Session 4:
Gridding Considerations, Solution Basics, and Visualization

Eric Nielsen

Learning Goals

What we will cover

- Basic gridding requirements and formats
- Basic environment for running FUN3D
- Running FUN3D for typical steady-state RANS cases
  - Compressible transonic turbulent flow over a wing-body using a tetrahedral VGRID mesh
  - Turbulent flow over a NACA 0012 airfoil section
- Analysis of flowfield solution and solver behavior
- Things to watch out for
- Things to help diagnose problems
- Visualization co-processing overview

What we will *not* cover

- Other speed regimes
- Unsteady flows

Gridding Considerations

- FUN3D is a **node-based** discretization!
  - To get similar resolution when comparing with a cell-centered code, you must use a finer grid!
    - E.g., on a tetrahedral grid, the grid for FUN3D must be ~2 times finer on the surface, and ~6 times finer in the volume mesh to be fair
  - This is *critical* when comparing with cell-centered solvers!
- FUN3D integrates all of the way to the wall for turbulent flows
  - Wall function grids are not adequate
  - Goal is to place first grid point at $y^+=1$
    - Base Δy on a flat plate estimate using your Reynolds number; can examine result in solver output and tweak as necessary
- Customers use all of the common grid generators – VGRID, AFLR2/AFLR3/SolidMesh, ICEM, GridGen, etc.
- FUN3D also supports point-matched, multiblock structured grids through Plot3D file input
  - Subject to certain grid topologies:
    - Singularities treated – i.e., hexes with collapsed faces converted to prisms
    - But hexes with 180° internal angles cause FUN3D discretization to break down (LSQ)
- FUN3D can convert tetrahedral VGRID meshes to mixed elements
- FUN3D can convert any mixed element grid into tetrahedra using `'--make_tets'`

Supported Grid Formats

Grid Format	Formatted	Unformatted	Supports mixed elements	v11.3 Solver loads directly	File extension(s)
FAST	X	X		X	.fgrid, .mapbc
VGRID (single or multisegment)		X		X	.cogsg, .bc, .mapbc
VGRID v4		X	X	X	.gridu, .mapbc
AFLR3	X	X Also binary (stream)	X	X	.ugrid/.r8.ugrid/.b8.ugrid, .mapbc
FUN2D	X			X	.faces
Fieldview v2.4, v2.5, v3.0	X	X	X	X	.fvgrid_fmt, .fvgrid_unf, .mapbc
NSU3D		X	X	X	.mcell.unf, .mapbc
Point-matched, multiblock Plot3D	X	X	Structured Hexes	Eventually (Party capable)	.p3d, .nmf
Felisa	X			Eventually (Party capable)	.gri, .fro, .bco
CGNS		Binary	X	Eventually (Party capable)	.cgns

Nondimensionalization

- Notation: * indicates a dimensional variable, otherwise dimensionless; the reference flow state is ***usually*** free stream ("∞"), but need not be
- Define reference values:
  - L^*_{ref} = reference length of the physical problem (e.g. chord in ft)
  - L_{ref} = corresponding length in your grid (*dimensionless*)
  - ρ^*_{ref} = reference density (e.g. slug/ft^3)
  - μ^*_{ref} = reference molecular viscosity (e.g. slug/ft-s)
  - T^*_{ref} = reference temperature (e.g. ^{o}R, compressible only)
  - a^*_{ref} = reference sound speed (e.g. ft/s, compressible only)
  - U^*_{ref} = reference velocity (e.g. ft/s)
- Space and time are made dimensionless in FUN3D by:
  - $\vec{x} = \vec{x}^* / (L^*_{ref} / L_{ref})$ $\quad t = t^* a^*_{ref} / (L^*_{ref} / L_{ref})$ (compressible) $\quad t = t^* U^*_{ref} / (L^*_{ref} / L_{ref})$ (incompressible)

Nondimensionalization (cont)

- For the ***compressible-flow*** equations the dimensionless variables are:
  - $\vec{u} = \vec{u}^* / a^*_{ref}$ so $|\vec{u}|_{ref} = |\vec{u}|^*_{ref} / a^*_{ref} = M_{ref}$
  - $P = P^* / (\rho^*_{ref} a^{*2}_{ref})$ so $P_{ref} = P^*_{ref} / (\rho^*_{ref} a^{*2}_{ref}) = 1/\gamma$
  - $a = a^* / a^*_{ref}$ so $a_{ref} = 1$
  - $T = T^* / T^*_{ref}$ so $T_{ref} = 1$
  - $e = e^* / (\rho^*_{ref} a^{*2}_{ref})$ so $e_{ref} = e^*_{ref} / (\rho^*_{ref} a^{*2}_{ref}) = 1/(\gamma(\gamma - 1)) + M^2_{ref}/2$
  - $\rho = \rho^* / \rho^*_{ref}$ so $\rho_{ref} = 1$
  - From the equation of state and the definition of sound speed:

$$T = \gamma P / \rho = a^2$$

- The input Reynolds number in FUN3D is related to the Reynolds number of the physical problem by

