FUN3D v13.4 Training

Session 4:
Gridding and Solution Basics

Eric Nielsen

Learning Goals

What we will cover

- Basic gridding requirements and formats
- Nondimensionalizations and axis conventions
- Basic environment for running FUN3D
- FUN3D user inputs
- 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
- Things to help diagnose problems

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
  - Hanging nodes are not currently supported
- Historically, FUN3D integrates all of the way to the wall for turbulent flows
  - 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
  - Wall functions have recently been added but are not covered here
- Users employ all of the common grid generators – VGRID, AFLR2/AFLR3/SolidMesh, ICEM, Pointwise, 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)
- Utilities provided with FUN3D can convert tetrahedral VGRID meshes to mixed elements or any mixed element grid to tetrahedral form

Supported Grid Formats

Grid Format	Formatted	Unformatted	Supports mixed elements	Direct load or converter	File extension(s)
FAST	X	X		Direct	.fgrid, .mapbc
VGRID (single or multisegment)		X		Direct	.cogsg, .bc, .mapbc
AFLR3	X	X Also Binary	X	Direct	.ugrid/.(l)r8.ugrid/.(l)b8.ugrid, .mapbc
FUN2D	X			Direct	.faces
Fieldview v2.4, v2.5, v3.0	X	X	X	Direct (Some details of format not supported)	.fvgrid_fmt, .fvgrid_unf, .mapbc
Felisa	X			Direct	.gri, .fro, .bco
Point-matched, multiblock Plot3D	X	X	Hexes, degenerates	Converter	.p3d, .nmf
CGNS		Binary	X	Converter	.cgns

The development team can work with you to handle other formats as needed

Boundary Condition Input File

- Where required, the FUN3D .mapbc file takes the form:

```
Number of boundary patches
Boundary patch index    BC index      Family name
```

- The BC index may be either a 4-digit FUN3D-style index or a GridTool-style index
- The family name is optional, but must be present if the user requests patch lumping by family

```
3
1  4000    Wing
2  5000    Farfield
3  6662    Symmetry plane
```

- Exception: The .mapbc format for VGRID meshes follows the GridTool/VGRID format

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

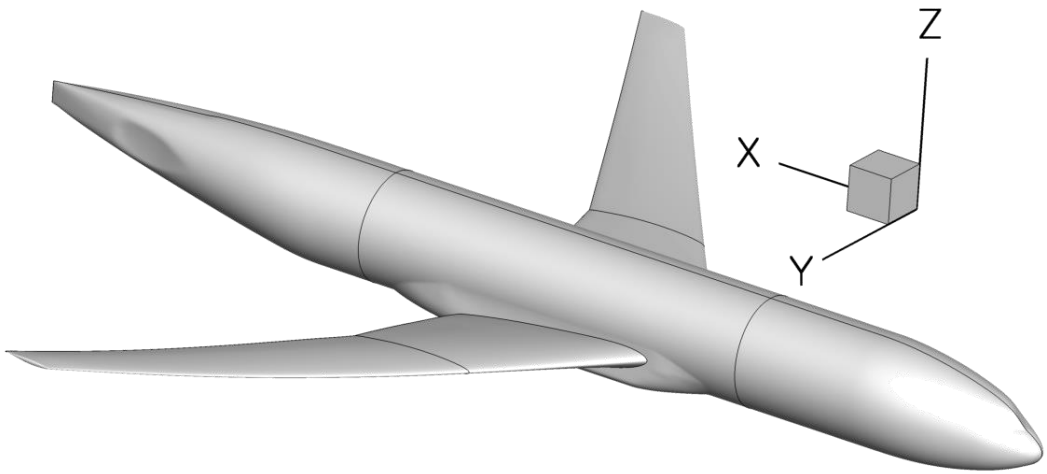
reynolds_number = $\mathrm{Re}_{ref} / L_{ref}$ where $\mathrm{Re}_{ref} = \rho^*_{ref} U^*_{ref} L^*_{ref} / \mu^*_{ref}$

