter    9  CL = -3.192930242e-04  CD =  6.357647136e-01  CMy = -1.212308508e-02
 maxMach =  7.4001062e-01   minTemp =  9.9986614e-01  minDens =  9.9877269e-01
 res =  1.3462e-04  5.0573e-03  5.3034e-06  1.2820e-04  1.0171e-03  2.1076e-05
 linear matvecs =      134   final res =    8.53029e-08   rate =    9.59122e-09
 CFL =  8.00000e-01  line search step size = 0.98985
```

- Current loads
- Extrema of state
- Current nonlinear residuals
- Linear solver convergence – iterations (matvecs), final linear problem residual and convergence rate
- Nonlinear solver - line search step size (ω) and updated CFL number

{project}_sfe.out – full iteration output 1 of 2

```
    9 CFL_star =    1.0000000000e+00  scale_factor =    1.0000000000e+00
 Wall clock time for residual =    7.9765624400e-01
    9 rms = [    2.2085858132e+02    1.5501516914e+03    5.9804670003e+03    2.1073767969e-03    1.1406551141e-01    1.1191056680e+01    1.1191056680e+01 ] cfl =            1.00000
   2 max residual and location =    4.4363810401e-04   1.6933129977e+00   3.2853693322e+01   9.6951930984e-02 global id = 100988 slen = 8.8555181762e-05 rank =     0
Pressure forces from inviscid surface bc =    0.0000000000e+00    0.0000000000e+00    0.0000000000e+00
Pressure forces from viscous surface bc  =    1.5194650723e+02   -1.1304305391e+00    1.2179625807e-01
Viscous forces from viscous surface bc  =    1.7356502616e+02    2.6011552552e-01   -2.8527428644e-01
Total forces from viscous surface bc     =    3.2551153339e+02   -8.7031501356e-01   -1.6347802837e-01
Total forces from all surfaces           =    3.2551153339e+02   -8.7031501356e-01   -1.6347802837e-01
CL =   -3.1929302417e-04 CD =    6.3576471365e-01 CLp =    2.3788331653e-04 CDp =    2.9677052193e-01 CLv =    -5.5717634070e-04 CDv =    3.3899419172e-01
CMx =   -2.2346581292e-03 CMxp =   -1.7799280073e-03 CMxv =   -4.5473012190e-04
CMy =   -1.2123085084e-02 CMyp =   -1.3899538465e-02 CMyv =    1.7764533812e-03
CMz =   -1.0421167260e+01 CMzp =   -4.7730402476e+00 CMzv =   -5.6481270120e+00
 currentIteration =       9     1.3461895268e-04     5.0573370660e-03     5.3034007850e-06     1.2820428491e-04     1.0171197936e-03     2.1075684098e-05
 Wall clock time for left-hand side via operator overloaded operations using expression templates =    1.6265510181e+01
    9 Number of zero or negative diagonals  =        0        0        0        0        0        0
    9 max preconditioner application growth =     6.0663693509e+00 rank =    44
Search direction       1 residual =    8.8938541522e+00 rate =    1.0000000000e+00
Search direction      10 residual =    6.2860759119e-01 rate =    7.0678873346e-02
Search direction      20 residual =    1.1561351602e-01 rate =    1.2999259269e-02
Search direction      30 residual =    2.4949710167e-02 rate =    2.8052753890e-03
Search direction      40 residual =    6.2341633675e-03 rate =    7.0095183267e-04
Search direction      50 residual =    1.7664334737e-03 rate =    1.9861282223e-04
Search direction      60 residual =    5.0761761823e-04 rate =    5.7075100349e-05
Search direction      70 residual =    1.5204205418e-04 rate =    1.7095181861e-05
Search direction      80 residual =    4.4990380781e-05 rate =    5.0585921481e-06
Search direction      90 residual =    1.3178913005e-05 rate =    1.4817999912e-06
Search direction     100 residual =    3.9102585218e-06 rate =    4.3965849393e-07
Search direction     110 residual =    1.1995190015e-06 rate =    1.3487055004e-07
Search direction     120 residual =    3.7157092235e-07 rate =    4.1778391684e-08
Search direction     130 residual =    1.3055244409e-07 rate =    1.4678950414e-08
   9 Final Search direction     134 residual =    8.5302882352e-08 rate =    9.5912166911e-09 actual residual =    8.5302882394e-08 actual rate =    9.5912166911e-09
 GMRES init wall clock time =    3.2581950000e-02
 GMRES core wall clock time =    3.8629861690e+00
 Preconditioner update wall clock time =    2.5410916340e+00
 Preconditioner application wall clock time =    1.0419944091e+01
 Matrix vector product wall clock time =    2.8636879170e+00
 Wall clock time for linear solve =    2.0008637394e+01
relax_r =    9.8985000860e-01
relax_t =    1.0000000000e+00
 Wall clock time for line search =    3.2880951660e+00
 rms_dq = 1.2061860971e-02 rms_q = 6.6653365484e-01 rms_dq/rms_q = 1.8096402010e-02
Current iteration CFL omega relaxSave relaxFit =    9   8.00000e-01   9.89850e-01   9.89850e-01   9.89850e-01
```

{project}_sfe.out – full iteration output 2 of 2

```
     Wall clock time for left-hand side via operator overloaded operations using expression templates =    1.6265510181e+01
    9 Number of zero or negative diagonals  =        0        0        0        0        0        0
     9 max preconditioner application growth =     6.0663693509e+00 rank =    44
Search direction       1 residual =    8.8938541522e+00 rate =    1.0000000000e+00
Search direction      10 residual =    6.2860759119e-01 rate =    7.0678873346e-02
Search direction      20 residual =    1.1561351602e-01 rate =    1.2999259269e-02
Search direction      30 residual =    2.4949710167e-02 rate =    2.8052753890e-03
Search direction      40 residual =    6.2341633675e-03 rate =    7.0095183267e-04
Search direction      50 residual =    1.7664334737e-03 rate =    1.9861282223e-04
Search direction      60 residual =    5.0761761823e-04 rate =    5.7075100349e-05
Search direction      70 residual =    1.5204205418e-04 rate =    1.7095181861e-05
Search direction      80 residual =    4.4990380781e-05 rate =    5.0585921481e-06
Search direction      90 residual =    1.3178913005e-05 rate =    1.4817999912e-06
Search direction     100 residual =    3.9102585218e-06 rate =    4.3965849393e-07
Search direction     110 residual =    1.1995190015e-06 rate =    1.3487055004e-07
Search direction     120 residual =    3.7157092235e-07 rate =    4.1778391684e-08
Search direction     130 residual =    1.3055244409e-07 rate =    1.4678950414e-08
    9 Final Search direction     134 residual =    8.5302882352e-08 rate =    9.5912166911e-09 actual residual =    8.5302882394e-08 actual rate =    9.5912166911e-09
 GMRES init wall clock time =    3.2581950000e-02
 GMRES core wall clock time =    3.8629861690e+00
 Preconditioner update wall clock time =    2.5410916340e+00
 Preconditioner application wall clock time =    1.0419944091e+01
 Matrix vector product wall clock time =    2.8636879170e+00
 Wall clock time for linear solve =    2.0008637394e+01
```

- Reported preconditioner application growth is used in dynamic reordering (see linear solver parameters)

Output forces

- There are two versions of the forces computed for SFE simulations
  - The `{project}_hist.dat` file will contain the FV integrated forces using the SFE state
  - The `{project}_sfe_hist.dat` file and screen output will contain the SFE integrated forces

- These values will be close for reasonably resolved meshes, but the SFE integrated forces should be considered more accurate

Cases that struggle to start

- SFE will stall when a step is rejected while **`cfl = cfl_min`**
- What to try if SFE stalls at the beginning of a simulation:
  - For subsonic cases, if SFE is reporting a max Mach number near 1 or small minimum density, turn on **`smoothing = .true.`**
  - Add **`ramped`** dissipation, see slides on smoothers
  - If the previous steps don't work, try turning off residual smoothing

Cases that struggle to converge

- For subsonic cases, if SFE is reporting a max Mach number near 1, turn on

  `smoothing = .true.`

- Check linear solver convergence. If consistently failing to converge the linear system, then adjust linear solver parameters (see linear solver parameter slides)
- Use a more dissipative smoother setting, **`smoother_clip(i) = 1.0`** for **`smoother_type(i) = metric_pressure`**

Recommendations for problem decomposition

- Forward problems:
  - 10,000 - 30,000 mesh points per processor
  - Memory required by solver: rough rule of thumb 2 GB per million points (not cells!)
  - MPI only or MPI + 2 OpenMP threads (with **`linear_solver = slat_fgmres`**, **`preconditioner = lsiluk`**) are recommended for robust performance
- Adjoint problems:
  - 10,000 - 30,000 mesh points per processor
  - Memory required by solver: rough rule of thumb 3 GB per million points (not cells!)
  - 1 MPI process per socket with OpenMP threads (with **`linear_solver = slat_fgmres`**, **`preconditioner = lsiluk`**) is recommended for robustness

Tutorial case: Steady BSCW 1 of 4

- The Benchmark Supercritical Wing (BSCW) is one of the Aeroelastic Prediction Workshop (AePW) cases.
- The tutorial cases here will step through Case 2 of the 2nd AePW: flutter prediction at Mach=0.74, AoA=0.0
  - Steady SFE analysis -> static aeroelastic SFE analysis -> LFD
  - Plunge and linearized pitch structural degrees of freedom
- Start by setting the flow conditions and solver selection in the fun3d.nml file:

```
&governing_equations
    eqn_type = "compressible"
    viscous_terms = "turbulent"
    prandtlnumber_molecular = 0.755
    flow_solver = "sfe"
 /
 &reference_physical_properties
    temperature_units = "Kelvin"
    mach_number =        0.74
    reynolds_number =  278125.0
    temperature =    304.911111
    angle_of_attack = 0.0
 /
```

Tutorial case: Steady BSCW 2 of 4

- Add the mesh information, steady solver parameters, sampling options to fun3d.nml:

```
&project
   project_rootname = "bscw_coarse_mixed_nc"
/
&raw_grid
   grid_format = "aflr3"
   patch_lumping = "family"
/
&nonlinear_solver_parameters
   time_accuracy = "steady"
/
&code_run_control
   steps = 200
   stopping_tolerance =   1.0E-15
   restart_read = "off"
/
&global
   boundary_animation_freq = -1
/
&boundary_output_variables
    number_of_boundaries = 2
    boundary_list = '1,3'
    mach                = .true.
    cp                  = .true.
    turb1               = .true.
    temperature         = .true.
/
```

Tutorial case: Steady BSCW 3 of 4

- Create the sfe.cfg file:
  - Turn on the default shock smoother because local flow will go supersonic
  - Add the uniform ramped smoother to help start the nonlinear solver

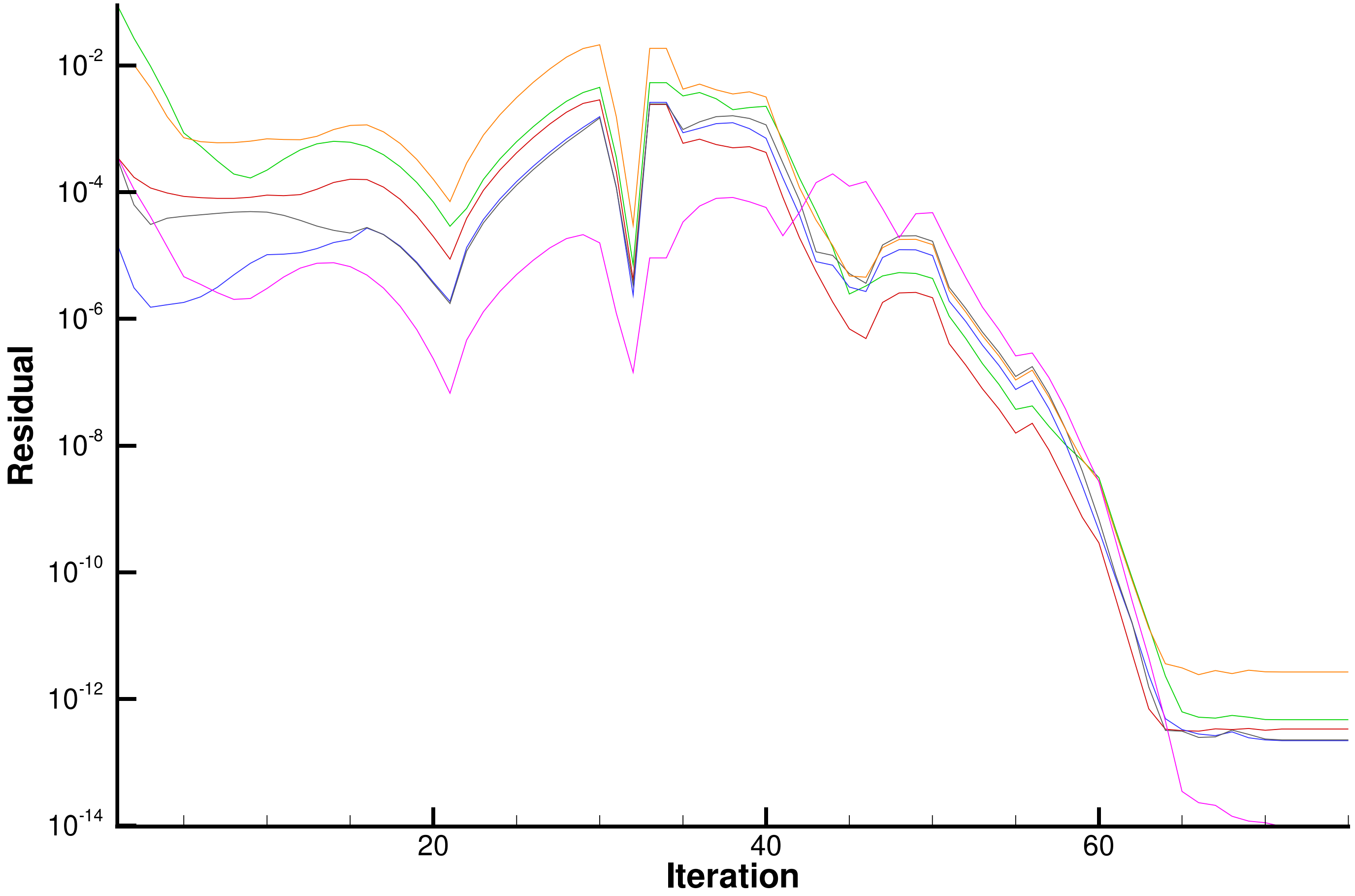
```
smoothing = .true.
number_of_smoothers = 2
smoother_type(1) = ramped
```

- Run the fun3d executable:
  - `mpirun nodet_mpi --gamma 1.136`

Langley Research Center

Tutorial case: Steady BSCW 4 of 4

Convergence:



Forces:

- SFE integrated: CL = **1.890**4727890e-01 CD = **1.4**203887901e-02
- FV integrated: CL = **1.890**2858800e-01 CD = **1.4**081163161e-02

Adjoint Overview 1 of 2

- SFE’s adjoint path is not run through dual_mpi rather, 1 iteration of nodet_mpi with

fun3d.nml

```
&code_run_control
    steps = 1
    stopping_tolerance =   1.0E-15
    restart_read = "on"
 /
```

sfe.cfg options

- `adjoint = .true.` to enable the adjoint mode
- `cost_function = 8`
    - output function of interest for the adjoint
        - '1,2,3' Cx,Cy,Cz force coefficients
        - '4,5,6' CMx,CMy,CMz, moments coefficients
        - '7,8' CL, CD
- `use_far_field_forces = .false.`
    - Toggles use of far-field boundaries to calculate forces

Adjoint Overview 2 of 2

- SFE's adjoint path is not run through dual_mpi rather, 1 iteration of nodet_mpi
  - SFE's adjoint solution cannot be visualized through the fun3d.nml sampling namelist
  - SFE writes state (primal) and costate (dual) solutions to prim_dual.solb
  - To visualize the adjoint solution
    - ref visualize {mesh file} prim_dual.solb prim_dual.tec
    - Produces Tecplot output with state (primal) and costate (dual) fields
      - Inviscid and laminar cases: state = [V1-V5], costate = [V6-V10]
      - Turbulent cases: state = [V1-V6], costate = [V7-V12]
- Recommendations
  - MPI+OpenMP with 1 MPI rank per socket
  - `linear_solver = slat_fgmres`
  - `preconditioner = lsiluk`
  - `relative_linear_residual_tolerance = 1e-14`
  - `absolute_linear_residual_tolerance = 1e-15`
  - If convergence is not satisfactory, reduce `prune_width` by 50%

Tutorial case: Steady ONERA M6 Adaptation 1 of 7

- The ONERA M6 wing validation case from the Turbulence Modeling Resource
  - https://turbmodels.larc.nasa.gov/ONERAwingnumerics_val.html
  - Mach = 0.84, Re = $14.6\text{x}10^6$, AoA = 3.06
- Workflow automated by Pyrefine:
  - Steady SFE analysis -> Adjoint SFE analysis -> Goal-oriented metric -> mesh adaptation
  - https://github.com/nasa/pyrefine/tree/main/examples/ONERA_m6/steady_sa_sfe_goal
  - Files:
    - adapt.py
    - fun3d.nml_forward
    - sfe.cfg_forward
    - fun3d.nml_adjoint
    - sfe.cfg_adjoint
- See Adaptation Tutorial for more details

Tutorial case: Steady ONERA M6 Adaptation 2 of 7

- adapt.py