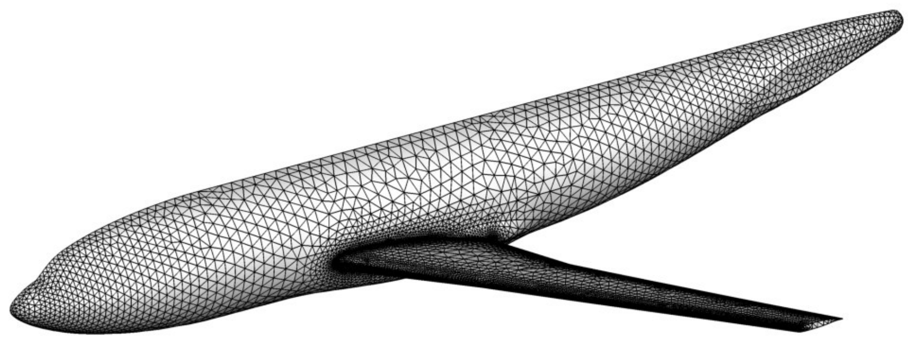
`reynolds_number` = $\mathrm{Re}_{ref} / L_{ref}$ where $\mathrm{Re}_{ref} = \rho^*_{ref} U^*_{ref} L^*_{ref} / \mu^*_{ref}$

i.e. `reynolds_number` is a Reynolds number ***per unit grid length***

Runtime Environment

- "Unlimit" your shell (also good idea to put this in any queue scripts):
  - csh: `'limit stacksize unlimited'`
  - bash: `'ulimit -s unlimited -d unlimited -m unlimited'`
- If unformatted, what "endianness" does your grid file have?
  - E.g., VGRID files are always big endian, regardless of platform
  - FUN3D must be set up with the corresponding endianness!
    - Either an environment variable or compile-time option, depending on what compiler you're using
    - I.e., Intel compiler can be controlled with an environment variable `F_UFMTENDIAN = big`
- Memory required by solver: *rough* rule of thumb is 3-3.5 GB per million points (not cells!)
  - Conversely, 200k-300k points per 1 GB of memory
    - Users generally partition into smaller domains than this anyways, but be aware of these numbers

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh



- Go down into the `Basics_Demos` directory

  `'cd Basics_Demos'`
- Go down into the `WingBody` directory

  `'cd WingBody'`
- This directory contains a set of files for a typical tetrahedral VGRID mesh

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- Edit the `f6fx2b_trn.mapbc` file and look at it
  - A good first step is to always examine/set the BC indices and family names
  - FUN3D understands VGRID-style BC indices; you may use either

Most Common FUN3D BC Indices (complete list on website)

BC Index (VGRID index)	Physical Boundary Condition
3000 (5)	Slip wall
4000 (4)	No-slip wall
5000 (3)	Characteristic inflow/outflow
6662 (1)	y-symmetry

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- Edit the `fun3d.nml` file and take a look at it
- Consists of several Fortran namelists
  - Many other optional namelists not shown
  - See website, subsequent tutorials for namelists, variable values

```
&version_number
    input_version      = 2.2
    namelist_verbosity = "off"
 /
```

Namelist version - leave alone

Cuts down on echos of input to screen

```
&project
    project_rootname = "f6fx2b_trn"
    case_title = "DPW-III DLR-F6FX2B Grid Mach 0.75 AOA 1deg REc 2.5M"
 /
```

Project name for your files

Case title for plotting files, etc

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

```
&governing_equations
   eqn_type       = "cal_perf_compress"
   viscous_terms = "turbulent"
/

&reference_physical_properties
   temperature_units = "Rankine"
   mach_number        = 0.75
   reynolds_number    = 17705.40
   temperature        = 580.0
   angle_of_attack    = 1.00
   angle_of_yaw       = 0.00
/
```

Use compressible, perfect gas physics
Use RANS equations

Temperature units
Freestream Mach number
Reynolds number
Freestream temperature
Freestream angle-of-attack
Freestream sideslip angle

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

<u>Setting the Reynolds Number Input</u>

- Frequent cause of confusion, even for developers
- Need to know what characteristic length your Reynolds number is based on – mean aerodynamic chord, diameter, etc.
- Your input Reynolds number is based on the corresponding length of that "feature" in your computational grid
- Example: You want to simulate a Reynolds number of 2.5 million based on the MAC:
  - If the length of the MAC in your grid is 1.0 grid units, you would input Re=2500000 into FUN3D
  - If the length of the MAC in your grid is 141.2 grid units (perhaps these physically correspond to millimeters), you would input 2500000/141.2, or Re=17705.4 into FUN3D

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

`&force_moment_integ_properties`		
`   area_reference =     72700.0`	Reference area	ALL IN GRID UNITS
`   x_moment_length =     141.20`	Length to normalize y-moments	
`   y_moment_length =     585.60`	Length to normalize x- and z-moments	
`   x_moment_center =     157.90`	x-moment center	
`   y_moment_center =     0.0000`	y-moment center	
`   z_moment_center =    -33.920`	z-moment center	
`/`		