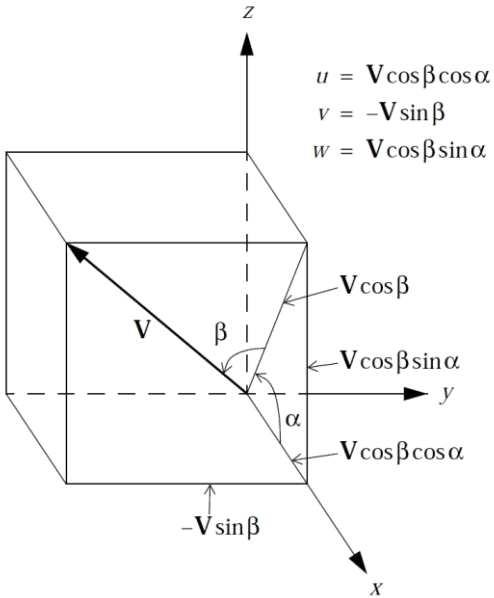
i.e., reynolds_number is a Reynolds number ***per unit grid length***

Setting the Reynolds Number Input

- 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

FUN3D Axis Convention





- FUN3D coordinate system differs from the standard wind coordinate system by a 180º rotation about the y-axis
  - Positive x-axis is toward the “back” of the vehicle (downstream)
  - Positive y-axis is out the “right wing”
  - Positive z-axis is “upward”
- The freestream angle of attack and yaw angle are defined as shown

Runtime Environment

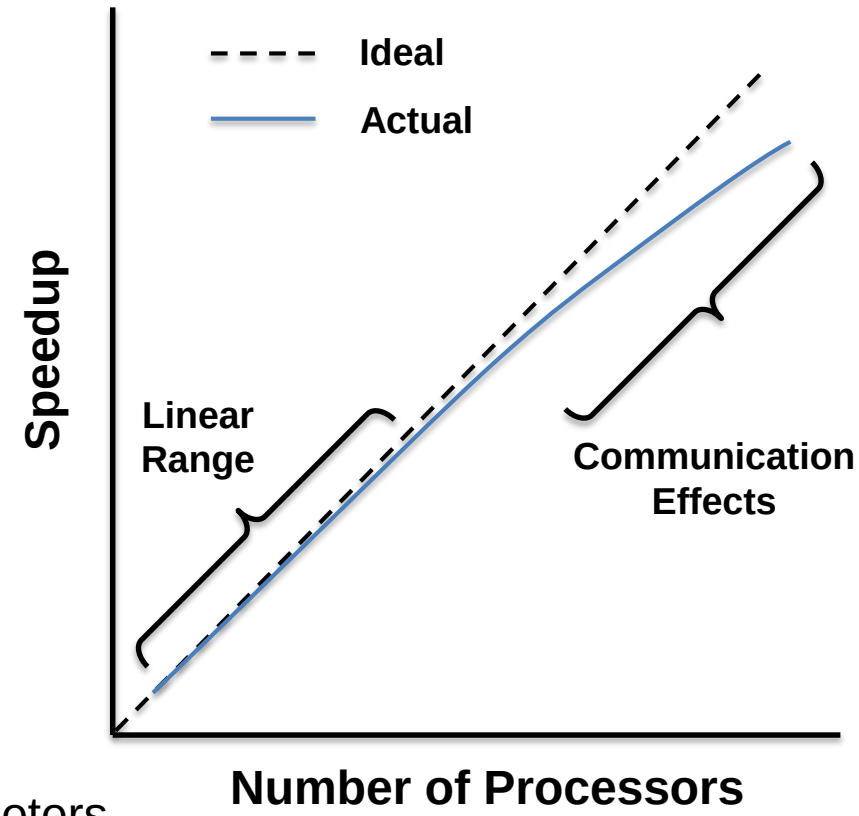
- "Unlimit" your shell (also good idea to put this in any queue scripts):

```
$ ulimit unlimited # for bash
$ unlimit          # for c shell
```

- If unformatted or binary, what "endianness" does your grid file have?
  - E.g., VGRID files are always big endian, regardless of platform
  - If your compiler supports it, FUN3D will attempt to open files using an `open(convert=…)` syntax
  - Most compilers support some means of conversion
    - Either an environment variable or compile-time option, depending on what compiler you're using
    - E.g., 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, but be aware of these numbers
  - This memory estimate will be higher if visualization options are used, etc.