```
from pyrefine import AdaptationDriver
from pyrefine.refine import RefineGoalOriented
from pyrefine.simulation import SimulationFun3dSFEAdjoint
from pbs4py import PBS

project = "om6ste"

pbs = PBS.k4(time=4)
pbs.mpiexec = 'mpiexec_mpt’

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()
```

Tutorial case: Steady ONERA M6 Adaptation 3 of 7

- Case files

fun3d.nml_forward

```
&code_run_control
  steps              = 400
  stopping_tolerance = 1.0e-11
  restart_read       = 'off'
  use_openmp = .true.
  grid_coloring = .true.
/
```

fun3d.nml_adjoint

```
&code_run_control
  steps              = 1
  stopping_tolerance = 1.0e-12
  restart_read       = 'on'
  use_openmp = .true.
  grid_coloring = .true.
/
```

Tutorial case: Steady ONERA M6 Adaptation 4 of 7

- Case files

sfe.cfg_forward

```
smoothing = .true.
weakBC = 0

linear_solver = slat_fgmres
preconditioner = lsiluk
--------------------------------------
--------------------------------------
--------------------------------------
--------------------------------------
--------------------------------------
```

sfe.cfg_adjoint

```
smoothing = .true.
weakBC = 0

linear_solver = slat_fgmres
preconditioner = lsiluk
relative_linear_residual_tolerance = 1e-14
absolute_linear_residual_tolerance = 1e-15

adjoint = .true.
cost_function = 8 ! CD
```

Tutorial case: Steady ONERA M6 Adaptation 5 of 7

- Pyrefine output

Begin adaptation step 11
Complexity: 30000.0
Running the flow forward solver
2929257.pbssrv2
Running the flow adjoint solver
2929275.pbssrv2
Complexity: 30000.0
Running goal-oriented refine
2929277.pbssrv2
Begin adaptation step 12

Tutorial case: Steady ONERA M6 Adaptation 6 of 7

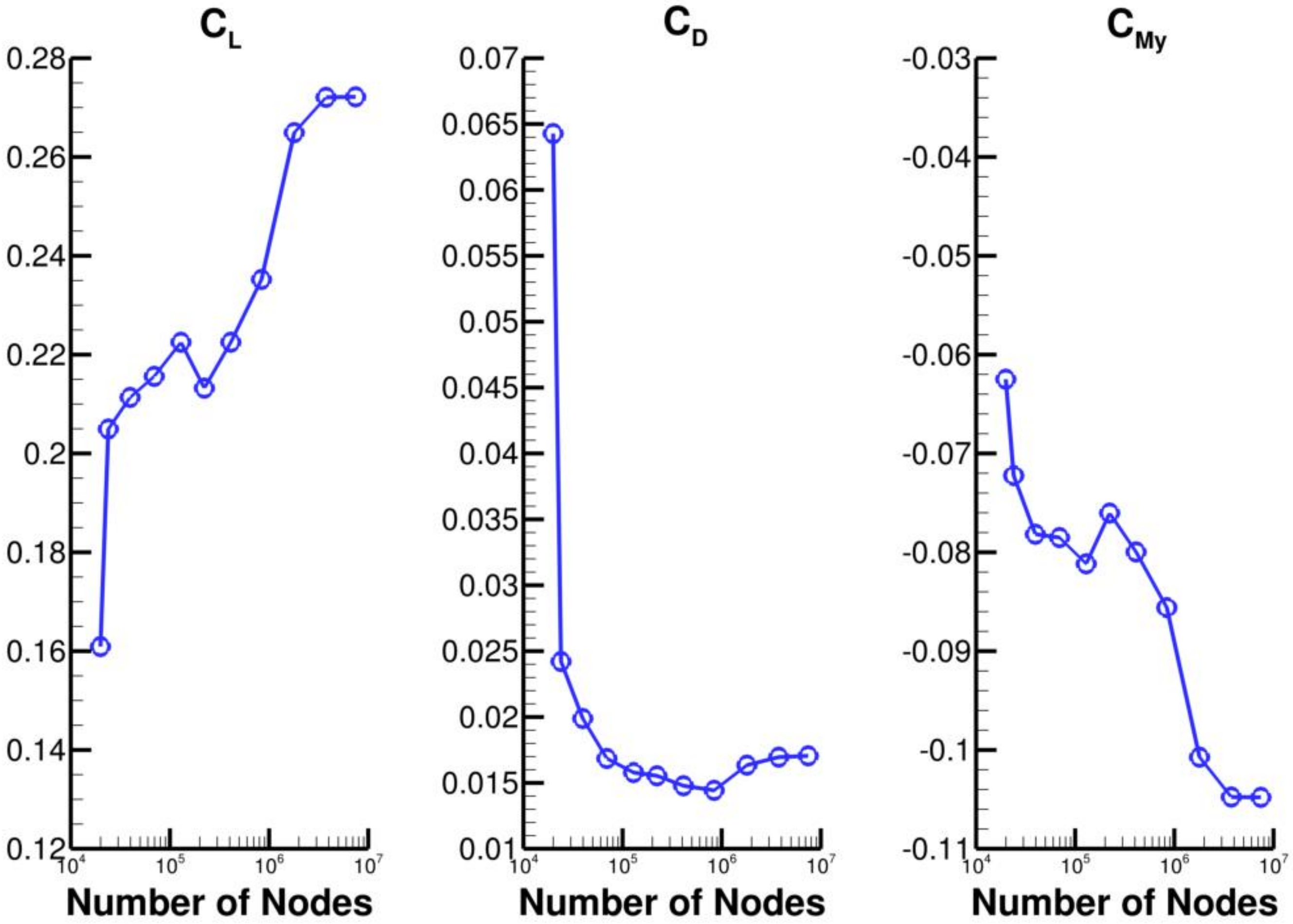
- SFE Adjoint output

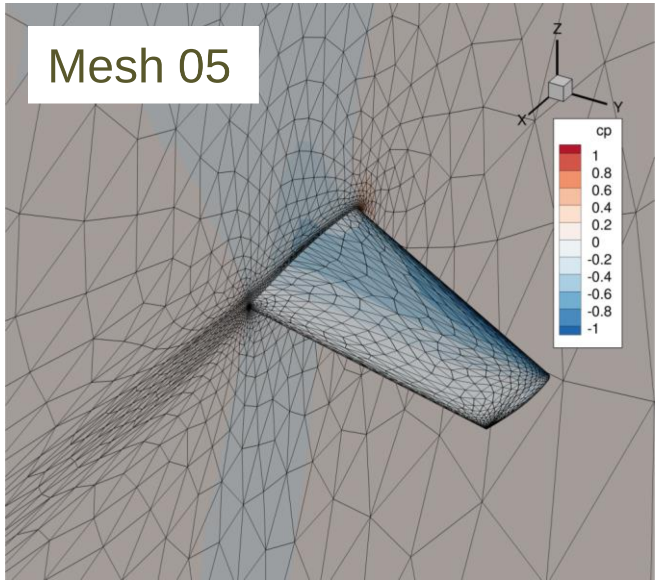
```
 linear matvecs =   1e+02   final res =    1.90769e-15   rate =    1.00708e-13
Adjoint derivative, Mach      =    1.3875931560e-01
Adjoint derivative, Reynolds =   -2.0016400716e-11
Adjoint derivative, AOA      =    7.0046118701e-03
Adjoint derivative, Yaw      =   -1.2069942780e-03
Writing prim_dual.solb ...  complete.
Writing prim_dual_rhs.solb ...  complete.
Writing sfe_restart.cfg ...  complete.

