FUN3D v12.7 Training

Session 4:
Gridding, Solution, and Visualization 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
- Visualization 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
  - Hanging nodes are not currently supported
- 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
- 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)
- FUN3D can convert tetrahedral VGRID meshes to mixed elements
- FUN3D can convert any mixed element grid into tetrahedra using command line option '--make_tets'

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

NASA   http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D

5

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:
  - $\vec{x} = \vec{x}^* / (L_{ref}^* / L_{ref})$ $t = t^* a_{ref}^* / (L_{ref}^* / L_{ref})$ $t = t^* U_{ref}^* / (L_{ref}^* / L_{ref})$
    (compressible)   (incompressible)

NASA   http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D

6

Nondimensionalization (cont)

- For the *compressible flow* equations the dimensionless variables are:

$$-\vec{u} = \vec{u}^* / a^*_{ref} \qquad \text{so} \quad |\vec{u}|_{ref} = |\vec{u}|^*_{ref} / a^*_{ref} = M_{ref}$$

$$-P = P^* / (\rho^*_{ref} a^{*2}_{ref}) \quad \text{so} \quad P_{ref} = P^*_{ref} / (\rho^*_{ref} a^{*2}_{ref}) = 1/\gamma$$

$$-a = a^* / a^*_{ref} \qquad \text{so} \quad a_{ref} = 1$$

$$-T = T^* / T^*_{ref} \qquad \text{so} \quad T_{ref} = 1$$

$$-e = e^* / (\rho^*_{ref} a^{*2}_{ref}) \quad \text{so} \quad e_{ref} = e^*_{ref} / (\rho^*_{ref} a^{*2}_{ref}) = 1/(\gamma(\gamma-1)) + M^2_{ref}/2$$

$$-\rho = \rho^* / \rho^*_{ref} \qquad \text{so} \quad \rho_{ref} = 1$$

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

  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
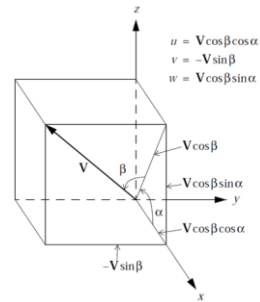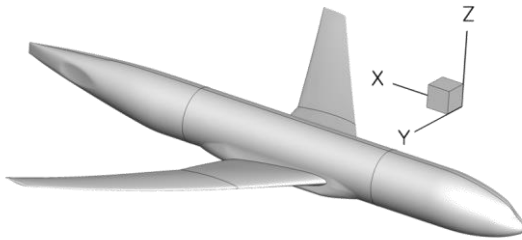
FUN3D Axis Convention

- FUN3D coordinate system differs from the standard wind coordinate system by a 180° 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

FUN3D Training Workshop
June 20-21, 2015
http://fun3d.larc.nasa.gov
9

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

FUN3D Training Workshop
June 20-21, 2015
http://fun3d.larc.nasa.gov
10

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
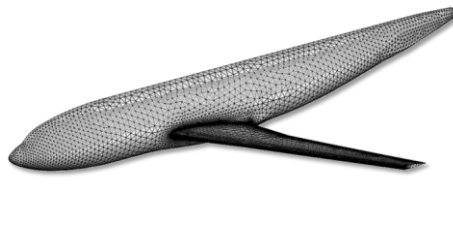- Most CLOs are for developer use; namelist options are preferred where available

NASA   http://fun3d.larc.nasa.gov   FUN3D Training Workshop June 20-21, 2015   **FUN3D**   11

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

NASA   http://fun3d.larc.nasa.gov   FUN3D Training Workshop June 20-21, 2015   **FUN3D**   12

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 BC's 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 BC's 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       1      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
```
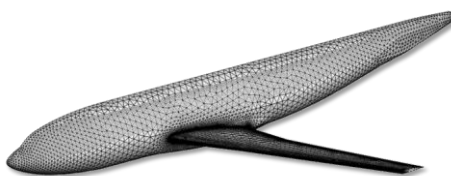
Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- 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'          Project name
/
&raw_grid
  grid_format = 'vgrid'                     Read a set of VGRID files
/
&reference_physical_properties
  mach_number       = 0.75                  Sets freestream Mach number
  reynolds_number   = 17705.40              Sets Reynolds number
  angle_of_attack   = 1.0                   Sets freestream angle of attack
  temperature       = 580.0                 Sets freestream temperature
  temperature_units = "Rankine"             Uses Rankine temperature units for input
/
&code_run_control
  restart_read = 'off'                      Perform a cold start
  steps        = 500                        Perform 500 time steps
/
&force_moment_integ_properties
  area_reference  = 72700.0                 Sets reference area
  x_moment_length = 141.2                   Sets length for normalizing y-moments      All in
  y_moment_length = 585.6                   Sets length for normalizing x-, z-moments  grid units
  x_moment_center = 157.9                   Sets x-moment center
  z_moment_center = -33.92                  Sets z-moment center
/
&nonlinear_solver_parameters
  schedule_cfl     = 10.0 200.0             CFL for meanflow is ramped from 10.0 to 200.0
  schedule_cflturb = 1.0 30.0               CFL for turbulence is ramped from 1.0 to 30.0
/
```