Parallel Processing

- Parallel processing allows FUN3D to complete your job much faster than when using a single processor
- At best, your parallel speedup will be proportional to the additional number of processors you allow FUN3D to use; e.g., twice as many processors should run your job twice as fast
- Parallel processing requires communication, so actual speedup is related to the ratio of computation to communication
- These factors depend on many parameters of your problem, your hardware, and FUN3D



- It is most efficient to run in the linear range, but you may choose to use more resources if you just want faster turnaround
  - A conservative guess for FUN3D is to use ~50,000 grid points per processor

User Inputs for FUN3D

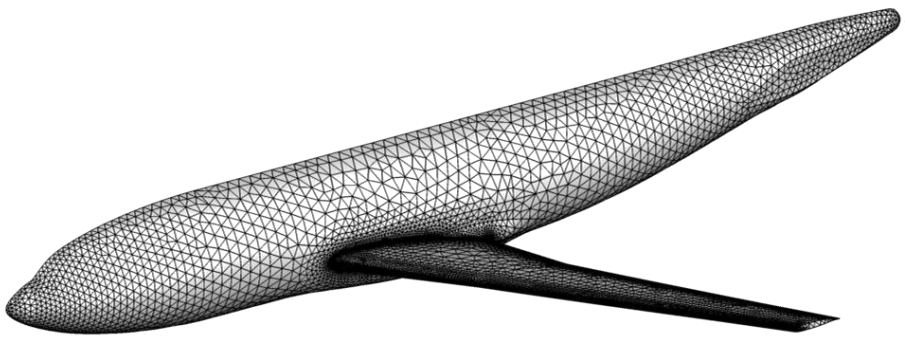
Input deck `fun3d.nml`

- The user is required to supply an input deck for FUN3D named `fun3d.nml` (fixed name)
- This filename contains a collection of Fortran namelists that control FUN3D execution – all namelist variables have default values as documented
- But user will need to set at least some high-level variables, such as the project name

Command Line Options (CLOs)

- CLOs always take the form `--command_line_option` after the executable name
  - Some CLOs may require trailing auxiliary data such as integers and/or reals
- User may specify as many CLOs as desired
- CLOs always trump `fun3d.nml` inputs
- CLOs available for a given code in the FUN3D suite may be viewed by using `--help` after the executable name
- Most CLOs are for developer use; namelist options are preferred where available

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh



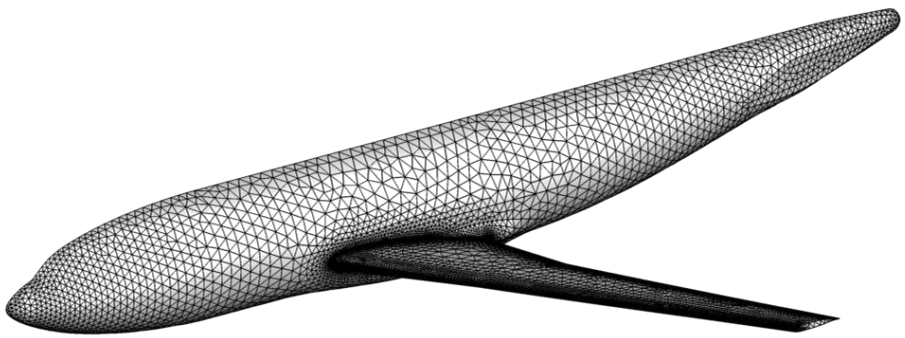
- For this case, we will assume that someone has provided a set of VGRID files containing the mesh
  - f6fx2b_trn.cogsg, f6fx2b_trn.bc, and f6fx2b_trn.mapbc
- It is always a good idea to examine the .mapbc file first to check the boundary conditions and any family names
  - Note that specific boundary conditions will be covered in a separate session