`&inviscid_flux_method`	
`   flux_limiter           = "none"`	No limiting on reconstruction
`   first_order_iterations = 0`	No first-order iterations
`   flux_construction      = "roe"`	Use Roe's scheme for inviscid fluxes
`/`	

`&molecular_viscous_models`	
`   prandtlnumber_molecular =    0.72`	Prandtl number
`/`	

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

```
&turbulent_diffusion_models
   turb_model = "sa"
/
```

Use Spalart-Allmaras one-equation model

```
&nonlinear_solver_parameters
   time_accuracy         = "steady"
   pseudo_time_stepping  = "on"
   schedule_number       = 2
   schedule_iteration    = 1   50
   schedule_cfl          = 10.0  200.0
   schedule_cflturb      =  1.0   30.0
/
```

- time_accuracy: Perform steady-state integration
- pseudo_time_stepping: Use pseudo-time stepping
- schedule_number: No. of steps defining CFL ramp; must=2
- schedule_iteration: Starting and ending timesteps for CFL ramp
- schedule_cfl: Min and max CFL for meanflow equations
- schedule_cflturb: Min and max CFL for turbulence equations

```
&linear_solver_parameters
   meanflow_sweeps   = 15
   turbulence_sweeps = 10
/
```

- meanflow_sweeps: Number of sweeps through meanflow linear system
- turbulence_sweeps: Number of sweeps through turbulence linear system

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

```
&code_run_control
   steps                = 1000          Run 1000 time steps
   stopping_tolerance = 1.00E-12        Halt if density norm hits this (absolute measure)
   restart_write_freq = 500             How often to write restart info, forces, etc.
   restart_read         = "off"         No prior solution to start from
/

&special_parameters
   large_angle_fix = "on"               Ignore viscous fluxes in VGRID
/                                       “sliver” cells (>178° face angle)

&raw_grid
   grid_format    = "vgrid"             Input grid format
   data_format    = “unformatted”       Formatted grid?
   patch_lumping = “none”               How to lump raw boundary patches
/
```

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

```
&boundary_output_variables
   primitive_variables = .false.     Do not plot the primitive variables
   cp                  = .true.      Plot pressure coefficient
   yplus               = .true.      Plot y+
   uavg                = .true.      Plot averaged off-surface u-velocity
   vavg                = .true.      Plot averaged off-surface v-velocity
   wavg                = .true.      Plot averaged off-surface w-velocity
   cf_x                = .true.      Plot Cfx
   cf_y                = .true.      Plot Cfy
   cf_z                = .true.      Plot Cfz
   slen                = .true.      Plot distance function
/

&slice_data
   nslices           = 1             Number of slices to perform
   slice_location(1) = -160.         Location of slice
/
```

Extensive documentation on these and many other namelist input options available on the website

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- We now have the boundary conditions and input deck set up to run FUN3D
- The execution we will perform here is based on the following command line:

```
mpirun nodet_mpi --animation_freq -1 --slice_freq -1 > screen.output
```

  – Note command line options given to FUN3D:

`--animation_freq -1`	Write boundary visualization files after last time step
`--slice_freq -1`	Write slice visualization files after last time step

  – We will cover visualization specifics towards the end of this session
  – *Many* other command line options available
    - Obtain a list with `'--help'` or see the website for common user choices
    - Most are development-related
    - Command line options always trump `fun3d.nml`

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- Execute the job with the specified command line (modify MPI input for your environment)
- Watch FUN3D's screen output using `'tail -f screen.output'`
- The tail end of the screen output will look like this when FUN3D is complete:

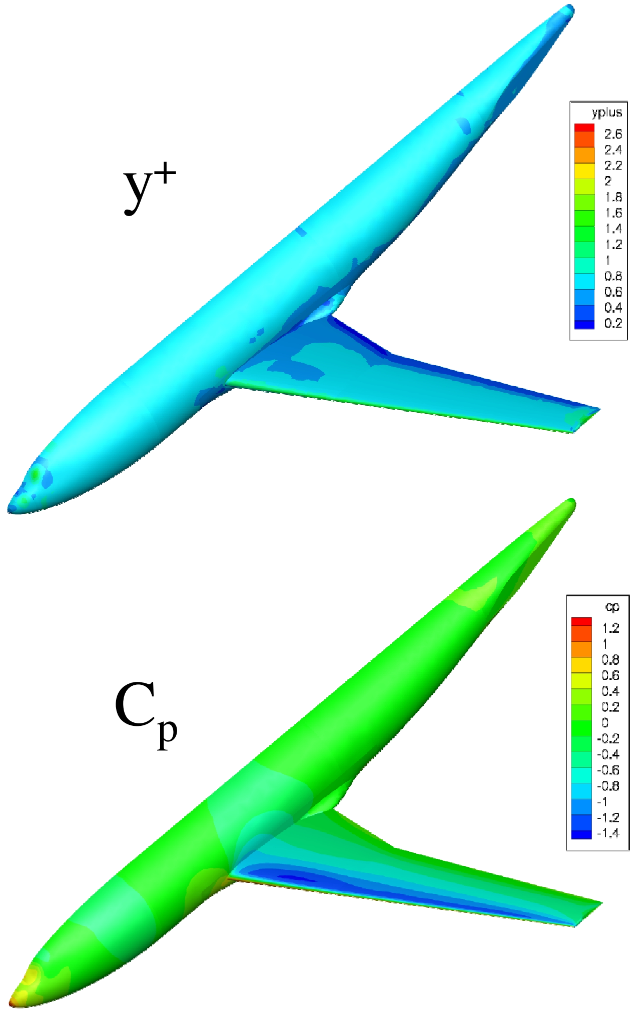
```
 498  0.190568203044346E-04  0.28547E-02  0.63496E+04 -0.38199E+04  0.18712E+04
      0.728925753784098E-01  0.11568E+02  0.46732E+04 -0.15204E+04  0.26710E+03
      Lift  0.554840528943358E+00         Drag  0.387987580717002E-01
 499  0.189000809038698E-04  0.28125E-02  0.63496E+04 -0.38199E+04  0.18712E+04
      0.719437041073944E-01  0.11410E+02  0.46732E+04 -0.15204E+04  0.26710E+03
      Lift  0.554839375833070E+00         Drag  0.387986031689365E-01
 500  0.187442268911612E-04  0.27699E-02  0.63496E+04 -0.38199E+04  0.18712E+04
      0.710082407016004E-01  0.11254E+02  0.46732E+04 -0.15204E+04  0.26710E+03
      Lift  0.554838233895265E+00         Drag  0.387984519049824E-01

Writing Tecplot boundary file for time step 500

Slicing global boundary data for time step   500

Writing sectional force/moment data for time step  500

Writing sliced global boundary data to tecplot file for time step  500

Writing f6fx2b_trn.flow (version   11.0000000000000      )           2
    ... Time for WREST    2.073100
Done.
```

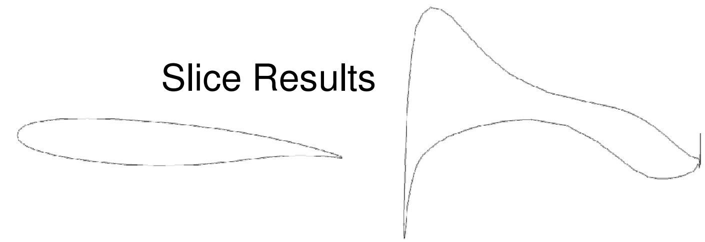
Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

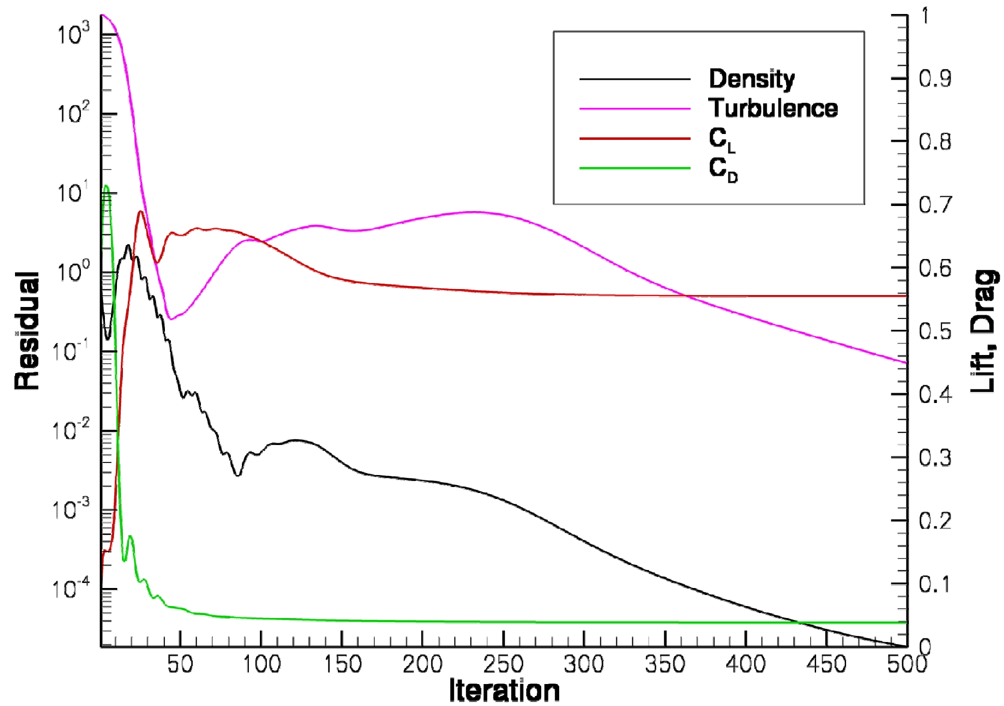
- Let's look at some quick text-based solver output:
  - Bring up `f6fx2b_trn.grid_info` File containing basic mesh statistics
  - Bring up `f6fx2b_trn.forces` File containing force breakdowns and totals
- You will also see several other files that got created:

`f6fx2b_trn_hist.dat`	Tecplot file with residual, force histories
`f6fx2b_trn_tec_boundary.dat`	Tecplot file with requested quantities on boundaries
`f6fx2b_trn_slice.dat`	Tecplot file with requested quantities at slice location
`f6fx2b_trn.flow`	Solver restart information

Demo Case #1: Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

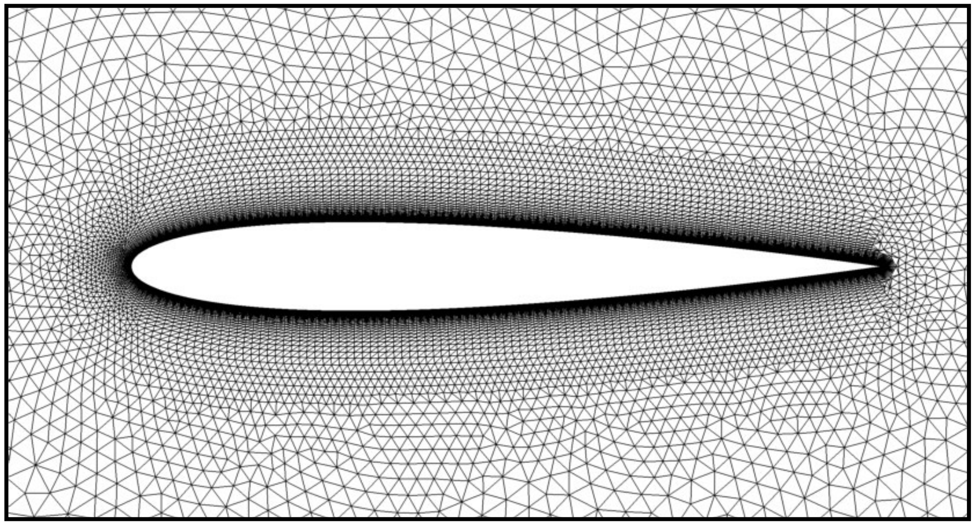






Congratulations, you just performed your first FUN3D solution!

Demo Case #2: NACA 0012 Airfoil



- Let's do turbulent flow over a NACA 0012 airfoil
- Go up one level and down into the `0012` directory
  - `'cd ../0012'`
- This directory contains a set of files for a typical FUN2D mesh

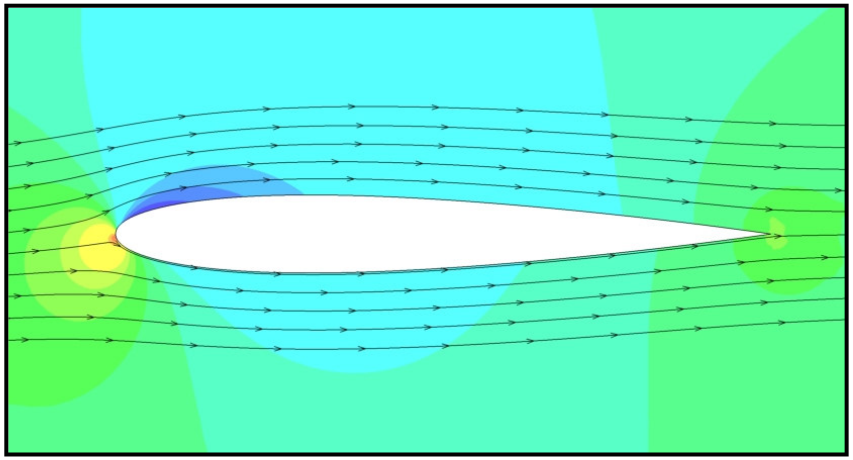
Demo Case #2: NACA 0012 Airfoil

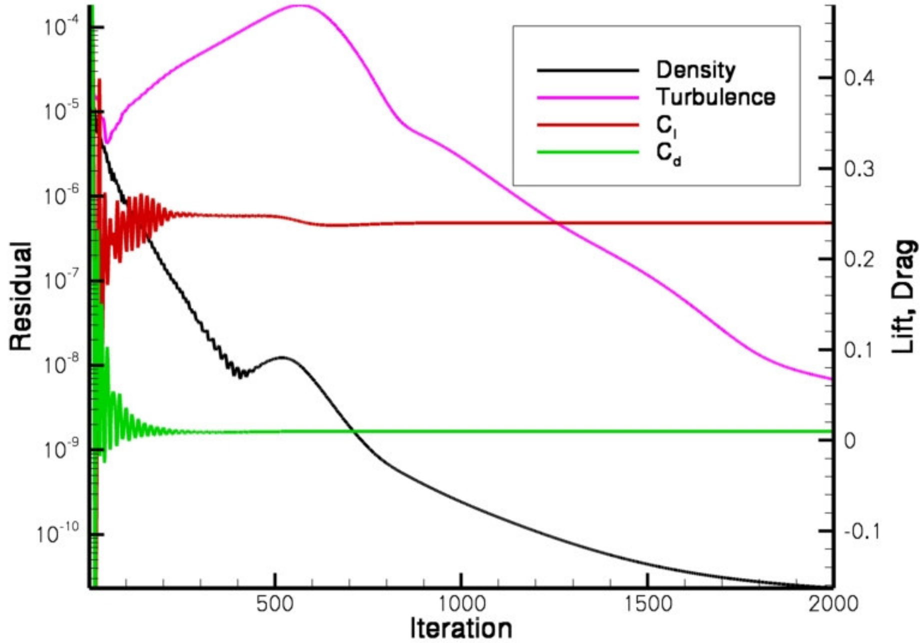
- The execution we will perform here is based on the following command line:

```
mpirun nodet_mpi --animation_freq -1 > screen.output
```