NASA   http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D   15

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

NASA   http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D   16

Transonic Turbulent Flow on a Tetrahedral Wing-Body Mesh

- Using 1 Intel Haswell node (24 cores), this case runs in 2-3 minutes
- The top of the screen output will include an echo of your `fun3d.nml`, as well as some preprocessing information:

```
FUN3D 12.7-74063 Flow started 05/18/2015 at 06:09:15 with 24 processes        FUN3D version, start time, job size
        [Echo of fun3d.nml]
 The default "unformatted" data format is being
  used for the grid format "vgrid".                                           VGRID input is being used
... nsegments,ntet,nnodesg      1    2994053    513095                        Grid contains 2,994,053 tets and 513,095 points
 cell statistics: type,     min volume,       max volume, max face angle      Min/max cell volumes, max internal face angles
 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      140453                                                       # of edges cut by partitioning (measure of communication)
    ... 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      1.6 secs required to preprocess the mesh
 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: 23601/23601
 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                    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  0.567457404772944E+00  0.28035E+02  0.16377E+03 -0.16562E+03  0.20117E+02
       0.764159584901413E+04  0.13249E+07  0.79654E+04 -0.88280E+04  0.25675E+02
       Lift  0.103226565173772E+00         Drag  0.646513396068887E+00
    2  0.300679598726331E+00  0.12718E+02  0.29226E+03 -0.72487E+02 -0.12411E+02
       0.753354470463467E+04  0.12868E+07  0.79654E+04 -0.88280E+04  0.25675E+02
       Lift  0.146829230859457E+00         Drag  0.721243167013704E+00
     .
     .
     .
  999  0.383370843514542E-05  0.13909E-03  0.35380E+03 -0.58429E+02 -0.16200E+02
       0.318320572426105E-02  0.19891E+00  0.36848E+03 -0.68458E+02  0.31074E+01
       Lift  0.556387990643583E+00         Drag  0.388233647462313E-01
 1000  0.382497896407724E-05  0.13871E-03  0.35380E+03 -0.58429E+02 -0.16200E+02
       0.317436044959994E-02  0.19835E+00  0.36848E+03 -0.68458E+02  0.31074E+01
       Lift  0.556387923023456E+00         Drag  0.388233658091165E-01

 Writing f6fx2b.flow (version 11.8) lmpi_io 2
  inserting current history iterations 1000
 Time for write: .1 s
 Done.
```

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
       0.764159584901741E+04  0.13249E+07  0.79654E+04 -0.88280E+04  0.25675E+02
       Lift  0.103222129717669E+00         Drag  0.646514468368827E+00
    2  0.300676687726037E+00  0.12718E+02  0.29226E+03 -0.72487E+02 -0.12411E+02
       0.753354469872627E+04  0.12868E+07  0.79654E+04 -0.88280E+04  0.25675E+02
       Lift  0.146830367737086E+00         Drag  0.721243419758588E+00
    .
    .
    .
  499  0.235098406158263E-04  0.44827E-02  0.63496E+04 -0.38199E+04  0.18712E+04
       0.799698877237297E-01  0.12961E+02  0.46732E+04 -0.15204E+04  0.26710E+03
       Lift  0.556610229549889E+00         Drag  0.388376897833650E-01
  500  0.232908407834686E-04  0.44201E-02  0.63496E+04 -0.38199E+04  0.18712E+04
       0.789246351974423E-01  0.12785E+02  0.46732E+04 -0.15204E+04  0.26710E+03
       Lift  0.556607946389416E+00         Drag  0.388374809483346E-01