 Writing boundary output: om6ste11_tec_boundary.szplt
  Time step: 43, ntt: 1, Prior iterations: 42
 Writing INRIA solb volume file='om6ste11_volume.solb'
 No restart files written!
 Done.
```

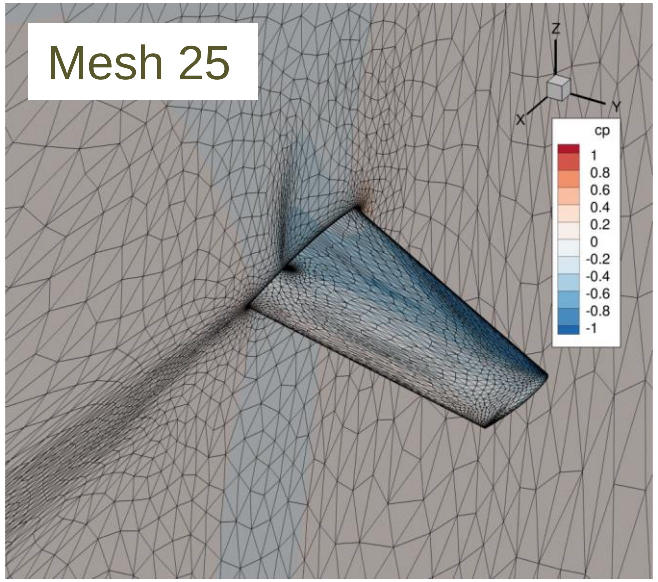
Tutorial case: Steady ONERA M6 Adaptation 7 of 7

- Post processing with pyrefine

  https://nasa.github.io/pyrefine/post_processing.html



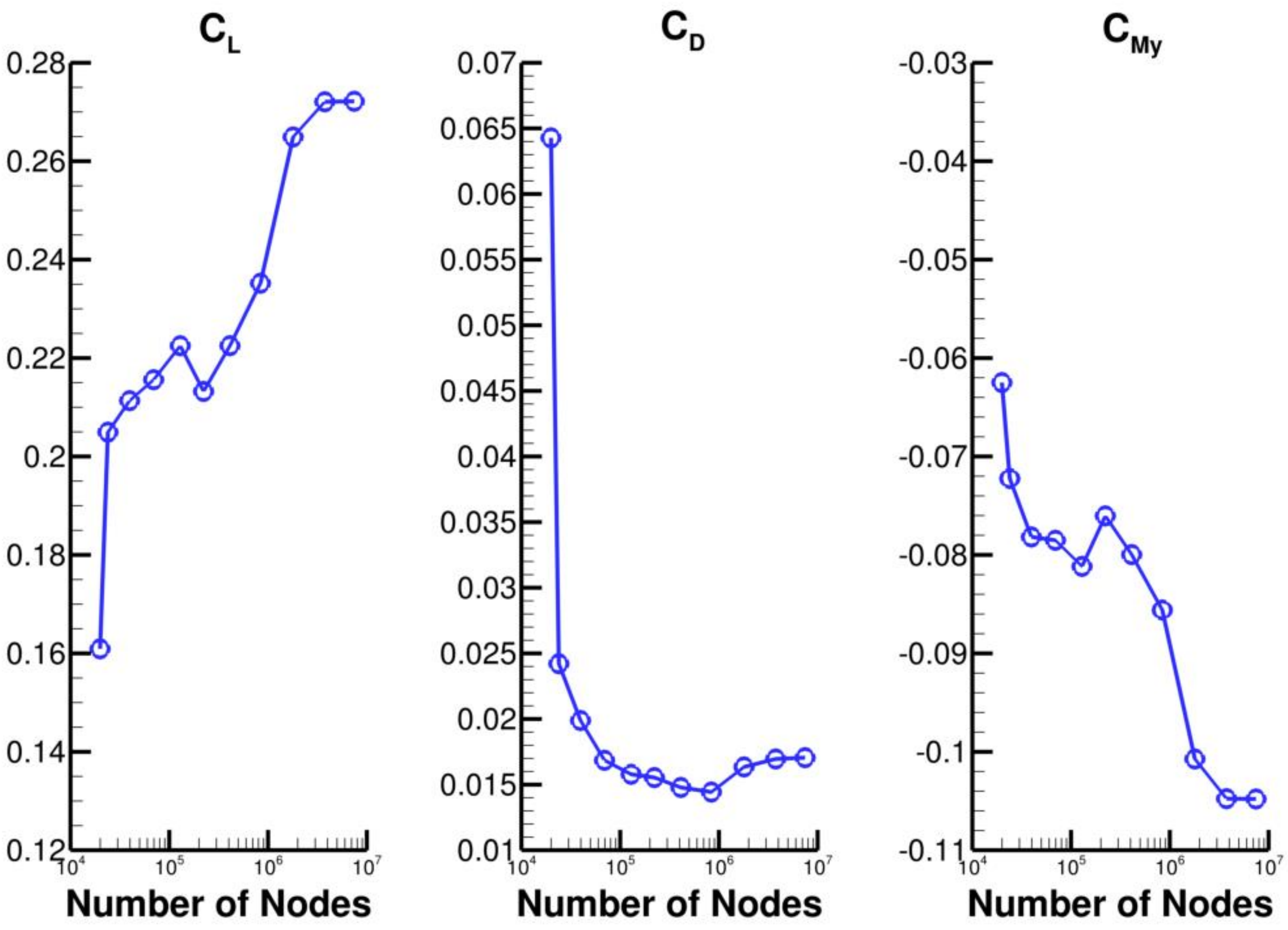


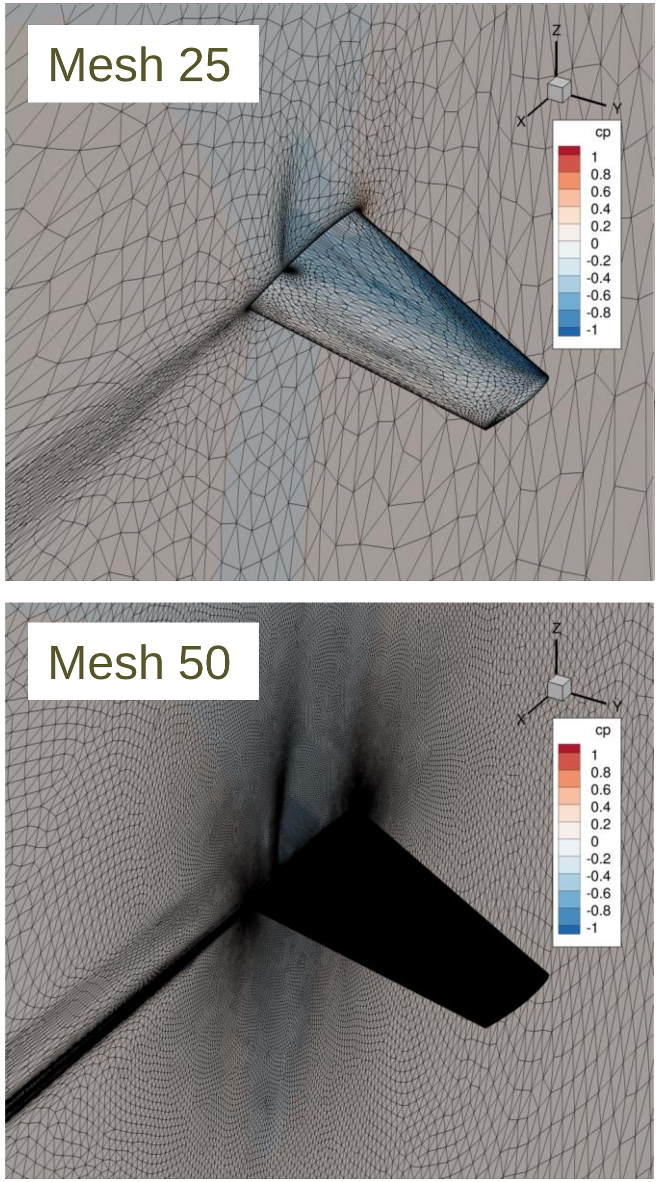


Tutorial case: Steady ONERA M6 Adaptation 7 of 7

- Post processing with pyrefine

  https://nasa.github.io/pyrefine/post_processing.html