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- For this case, the VGRID/GridTool-style .mapbc file is as shown
- Surface grid consists of 51 patches
- Note that VGRID/GridTool-style BCs are specified
- Family names are also as shown (required in this format)
- FUN3D does not use the other columns of data
- If you cannot easily visualize your mesh to set appropriate boundary conditions, one easy approach is to set them all to inflow/outflow, then run a single time step of FUN3D with boundary visualization activated – then set patch BCs as needed for actual simulation

```
#Thu Mar 11 13:42:40 2010
#bc.map
Patch #        BC             Family   #surf   surfIDs         Family
#-----------------------------------------------------------------------
1              3              3              0             0         Box
2              3              3              0             0         Box
3              3              3              0             0         Box
4              3              3              0             0         Box
5              3              3              0             0         Box
6              4              4              1             15        Wing
7              4              4              1             15        Wing
8              4              4              1             17        Wing
9              4              4              1             17        Wing
10             4              4              1             15        Wing
11             4              4              1             13        Fuselage
12             4              4              1             21        Fuselage
13             4              4              1             11        Fuselage
14             4              4              1             11        Fuselage
15             4              4              1             12        Fuselage
16             4              4              1             12        Fuselage
17             4              4              1             15        Wing
18             4              4              1             15        Wing
19             4              4              1             15        Wing
20             4              4              1             15        Wing
21             4              4              1             17        Wing
22             4              4              1             17        Wing
23             4              4              1             16        Wing
24             4              4              1             15        Wing
25             4              4              1             17        Wing
26             4              4              1             8         Fuselage
27             4              4              1             16        Wing
28             4              4              1             16        Wing
29             4              4              1             16        Wing
30             4              4              1             16        Wing
31             4              4              1             18        Wing
32             4              4              1             18        Wing
33             4              4              1             17        Wing
34             4              4              1             18        Wing
35             4              4              1             18        Wing
36             4              4              1             1         Wing
37             4              4              1             18        Wing
38             4              4              1             18        Wing
39             4              4              1             18        Wing
40             4              4              1             22        Fuselage
41             1              1              0             0         Symmetry
42             4              4              1             10        Fuselage
43             4              4              1             9         Fuselage
44             4              4              1             14        Fuselage
45             4              4              1             23        Fuselage
46             4              4              1             19        Wing
47             4              4              1             20        Wing
48             4              4              1             27        Fairing
49             4              4              1             29        Fairing
50             4              4              1             28        Fairing
51             4              4              1             30        Fairing
```

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh



- Now we will look at the minimum set of user inputs needed in `fun3d.nml` to run this case

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

```
&project
  project_rootname = 'f6fx2b_trn'
/
&raw_grid
  grid_format = 'vgrid'
/
&reference_physical_properties
  mach_number       = 0.75
  reynolds_number   = 17705.40
  angle_of_attack   = 1.0
  temperature       = 580.0
  temperature_units = "Rankine"
/
&code_run_control
  restart_read = 'off'
  steps        = 500
/
&force_moment_integ_properties
  area_reference   = 72700.0
  x_moment_length = 141.2
  y_moment_length = 585.6
  x_moment_center = 157.9
  z_moment_center = -33.92
/
&nonlinear_solver_parameters
  schedule_cfl     = 10.0 200.0
  schedule_cflturb = 1.0 30.0
/
```

Project name

Read a set of VGRID files

Sets freestream Mach number
Sets Reynolds number
Sets freestream angle of attack
Sets freestream temperature
Uses Rankine temperature units for input

Perform a cold start
Perform 500 time steps

Sets reference area
Sets length for normalizing y-moments
Sets length for normalizing x-, z-moments
Sets x-moment center
Sets z-moment center

All in grid units

CFL for meanflow is ramped from 10.0 to 200.0
CFL for turbulence is ramped from 1.0 to 30.0

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- We now have the boundary conditions and input deck set up to run FUN3D
- To execute FUN3D, we use the following basic command line syntax:

```
mpirun ./nodet_mpi
```

  - Note your environment may require slightly different syntax:
    - `mpirun` vs `mpiexec` vs `aprun` vs …
    - May need to specify various MPI runtime options:
      - `-np #`
      - `-machinefile filename`
      - `-nolocal`
      - Others