Writing f6fx2b_trn.flow (version 11.8) lmpi_io 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
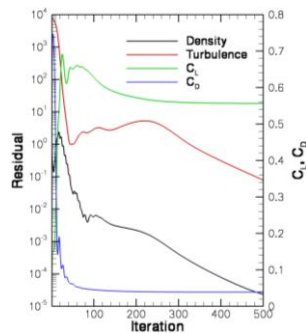
- For this particular case, the mean flow and turbulence residuals are reduced by ~5 orders of magnitude over 500 time steps
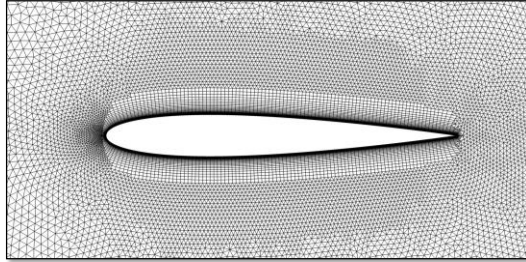- Lift and drag come in after a few hundred time steps

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'          Read an AFLR3 grid
  twod_mode   = .true.           Execute in 2D mode
/
&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
/
```

NACA 0012 Airfoil

FUN3D 12.7-74063 Flow started 05/18/2015 at 09:06:46 with 24 processes
     [Echo of fun3d.nml]
 The default "stream" data format is being   **Binary AFLR3 format is the default**
   used for the grid format "aflr3".
Preparing to read binary AFLR3 grid: 0012.b8.ugrid   **Binary AFLR3 grid being read**

   nnodes   116862   **Grid contains 116,862 points**
   ntface,nqface   204510 14607   **Grid contains 204,510 tris, 14,607 quads**
   ntet,npyr,nprz,nhex 0 0 102255 7047   **Grid contains 102,255 prisms, 7,047 hexes**

 cell statistics: type,   min volume,   max volume, max face angle   **Cell stats now broken out by cell type**
 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   11490
     ... 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)   **Solver running in 2D mode**

 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

NACA 0012 Airfoil
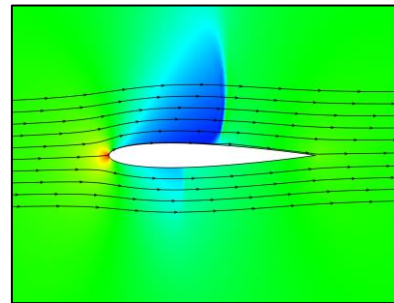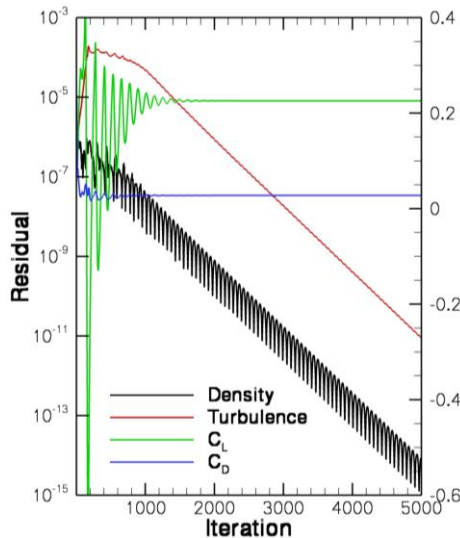
List of Key Input/Output Files

- 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`

FUN3D Training Workshop
June 20-21, 2015

FUN3D   25

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

FUN3D Training Workshop
June 20-21, 2015

FUN3D   26

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?

NASA http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D 27

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
  - 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

NASA http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D 28

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*

NASA   http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D   29

Visualization Learning Goals

- What this will teach you
    - Run-time flow visualization output
        - 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
    - The plethora of output options available for visualization
    - Tecplot usage
- What should you already know
    - Basic flow solver operation and control

NASA   http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D   30

Background

- Datasets are getting simply too large to post-process in a traditional manner