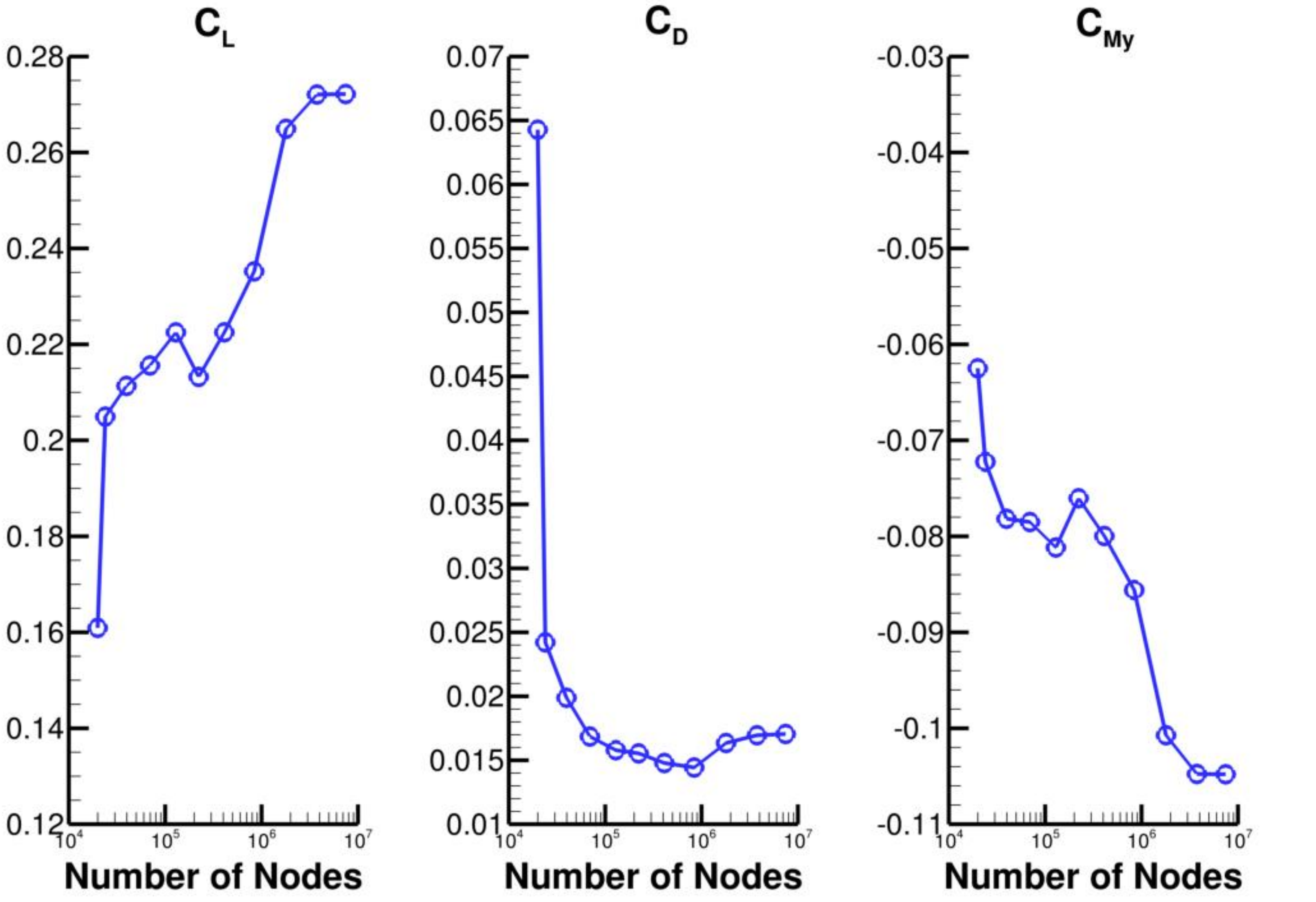


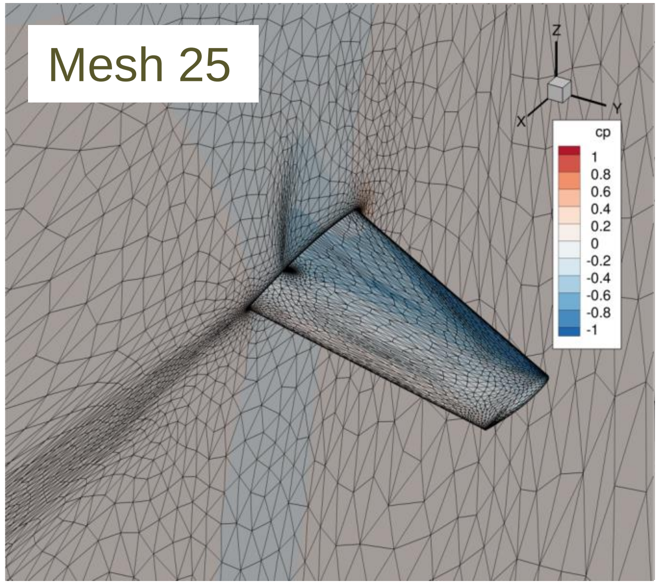


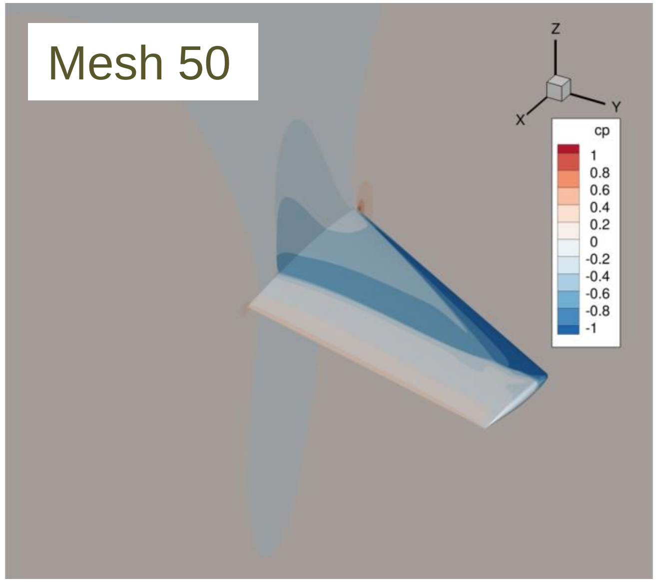
Tutorial case: Steady ONERA M6 Adaptation 7 of 7

- Post processing with pyrefine

  https://nasa.github.io/pyrefine/post_processing.html







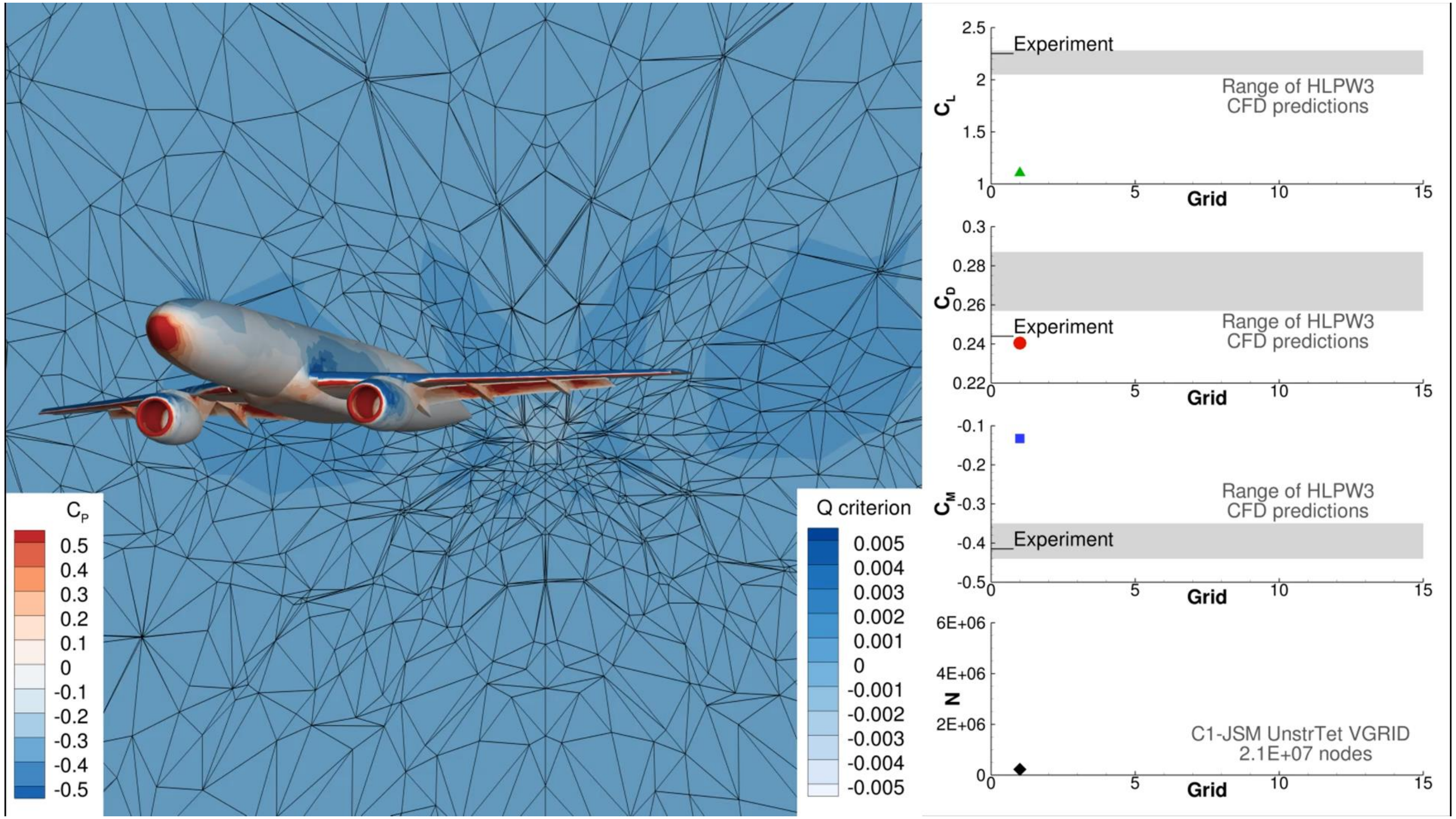
Summary

- Why add a SFE Solver?
- Training scope
- Compilation
- Shared components for SFE and FV in FUN3D
- Steady analysis with SFE
  - Supported modes
  - Nonlinear iteration
  - Input files
  - Output files
- Trouble shooting
- Tutorial cases