- Execute the job
- Have a look at the output, just like before:

```
0012.forces
0012_hist.dat
0012_tec_boundary.dat
```





List of Key Input/Output Files

- Input
  - Grid files (prefixed with `project`, suffixes depend on grid format)
  - `fun3d.nml`
- Output
  - `project.grid_info`
  - `project.forces`
  - `project_hist.dat`
  - `project.flow`
- Optional output generated here
  - `project_tec_boundary.dat` (or `.plt` if using binary Tecplot libraries)
  - `project_slice.dat` (or `.plt` if using binary Tecplot libraries)

What Could *Possibly* Go Wrong?

Problem

- Common complaint from VGRID meshes during initial preprocessing phase at front end of solver:

```
Checking volume-boundary connectivity...

stopping...unable to find common element for face       1 of
boundary       3
boundary nd array      46  17368 334315

node,locvc          46******************************
node,locvc_type     46    tet    tet    tet    tet    tet

node,locvc       17368************************************
node,locvc_type  17368    tet    tet    tet    tet    tet    tet
```

- This is due to a 15-year old VGRID bug that causes an incompatibility between the `.cogsg` and `.bc` files
  - Compile and run `utils/repair_vgrid_mesh.f90` to generate a valid `.bc` file to replace your original one

What Could *Possibly* Go Wrong?

Problem

- Common complaint from unformatted meshes during initial preprocessing phase at front end of solver:

```
Read/Distribute Grid.
forrtl: severe (67): input statement requires too much data, unit 16100,
file /misc/work14/user/FUN3D/project.cogsg
```

- Check the endian-ness of the grid and your environment/executables

Problem

- Unexpected termination, especially during preprocessing or first time step
  - Are your shell limits set?
  - Do you have enough local memory for what you are trying to run?

What Could *Possibly* Go Wrong?

Problem

- Solver diverges or does not converge
  - Problem-dependent, very tough to give general advice here
  - Sometimes require first-order iterations (primarily for high speeds)
  - Sometimes require smaller CFL numbers
  - Sometimes require alternate flowfield initialization (not freestream) in some subregion of the domain: e.g., chamber of an internal jet
  - Perhaps your problem is simply unsteady

Problem

- Solver suddenly dies during otherwise seemingly healthy run
  - Sometimes useful to visualize solution just before failure
  - Is it a viscous case on a VGRID mesh? Try turning on `large_angle_fix` in `fun3d.nml` (viscous flux discretization goes haywire in sliver cells common to VGRID meshes)
  - Is it a turbulent flow on a mesh generated using AFLR3? Look for “eroded” boundary layer grids near geometric singularities – AFLR3 has trouble adding viscous layers near complex corners, etc

What Could *Possibly* Go Wrong?

In General…

- Do not hesitate to send questions to *fun3d-support@lists.nasa.gov* ; we are happy to try to diagnose problems
  - Please send as much information about the problem/inputs/environment that you can, as well as all screen output
  - In extreme cases, we may request your grid and attempt to run a case for you to track down the problem
  - If you cannot send us a case due to restrictions, size, etc., a generic/smaller representative case that behaves similarly can be useful
  - Check the website documentation for guidance

