FUN3D v14.0 Training
6-DOF Simulations

Eli Shellabarger

Learning Goals

What we will cover

- Background & Capabilities
- Acquisition & Installation
- Inputs & Outputs
- Order of Operations & Demo Cases
- Considerations & Failure Modes

What we will not cover

- Best practices of time-accurate and dynamic-mesh solver operation
- Case flow solver initialization (pre-restart solution)
- Building the external 6-DOF library
- Stand-alone use of the 6-DOF library
- External force and moment specification (see July 2010 training)

Background

- 6-DOF Solver
  - "Libmo" 6-DOF library developed by Koomullil & Prewitt [1]
  - Originally implemented to FUN3D in 2000s [2]
    - Only one 6-DOF example existed
    - Limited demonstration of capability due to computational limitations (Used external F&M to drive 6-DOF integration)
  - Recently "dusted-off" after minimal utilization

- Updates since July 2010 Training
  - Various changes and bug fixes (see release history: 2010, 2014, 2019)
  - Improved documentation in v14.0 user manual
  - Provide additional example cases
  - Identify common failure modes, problems, and their solutions

[1] Roy P. Koomullil and Nathan C. Prewitt. A Library Based Approach for Rigid Body Dynamics Simulation. In *18th AIAA Computational Fluid Dynamics Conference*, Miami, FL, June 2007.
[2] Robert T. Biedron and James L. Thomas. Recent Enhancements to the FUN3D Flow Solver for Moving-Mesh Applications. In *47th AIAA Aerospace Sciences Meeting*, Orlando, FL, January 2009.

Capabilities & Requirements

- Enables one to six degrees of freedom for rigid body motion
  - Converts aerodynamic forces and moments to linear and angular accelerations
  - Independent gravitational acceleration
  - Option to add external forces and moments read from files

- Supports variety of mesh motion strategies
  - Frequently used with static and overset mesh motion options
  - Mesh deformation is untested, but no reason to suspect it doesn't work

- Requires unsteady flow solver
  - Motion is within the inertial frame of flow solver,
    i.e., constant freestream conditions

Acquisition and Installation

- Acquiring the external 6-DOF library
  - See manual for most up-to-date information
  - As of this training (April 2023):
    Interested parties may reach out to
    Nathan Prewitt at Nathan.C.Prewitt@erdc.dren.mil

- Building FUN3D with the external 6-DOF Library
  - Library specific questions to be directed to maintainer(s)
  - When building FUN3D, only need to add one option at compile time:
    `--with-sixdof=</path/to/sixdof>`
  - Where `</path/to/sixdof>` is the 6-DOF installation directory

Input Files (1/4)

Input Files

- Solver Namelists
  - `fun3d.nml`
  - `moving_body.input`
- Grid Files, e.g.,
  - `{project}.mapbc`
  - `{project}.*b8.ugrid`
- Restart file (flow initialization):
  - `{project}.flow`

fun3d.nml

```
&nonlinear_solver_parameters
...{unsteady case options}...
/

&global
  moving_grid = .true.
/
```

Flow Initialization:
- Temporal boundary condition to IVP
- Extremely case dependent
- Result is only as good as the input (model and I.C.'s)
- Same can be said for moving body I.C.'s…

Training Resources from Dec 2018:
- Time-Dependent Simulations
- Dynamic Grid Simulations

Input Files (2/4)

moving_body.input

```
&body_definitions
  n_moving_bodies = 1                ! Number of moving bodies in motion
  extra_digits_body_state = .false.  ! Output body state with 15 decimal digits rather than 5
  dimensional_output = .false.       ! Write body state history as dimensional data

  ! Quantities to make forces and moments dimensional in desired unit system (required for 6-DOF):
  ref_velocity = 1117.0    ! Reference velocity, in consistent units with case
  ref_density  = 0.002378  ! Reference density, in consistent units with case
  ref_length   = 1.00      ! NOT THE REFERENCE LENGTH, actually the grid scaling parameter: L*ref/Lref

  body_name(1) = 'name'          ! Name of moving body (1)
  n_defining_bndry(1) = 2        ! Number of boundaries defining this body number
  defining_bndry(:,1) = 0        ! Define boundary numbers which compose body (1)
  motion_driver(1) = '6dof'      ! Use the UAB "Libmo" 6-DOF library to drive body motion
  mesh_movement(1) = 'rigid'     ! Use rigid mesh motion

  x_mc(1)   = 0.0  ! X-coord of MRC at t=0 of Body (1)
  y_mc(1)   = 0.0  ! Y-coord of MRC at t=0 of Body (1)
  z_mc(1)   = 0.0  ! Z-coord of MRC at t=0 of Body (1)
  s_ref(1) = 1.0   ! Nondim. ref. area of Body (1), defaults to area_reference
  c_ref(1) = 1.0   ! Nondim. ref. chord length of Body (1), defaults to x_moment_length
  b_ref(1) = 1.0   ! Nondim. ref. span length of Body (1),  defaults to y_moment_length

  move_mc(1) = 1   ! 0: leave MRC fixed in space, 1 (default): move MRC following Body (1)
/
```