- FUN3D allows visualization data to be generated as the solver is running
    - User specified frequency and output type
    - User specified output variables from a fairly extensive list
- Majority of output options are Tecplot-based
    - Volume output may also be generated in Fieldview, CGNS formats
- Note FUN3D also supports true in-situ visualization at scale using the DoE VisIt package; however, this is not covered here
    - Intelligent Light is currently integrating VisIt's in-situ capabilities with Fieldview

Selected Visualization Output Examples

Iso-surfaces

Schlieren, boundary output
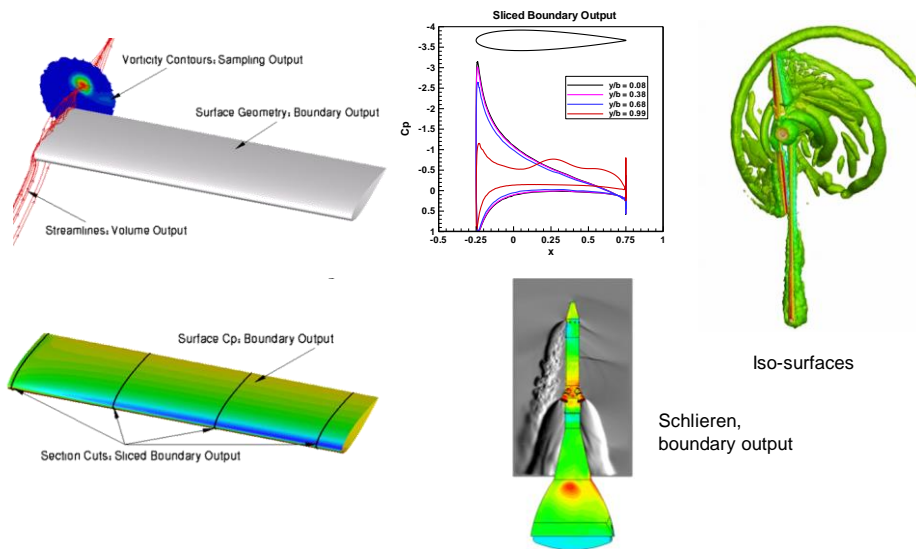
Visualization Overview

- All of the visualization outputs require similar namelist-specified "frequency" N to activate:
    - 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 is ignored
        - N > 0 generates output every Nth time step - typically used to generate animation for unsteady flows; can also be used to observe how a steady flow converges

FUN3D Training Workshop
June 20-21, 2015

33

Visualization 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 manual 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"

FUN3D Training Workshop
June 20-21, 2015

34

Visualization Overview

- For boundary surface output, default is all solid boundaries in 3D and one y=const plane in 2D; alternate output boundaries 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**
  - You must have a restart file (**[project].flow**)
  - Run the solver with the appropriate namelist input to get desired output
  - **[project].flow** will remain unaltered after completion

NASA   http://fun3d.larc.nasa.gov   FUN3D Training Workshop
June 20-21, 2015   FUN3D   35

Visualization Overview

- Sampling output requires additional data to describe the desired sampling surface(s)
  - Specified in namelist **&sampling_parameters**
  - Surfaces may be planes, quadrilaterals or circles of arbitrary orientation, or may be spheres or boxes
  - Isosurfaces and schlierens also available
  - Points may also be sampled
  - See manual for complete info
- Sliced boundary surface output requires additional data to describe the desired slice section(s)
  - Specified in namelist **&slice_data**
  - Always / only outputs x, y, z, C_p, C_{fx}, C_{fy}, C_{fz}
  - User specifies which (solid) boundaries to slice, and where
  - See manual for complete info

NASA   http://fun3d.larc.nasa.gov   FUN3D Training Workshop
June 20-21, 2015   FUN3D   36

Visualization Overview

- Output files will be ASCII unless you have built 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

Boundary Output Visualization Example