Visualization Learning Goals

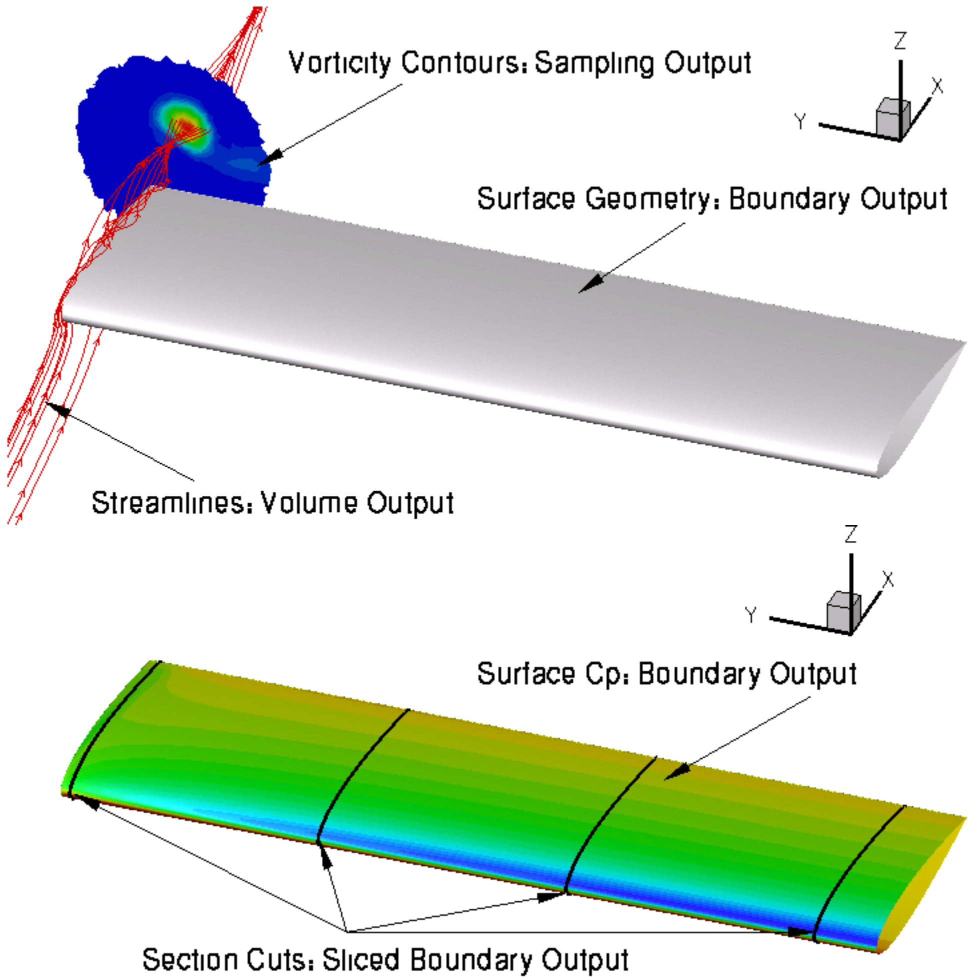
- What this will teach you
  - Flow visualization output from v11.x
    - Output on boundary surfaces
    - Output on user-specified “sampling” surfaces within the volume
    - Output of full volume data
    - Output generated by “slicing” boundary data - “sectional” output
- What you will not learn
  - Tecplot usage
- What should you already know
  - Basic flow solver operation and control

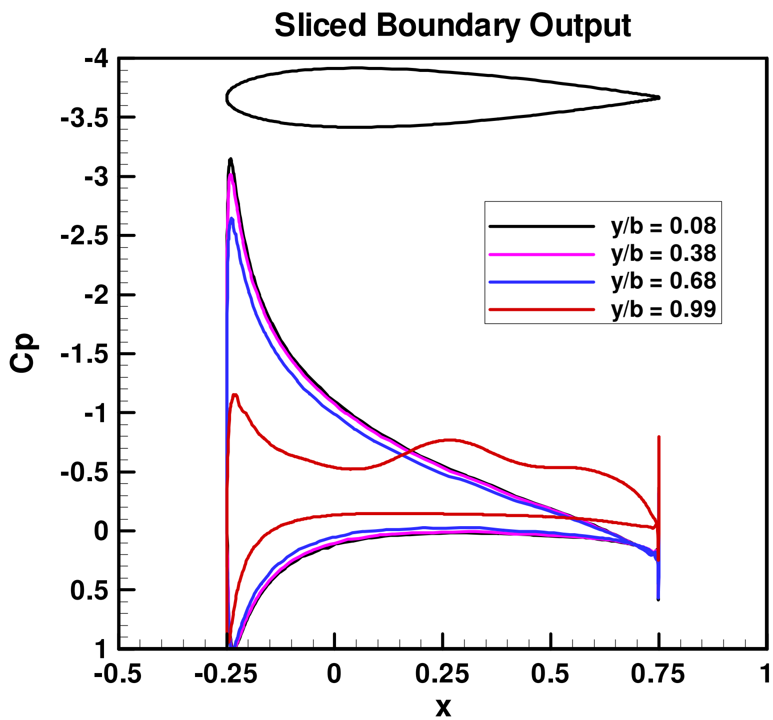
Setting

- Background
  - FUN3D v11.x effectively does away with the Party code for both pre- and post-processing
    - Post-processing with Party code was always done on single processor - slow and memory bound
    - Limited variable choice within a particular output option
    - Party supported both Tecplot and FieldView visualizers
  - v11.x allows visualization data to be generated as the solver is running - “co-processing” - in parallel
    - User selectable output variables from a fairly extensive list
    - User selectable output boundaries
- Compatibility
  - Only Tecplot is supported at this time
- Status
  - Hope to support FieldView visualizer in the future