  - BSCW
  - ONERA M6 with goal-oriented adaptation

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

Request

- Share your successes with us! They help us advocate for resources to better support you.
- If there are capabilities you'd like to see in SFE, email us.



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

References

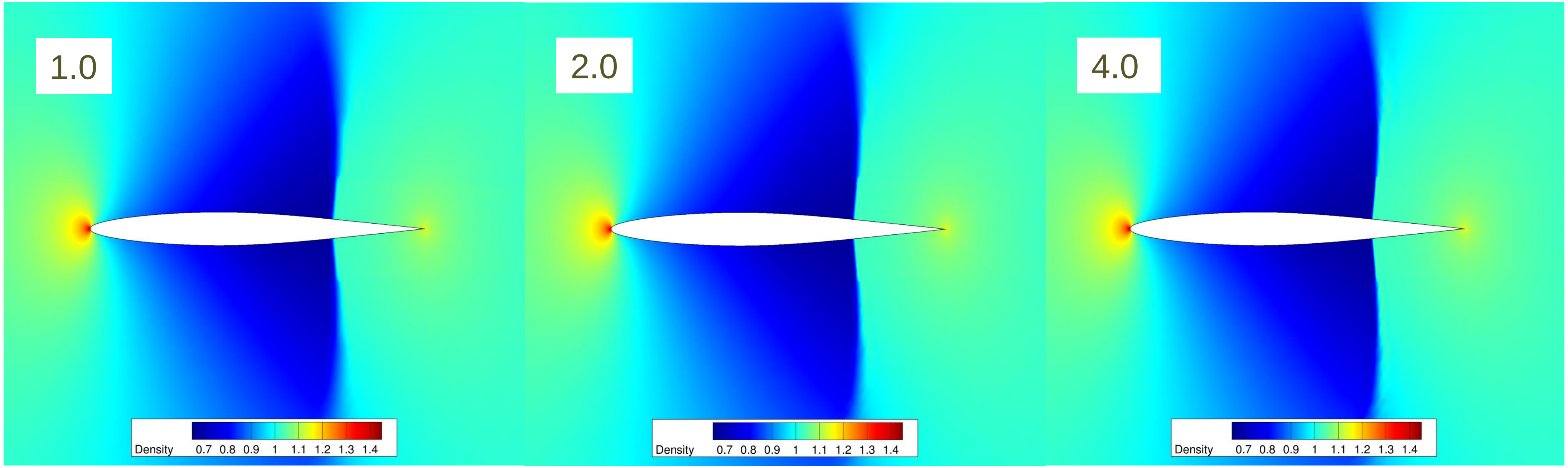
- **SFE**: "Stabilized Finite Elements in FUN3D" W.K. Anderson, J.C. Newman, S.L. Karman, Journal of Aircraft, 2018.