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- Using 1 Intel Xeon Skylake node (40 cores), this case runs in ~1.5 minutes
- The top of the screen output will include an echo of your `fun3d.nml`, as well as some preprocessing information:

```
FUN3D 13.4-6cad5cb1ad Flow started 11/15/2018 at 10:49:07 with 40 processes
        [Echo of fun3d.nml]
 The default "unformatted" data format is being
   used for the grid format "vgrid".
... nsegments,ntet,nnodesg            1     2994053      513095
cell statistics: type,      min volume,      max volume, max face angle
cell statistics:  tet,  0.41152313E-06,  0.66593449E+11,  179.973678915
cell statistics:  all,  0.41152313E-06,  0.66593449E+11,  179.973678915

     ... PM (64,skip_do_min) :            0 F
     ... Calling ParMetis (ParMETIS_V3_PartKway) ....           0 F
     ... edgeCut       181874
     ... Time for ParMetis: .2 s
     ... Constructing partition node sets for level-0...               2994053 T
     ... Edge Partitioning ....
     ... Boundary partitioning....
     ... Reordering for cache efficiency....
     ... Write global grid information to f6fx2b.grid_info
     ... Time after preprocess TIME/Mem(MB):      1.60    180.52    180.52
  NOTE: kappa_umuscl set by grid: .00

 Grid read complete
  Repaired 82 nodes of symmetry plane 6662, max deviation: 0.172E-03
  y-symmetry metrics modified/examined: 23869/23869
  Distance_function unique ordering  T    20000000
  construct partial boundary...nloop=           1
  find closer surface edge...
  find closer surface face...

 Wall spacing: 0.766E-03 min, 0.120E-02 max, 0.115E-02 avg
```

FUN3D version, start time, job size

VGRID input is being used
Grid contains 2,994,053 tets and 513,095 points
Min/max cell volumes, max internal face angles

# of edges cut by partitioning (measure of communication)

1.6 secs required to preprocess the mesh

Min/max/avg wall spacing statistics

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- At this point, time stepping commences
- For each time step:
  - The L2-norm of the density|turbulence equation is red|blue; max and location are also included
  - Lift and drag are reported in green
- "`Done.`" indicates execution is complete

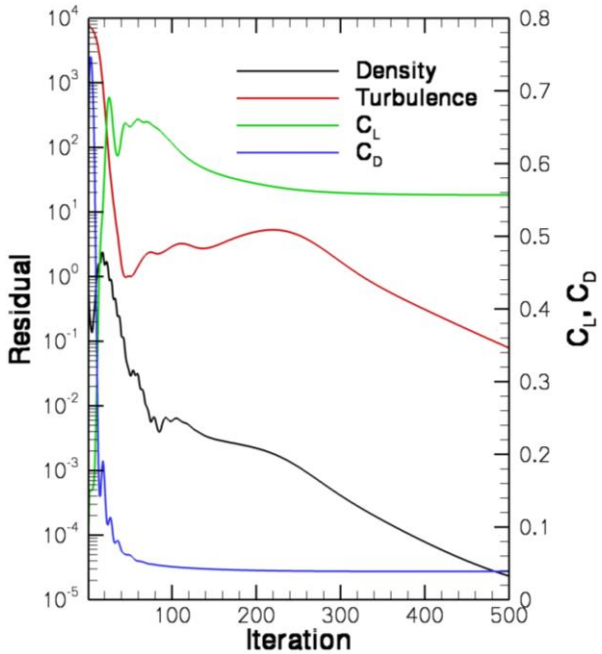
```
Iter            density_RMS  density_MAX   X-location   Y-location   Z-location
                   turb_RMS     turb_MAX   X-location   Y-location   Z-location
   1  0.567454200028342E+00  0.28035E+02  0.16377E+03 -0.16562E+03  0.20117E+02
      0.764159584889715E+04  0.13249E+07  0.79654E+04 -0.88280E+04  0.25675E+02
      Lift  0.103222595738744E+00         Drag  0.646514730726450E+00
   2  0.300676987599762E+00  0.12718E+02  0.29226E+03 -0.72487E+02 -0.12411E+02
      0.753354470094994E+04  0.12868E+07  0.79654E+04 -0.88280E+04  0.25675E+02
      Lift  0.146827898102870E+00         Drag  0.721246495313369E+00
.
.
.
 499  0.234725191084085E-04  0.46527E-02  0.63496E+04 -0.38199E+04  0.18712E+04
      0.801509219760057E-01  0.12900E+02  0.46732E+04 -0.15204E+04  0.26710E+03
      Lift  0.556613406060353E+00         Drag  0.388400267390697E-01
 500  0.232531701253396E-04  0.45872E-02  0.63496E+04 -0.38199E+04  0.18712E+04
      0.791050882262667E-01  0.12725E+02  0.46732E+04 -0.15204E+04  0.26710E+03
      Lift  0.556611115425931E+00         Drag  0.388398198948688E-01

Writing f6fx2b_trn.flow (version 12.2)
 inserting current history iterations 500
Time for write: .0 s

Done.
```