Selected Co-Processing Output Examples

- A few of the possibilities for co-processed output - all generated at one time





Iso-surfaces and schlieren visualization will be available shortly

Co-Processing Overview

- All of the "co-processing" visualization output requires similar command line options to activate:
  - Boundary surface output: **`--animation_freq +/-N`**
  - Sampling surface output: **`--sampling_freq +/-N`**
  - Volume output: **`--volume_animation_freq +/-N`**
  - Sliced boundary surface output: **`--slice_freq +/-N`**
  - In all cases, N = 0, 1, 2, 3, …
    - N = 0 generates no output
    - N < 0 generates output only at the ***end*** of the run - typically used for steady-state cases. The actual value of N (other than 0) is ignored
    - N > 0 generates output every N^{th} time step - typically used to generate animation for unsteady flows; can also be used to observe how a steady flow converges

Co-Processing Overview

- Customizable output variables (except sliced boundary data):
  - Most variables are the same between the boundary surface, sampling and volume output options; boundary surface has a few extra
  - See website for lists of all available variables
  - Default variables always include x, y, z, and the "primitive" flow variables u, v, w, and p (plus density if compressible)
  - Several "shortcut" variables: e.g. primitive_variables = rho, u, v, w, p
  - Must explicitly turn off the default variables if you don't want them (e.g. **`primitive_variables = .false.`**)
  - Variable selection for each co-processing option done with a different namelist to allow "mix and match"
    - **`--animation_freq --> &boundary_output_variables`**
    - **`--volume_animation_freq --> &volume_output_variables`**
    - **`--sampling_freq --> &sampling_output_variables`**

Co-Processing Overview

- For boundary surface output, default is all solid boundaries in 3D and one y=const. plane in 2D; alternate output boundaries are selected with (e.g.)

```
&boundary_output_variables
   number_of_boundaries = 3
   boundary_list = '3,5,9'      ! blanks OK as delimiter too: '3 5 9'
                                ! dashes OK as delimiter too: '3-9'
 /
```

- If you already have a converged solution and don't want to advance the solution any further, can do a "pass through" run:
  - set **`steps = 0`** in **`&code_run_control`** of **`fun3d.nml`**
  - You must have a restart file (**`[project].flow`**)
  - Run the solver with the appropriate CLO's and namelist input to get desired output
  - **`[project].flow`** will remain unaltered after completion

Co-Processing Overview

- Sampling output requires additional data to describe the desired sampling surface(s)
  - Specified in namelist **`&sampling_parameters`** in **`fun3d.nml`**
  - Surfaces may be planes, quadrilaterals or circles of arbitrary orientation, or may be spheres or boxes
  - See website for complete info
- Sliced boundary surface output requires additional data to describe the desired slice section(s)
  - Specified in namelist **`&slice_data`** in **`fun3d.nml`**
  - Always / only outputs x, y, z, Cp, Cfx, Cfy, Cfz
  - User specifies which (solid) boundaries to slice, and where
  - See website for complete info

Co-Processing Overview

- Output files will be ASCII, unless you have configured FUN3D against the Tecplot library (exception: sliced boundary data is always ASCII)
  - ASCII files have .dat extension
  - Binary files have .plt extension - smaller files; load into Tecplot faster
  - Boundary output file naming convention (T = time step counter):
    - **`[project]_tec_boundary_timestepT.dat`** if N > 0
    - **`[project]_tec_boundary_.dat`** if N < 0
  - Volume output file naming convention (note: 1 file *per processor* P)
    - **`[project]_partP_tec_volume_timestepT.dat`** if N > 0
    - **`[project]_partP_tec_volume_.dat`** if N < 0
  - Sampling output file naming convention (one file per sampling geometry G):
    - **`[project]_tec_sampling_geomG_timestepT.dat`** if N > 0
    - **`[project]_tec_sampling_geomG_.dat`** if N < 0

Troubleshooting/FAQ

- When linked to the Tecplot library archive an error occurs:
  - `Err: (TECEND111) File 1 is being closed without writing connectivity data.    Zone 46 was defined with a Classic FE zone type but TECNOD111() was not called.`
  - If this is from a Tecplot2008 installation, likely due to the fact that the original archive from Tecplot had an error - get an updated version from Tecplot
- I can see what look like ragged dark lines on sampling surfaces and volume data
  - Duplicate information at partition boundaries is not removed; if surface is not completely opaque, double plotting locally doubles the opaqueness (duplicate info ***is*** removed from boundary surface output)
  - Turn off transparency in Tecplot (seems on by default in Tecplot2009)

What We Learned

- Basic gridding requirements and file formats
- Runtime environment
- How to set up boundary conditions for the grid and a very basic FUN3D input deck `fun3d.nml`
- How to run a tetrahedral RANS solution for a wing-body VGRID mesh
  - How to monitor convergence of the solution
  - Visualizing the solution
- How to perform a 2D airfoil solution using a FUN2D grid
- Some unhealthy things to watch for and possible remedies
- Overview of visualization co-processing

Have fun and don't hesitate to send questions our way!

fun3d-support@lists.nasa.gov