- **SLAT linear solver**: "Sparse Linear Algebra Toolkit for Computational Aerodynamics" S.L. Wood, K.E. Jacobson, W.T. Jones, W.K. Anderson, AIAA SciTech Forum, 2020.
- **K-ordering and Q-ordering**: "Node Numbering for Stabilizing Preconditioners Based on Incomplete LU Decomposition" W.K. Anderson, S.L. Wood, K.E. Jacobson, AIAA Aviation Forum 2020.
- **Goal-based mesh adaptation**: "Anisotropic Goal-Based Mesh Adaptation Metric Clarification and Development." D. S. Kamenetsky, J. A. Krakos, T. R. Michal, F. Clerici, F. Aluzet, A. Loseille, M. Park, S. L. Wood, A. Balan, M. C. Galbraith

Backup Slides

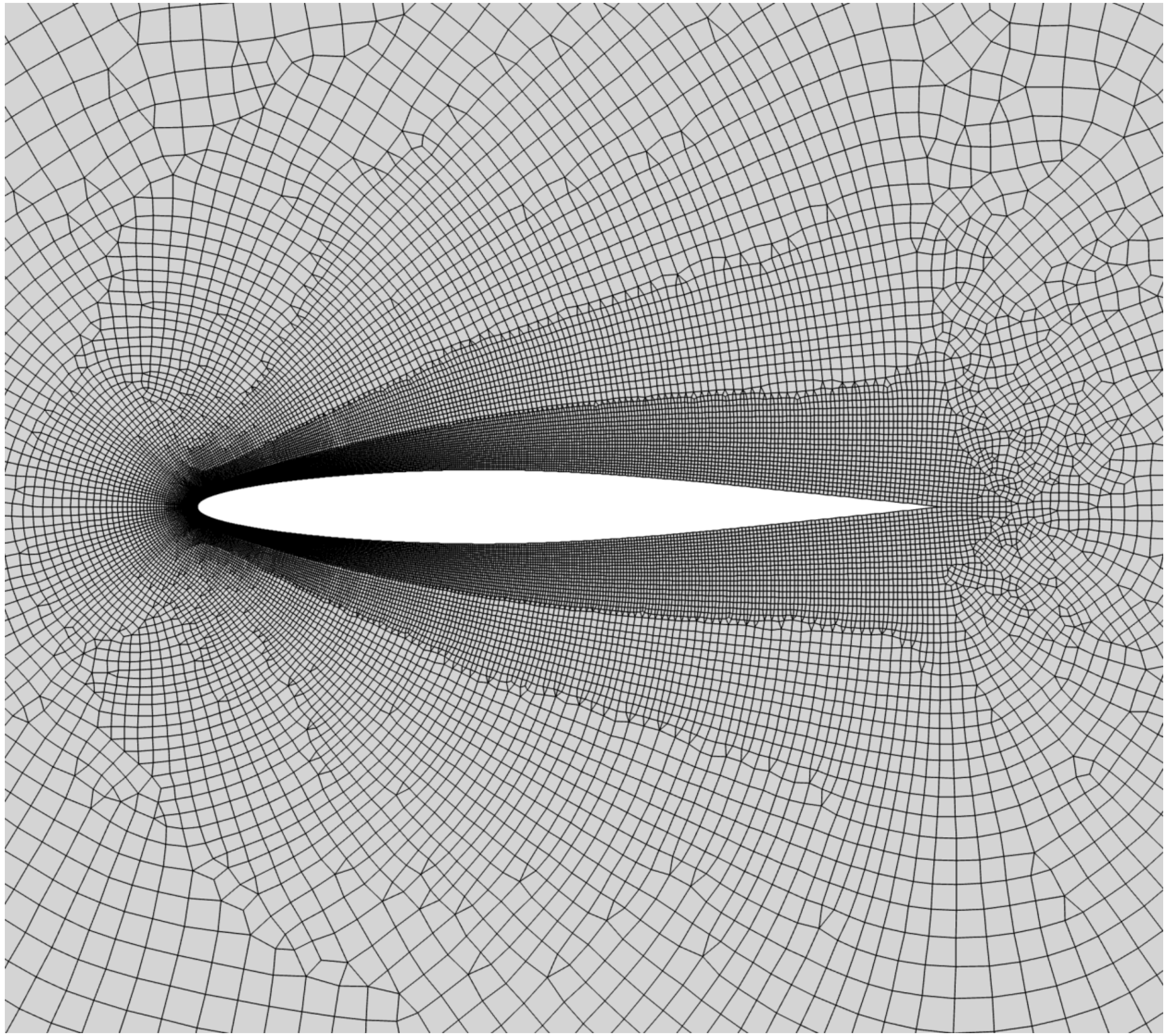