```
&global
  boundary_animation_freq = -1        Dump boundary vis at end of run
/
&boundary_output_variables
  primitive_variables = .false.       Turn off rho, u, v, w, p
  cp                  = .true.        Turn on Cp
  yplus               = .true.        Turn on y+
/
```
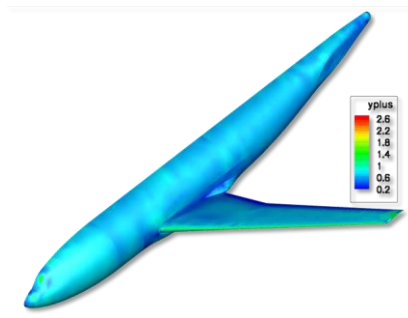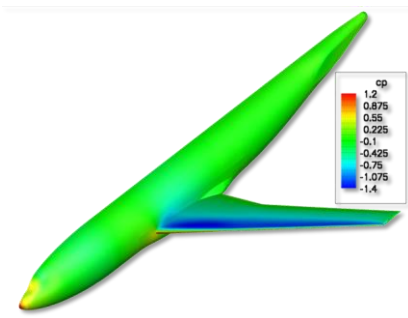
Sampling Visualization Example

```
&sampling_parameters
  number_of_geometries  = 3                Want 3 sampling geometries
  type_of_geometry(1)   = 'plane'          First geometry is a plane
  plane_center(2,1)     = -234.243         Plane y-coordinate
  plane_normal(2,1)     = 1.0              Plane y-normal
  sampling_frequency(1) = -1               Write at end of run
  type_of_geometry(2)   = 'sphere'         Second geometry is a sphere
  sphere_center(1,2)    = 74.9             Center x-coordinate
  sphere_center(2,2)    = -107.7           Center y-coordinate
  sphere_center(3,2)    = 50.0             Center z-coordinate
  sphere_radius(2)      = 20.0             Sphere radius
  sampling_frequency(2) = -1               Write at end of run
  type_of_geometry(3)   = 'isosurface'     Third geometry is an isosurface
  isosurf_variable(3)   = 'mach'           Isosurface will be based on Mach number
  isosurf_value(3)      = 1.00             Isosurface defined by Mach=1
  sampling_frequency(3) = -1               Write at end of run
/
&sampling_output_variables
  primitive_variables = .false.            Turn off rho, u, v, w, p
  mach                = .true.             Turn on Mach number
/
```
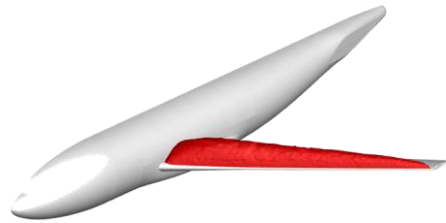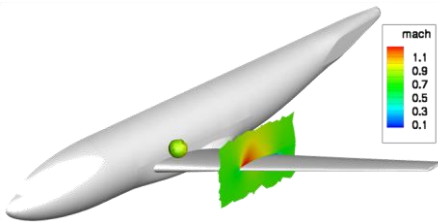
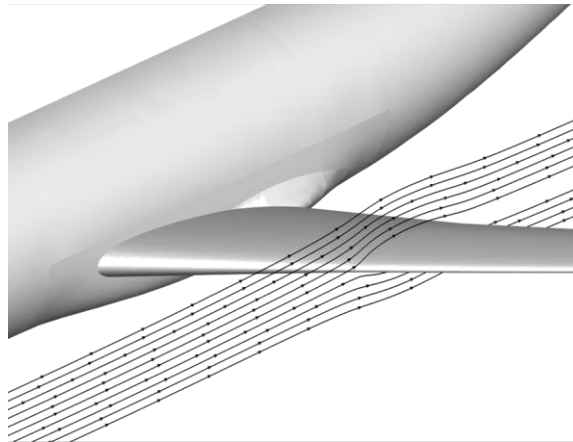# Volume Visualization Example

```
&global
  volume_animation_freq = -1     Dump output at end of run
/
&volume_output_variables
  export_to='tecplot'            Send results to Tecplot file
/
```

Slicing Visualization Example

```
&global
  slice_freq = -1          Dump output at end of run
/
&slice_data
  nslices = 1              Perform one slice
  slice_location(1) = -234.243   Coordinate of slice
/
```
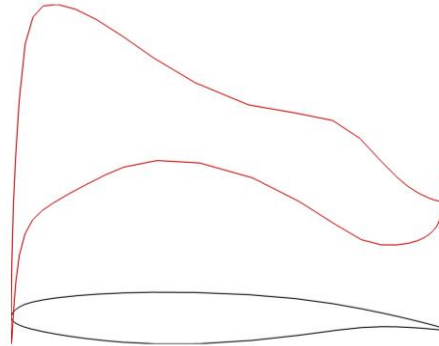
Troubleshooting/FAQ

- I can see what look like ragged dark lines on sampling surfaces and volume data – what is that?
  - 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
- When I dump out volume plot files in Tecplot format, I get a file for every processor – is there a way around this?
  - Not currently. However, Tecplot can be easily told to load all of the files at once without having to individually select them all.
  - The team is working with Tecplot to develop their next generation of I/O API's, with special focus on massively parallel needs
  - Alternative: switch to Fieldview or CGNS output, which uses a single file

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
- Overview of visualization output options and examples

Don't hesitate to send questions our way!
fun3d-support@lists.nasa.gov

NASA   http://fun3d.larc.nasa.gov

FUN3D Training Workshop
June 20-21, 2015

FUN3D   43