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- FUN3D provides a couple of text files with basic statistics and summary data:
  - `f6fx2b_trn.grid_info` File containing basic mesh statistics and partitioning info
  - `f6fx2b_trn.forces` File containing force breakdowns by boundary and totals
- FUN3D also produces:

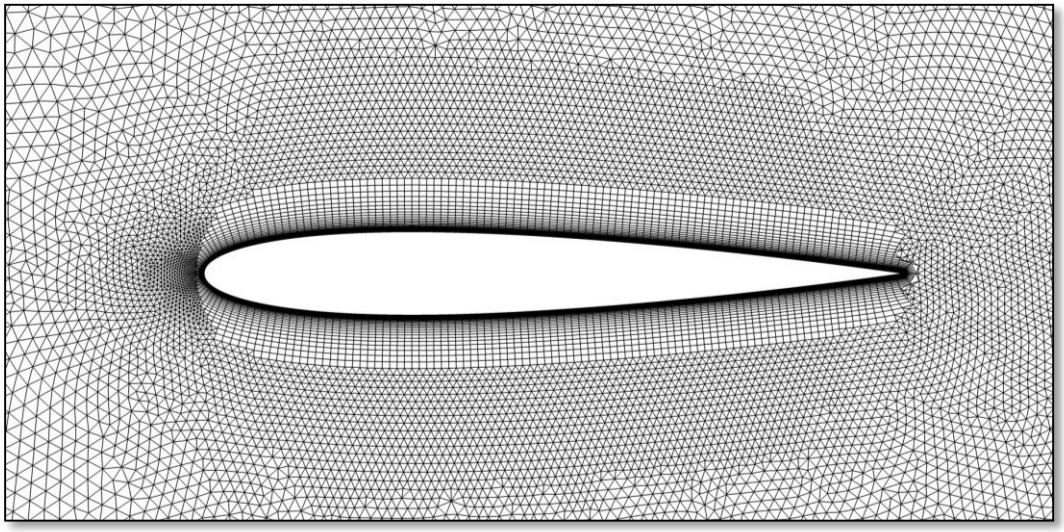
  `f6fx2b_trn_hist.dat` Tecplot file with residual, force convergence histories

  `f6fx2b_trn.flow` Solver restart information



- For this particular case, the mean flow and turbulence residuals are reduced by ~5 orders of magnitude over 500 time steps
- Lift and drag come in after a few hundred time steps

NACA 0012 Airfoil



- For this case, we have been given a set of binary, big endian AFLR3 files
  - 0012.b8.ugrid, 0012.mapbc
  - For computations in 2D mode
    - Grid must be one-element wide in the y-direction (except when using FUN2D format)
    - Grid must contain only prisms and/or hexes
- First check the .mapbc file
  - The y-planes must be separate boundary patches and should be given BC 6662

0012.mapbc

```
4
1  4000
2  5000
3  6662
4  6662
```

NACA 0012 Airfoil

- `fun3d.nml` is shown here
- FUN2D grid format will automatically be executed in 2D mode; all others must be explicitly put in 2D mode

```
&project
  project_rootname = '0012'
/
&raw_grid
  grid_format = 'aflr3'
  twod_mode   = .true.
/
&reference_physical_properties
  mach_number       = 0.80
  reynolds_number   = 1.e6
  angle_of_attack   = 1.25
  temperature       = 580.0
  temperature_units = "Rankine"
/
&code_run_control
  restart_read = 'off'
  steps        = 5000
/
&force_moment_integ_properties
  area_reference   = 0.1
  x_moment_center = 0.25
/
&nonlinear_solver_parameters
  schedule_cfl     = 10.0 200.0
  schedule_cflturb = 1.0 10.0
/
&global
  boundary_animation_freq = -1
/
```