sfe.cfg :: smoothers 2 of 4

`smoother_type(i) = metric_pressure` - the default smoother which applies smoothing at shocks and expansions

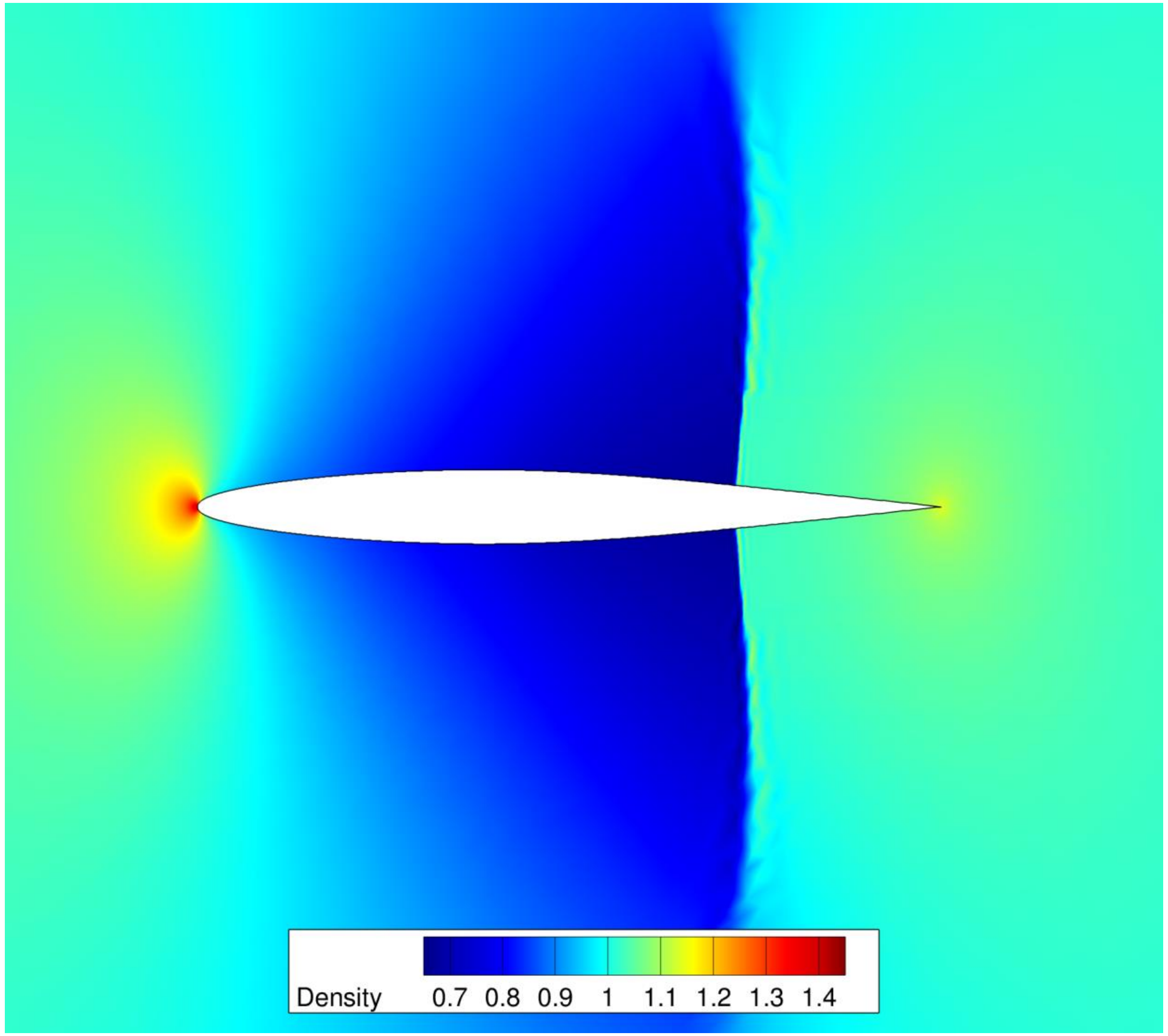
- **`smoother_clip(i) = 2.0`** – Typically use between 1.0 – 4.0, but we have used as low as 0.5
- **`smoother_coef(i) = 1.0`** – Typically leave at 1.0



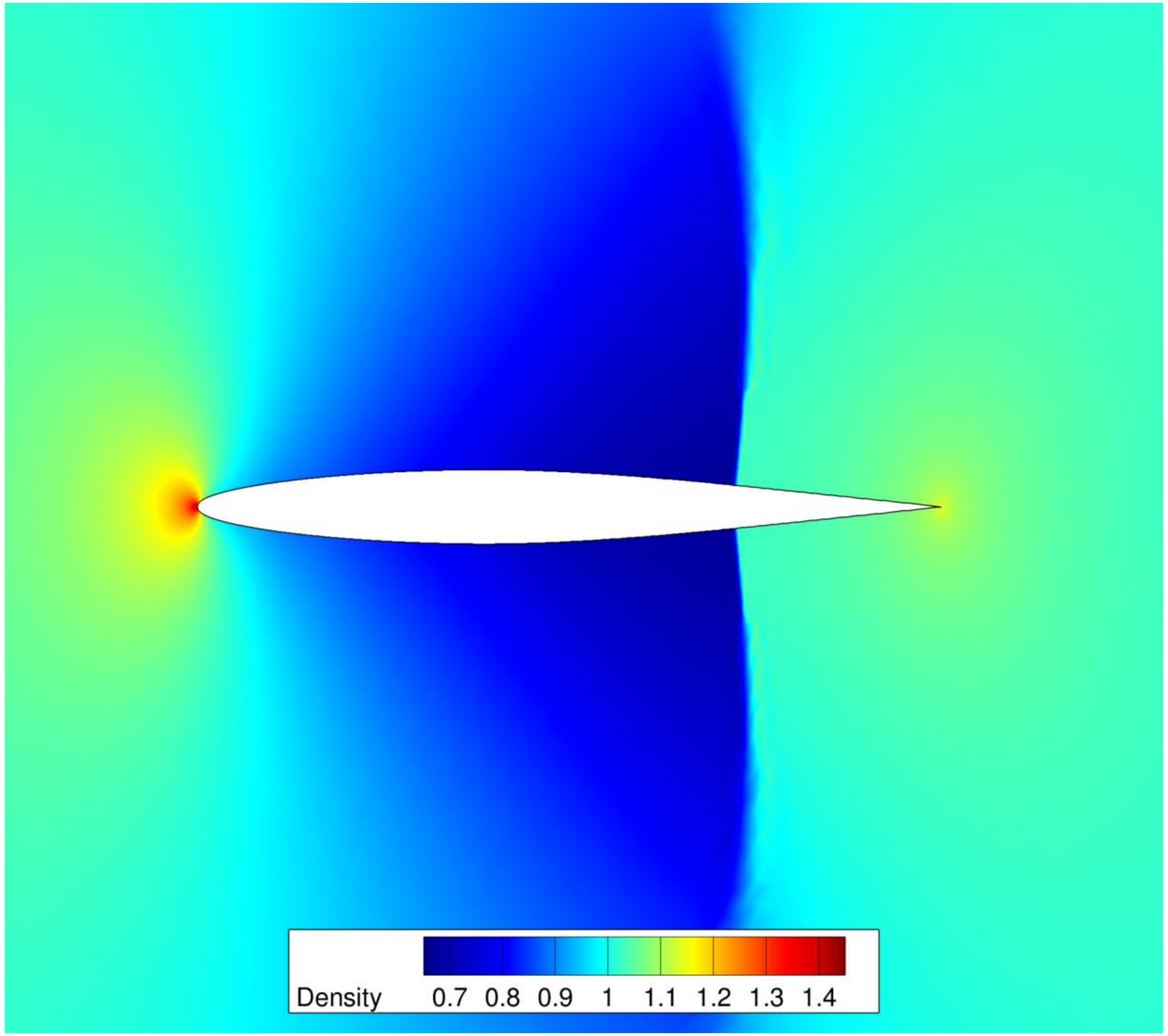
Mesh



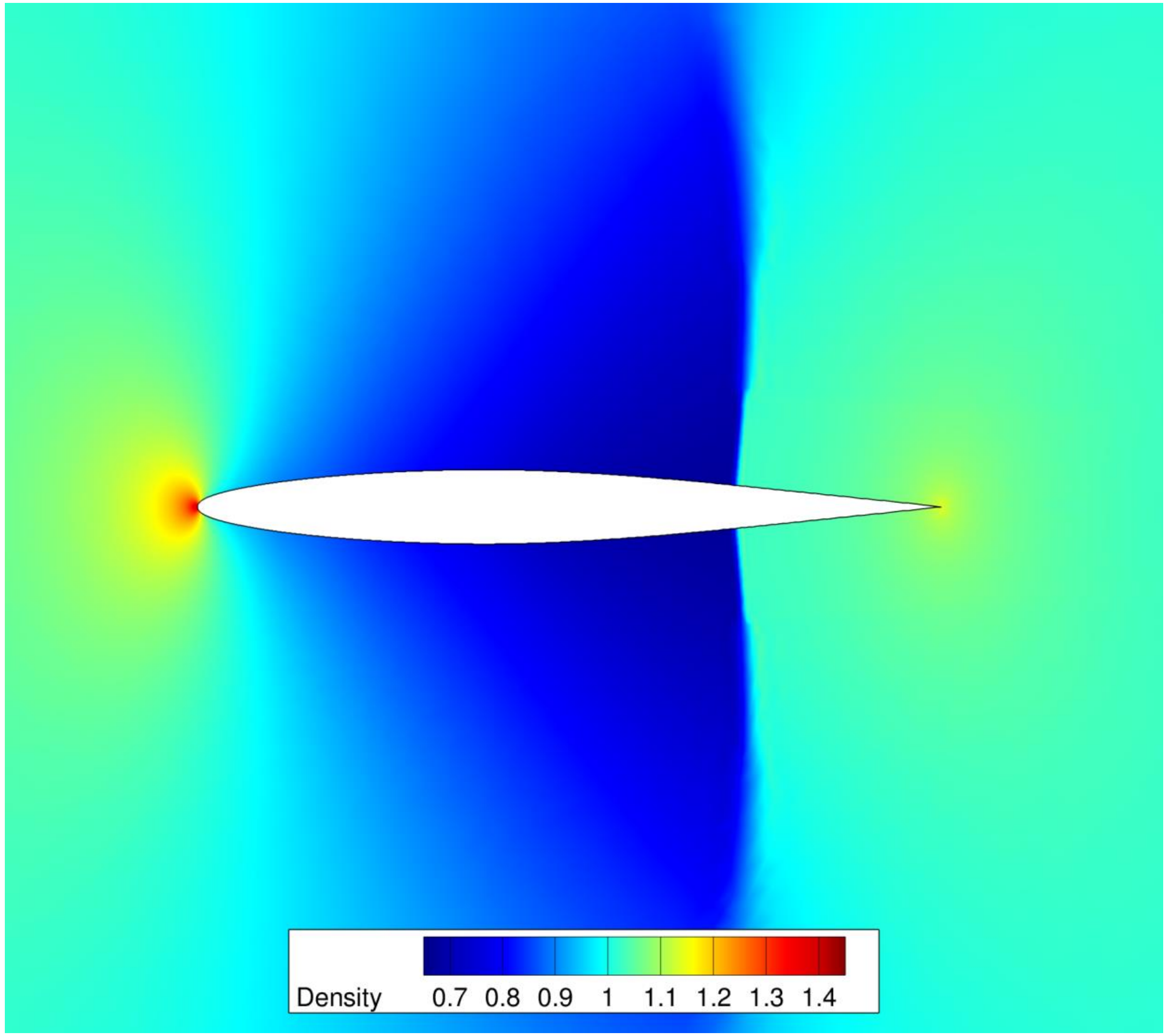
No Clip



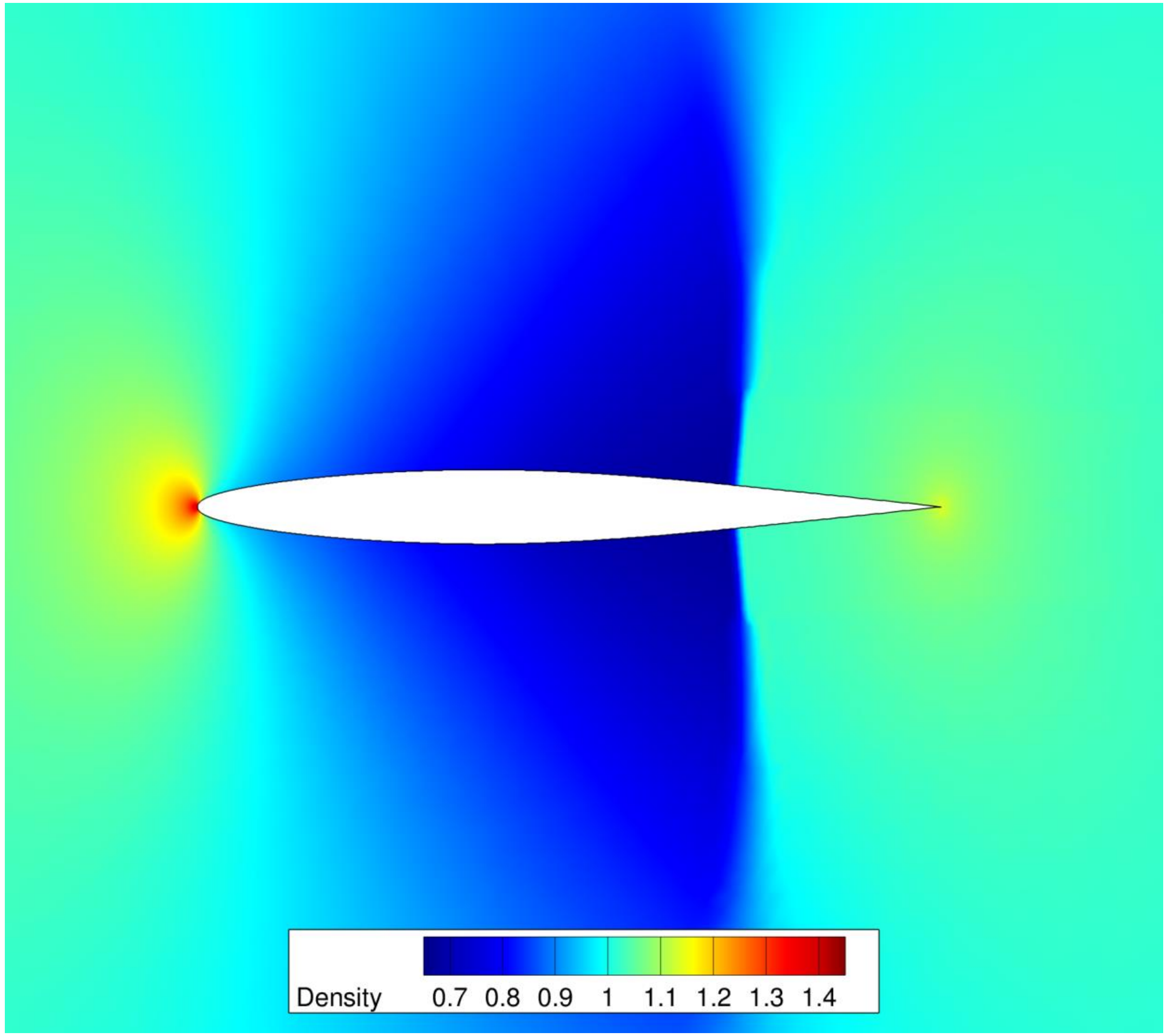
Clip=4.0



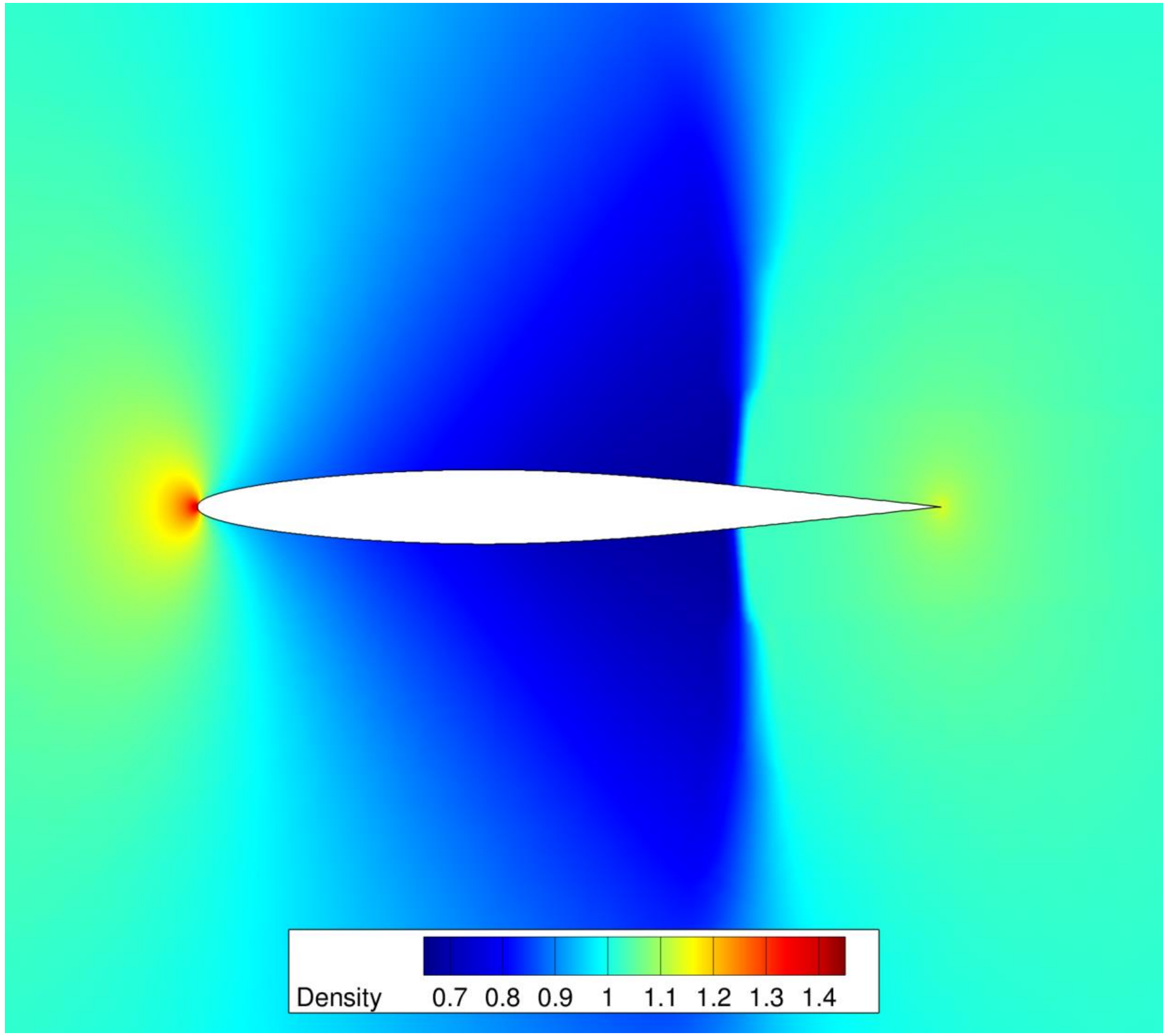
Clip=2.0



Clip=1.0



Clip=0.5



Clip=0.25