Read an AFLR3 grid
Execute in 2D mode

NACA 0012 Airfoil

```
FUN3D 13.4-6cad5cb1ad Flow started 11/15/2018 at 16:45:59 with 40 processes
        [Echo of fun3d.nml]
 The default "stream" data format is being
   used for the grid format "aflr3".
Preparing to read binary AFLR3 grid: 0012.b8.ugrid

  nnodes              116862
  ntface,nqface       204510 14607
  ntet,npyr,nprz,nhex 0 0 102255 7047

cell statistics: type,      min volume,      max volume, max face angle
cell statistics:  prz,  0.16960303E-06,  0.52577508E-01,  164.861624007
cell statistics:  hex,  0.83173480E-09,  0.12843645E-04,  123.906431556
cell statistics:  all,  0.83173480E-09,  0.52577508E-01,  164.861624007

 ... PM (64,skip_do_min) :            0 F
     ... Calling ParMetis (ParMETIS_V3_PartKway) ....           0 F
     ... edgeCut        14428
     ... Time for ParMetis: .1 s
     ... checking for spanwise edge cuts.
     ... Constructing partition node sets for level-0...                109302 T
     ... Edge Partitioning ....
     ... Boundary partitioning....
     ... Euler numbers Grid:1 Boundary:0 Interior:0
     ... Reordering for cache efficiency....
     ... ordering edges for 2D.
     ... Write global grid information to 0012.grid_info
     ... Time after preprocess TIME/Mem(MB):      0.31     90.82     90.82
  NOTE: kappa_umuscl set by grid: .00

 Grid read complete

 Using 2D Mode (Node-Centered)

Distance_function unique ordering  T    20000000
  construct partial boundary...nloop=           1
  find closer surface edge...
  find closer surface face...
 Wall spacing: 0.100E-03 min, 0.100E-03 max, 0.100E-03 avg
```

Binary AFLR3 format is the default

Binary AFLR3 grid being read

Grid contains 116,862 points

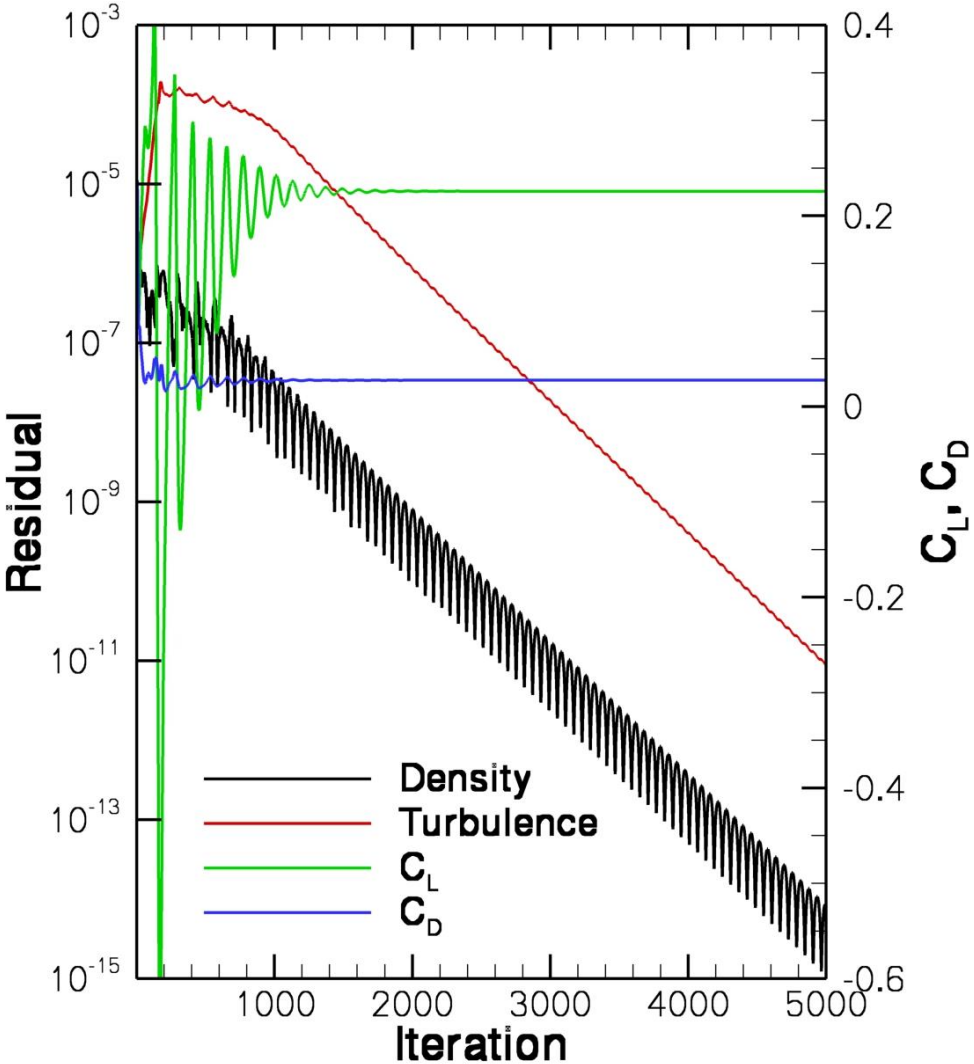
Grid contains 204,510 tris, 14,607 quads

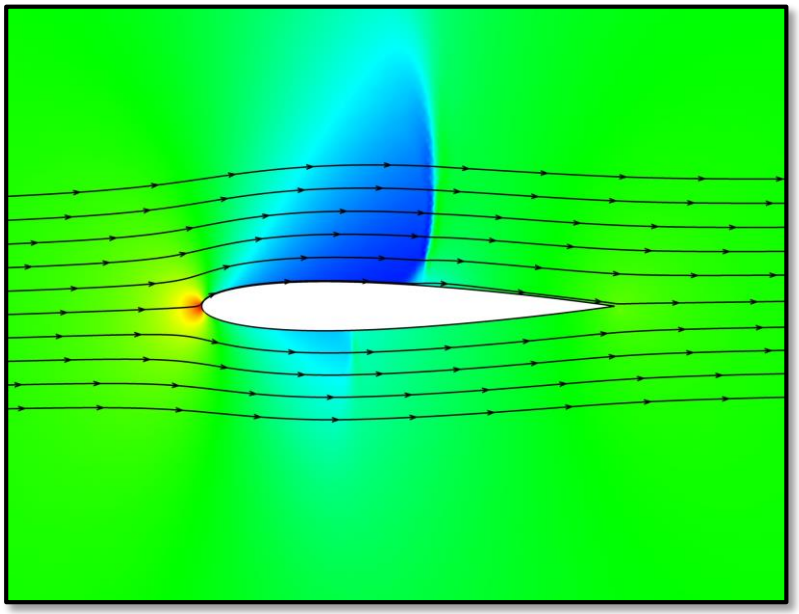
Grid contains 102,255 prisms, 7,047 hexes

Cell stats now broken out by cell type

Solver running in 2D mode

NACA 0012 Airfoil





List of Key Input/Output Files

- Input
  - Grid files (prefixed with project name, suffixes depend on grid format)
  - `fun3d.nml`
- Output
  - `[project].grid_info`
  - `[project].forces`
  - `[project]_hist.dat`
  - `[project].flow`

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 very 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/binary 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 endianness 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., nozzle plenum
  - Check your boundary conditions and gridding strategy
  - 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 `&special_parameters` namelist (viscous flux discretization degenerates 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 sometimes 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, any error output, and `config.log`
  - 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 manual for guidance
- Ask the FUN3D user community, *fun3d-users@lists.nasa.gov*

What We Learned

- Basic gridding requirements and file formats
- Runtime environment
- How to set up boundary conditions and very basic FUN3D input decks
- How to run a tetrahedral RANS solution for a wing-body VGRID mesh
- How to perform a 2D mixed element airfoil solution using an AFLR3 grid
- Some unhealthy things to watch for and possible remedies

Don’t hesitate to send questions our way!

fun3d-support@lists.nasa.gov