As defined in fun3d.nml, **&force_moment_integ_properties**

Input Files (3/4)

moving_body.input (continued)

```
&sixdof_motion
  ! Define body mass properties:
  mass(1) = 1.0 ! Mass of Body (1)
  cg_x(1) = 0.0 ! CG x-coordinate of Body (1)
  cg_y(1) = 0.0 ! CG y-coordinate of Body (1)
  cg_z(1) = 0.0 ! CG z-coordinate of Body (1)
  i_xx(1) = 1.0 ! Ixx moment of inertia of Body (1)
  i_yy(1) = 1.0 ! Iyy moment of inertia of Body (1)
  i_zz(1) = 1.0 ! Izz moment of inertia of Body (1)
  i_xy(1) = 0.0 ! Ixy product of inertia of Body (1)
  i_xz(1) = 0.0 ! Ixz product of inertia of Body (1)
  i_yz(1) = 0.0 ! Iyz product of inertia of Body (1)

  ! Define body initial conditions
  body_lin_vel(:,:) = 0.0 ! Initial linear velocity  (x,y,z components) of Body (1)
  body_ang_vel(:,:) = 0.0 ! Initial angular velocity (p,q,r components) of Body (1)
  euler_ang(1,:)    = 0.0 ! Initial euler angle, yaw,   of Body (1)
  euler_ang(2,:)    = 0.0 ! Initial euler angle, pitch, of Body (1)
  euler_ang(3,:)    = 0.0 ! Initial euler angle, roll,  of Body (1)

  ...(see next slide)...
/
```

Setting `euler_ang(:,:)` in `&sixdof_motion` appears to be overridden and reset to zero as of v14.0 (suspect a conflicting default), but unclear how far back the issue goes. Static grid transformations can be used to place a starting orientation – just be sure to verify your body axes are what you intend!

Depending on the case in question, there may also be more than one way to achieve a desired set of initial conditions.

Input Files (4/4)

moving_body.input (continued)

```
&sixdof_motion
  ...(see last slide)...

  ! Gravitational Acceleration: always on, turn off by setting gravity_mag = 0
  gravity_dir(1:3) = 0.0, 0.0, -1.0 ! Define direction (x,y,z) of gravitational acceleration
  gravity_mag = 32.2                ! Define magnitude of gravitational acceleration

  ! External F&M: See July 2010 Training Store-Sep example for more.
  n_extforce(1) = 0         ! Define number of external forces applied to Body (1), excluding gravity
  n_extmoment(1) = 0        ! Define number of external moments applied to Body (1)
  file_extforce(:,1) = ''   ! Define file(s) containing external forces vs time data for Body (1)
  file_extmoment(:,1) = '' ! Define file(s) containing external moments vs time data for Body (1)

  ! Select degrees of freedom to neglect (full 6-DOF = all false; gravity only = all true):
  ignore_x_aeroforce(:)   = .false.
  ignore_y_aeroforce(:)   = .false.
  ignore_z_aeroforce(:)   = .false.
  ignore_x_aeromoment(:) = .false.
  ignore_y_aeromoment(:) = .false.
  ignore_z_aeromoment(:) = .false.

  ! Other Options:
  use_specified_aero_data = .false.      ! Use F&M read from file(s) instead of computed values
  print_sixdof_summary = .false.         ! Print 6-DOF summary, for each body, for each timestep
  use_restart_mass_properties = .false. ! Reuse mass properties from restart
/
```

Output Files

Time History Output Files for 6-DOF Solutions:

- PositionBody_*.dat & VelocityBody_*.dat
  - For each body number
  - Inertial and Body frames
- AeroForceMomentBody_*.dat
  - For each body number
  - Inertial, Wind, and Body frames

Recommended to use **`print_sixdof_summary`** in **`&sixdof_motion`** namelist for additional useful output in screen output (defaults to false)

- Easiest way to catch a problem early

Typical 6-DOF Run Directory

```
[user@host sixdof]$ ls
6dof_case.o1234
AeroForceMomentBody_1.dat
AeroForceMomentBody_1_body_frame.dat
AeroForceMomentBody_1_wind_frame.dat
PositionBody_1.dat
PositionBody_1_body_frame.dat
VelocityBody_1.dat
VelocityBody_1_body_frame.dat
fun3d.nml
screen.out
moving_body.input
run_sixdof.pbs
project.flow
project.forces
project.grid_info
project.lb8.ugrid
project.mapbc
project.mapbc.override
project_geom_cons_law.dat
project_hist.dat
project_subhist.dat
project_tec_boundary_timestep0.szplt
project_tec_boundary_timestep100.szplt
```

Order of Operations & Demo Cases

General Order of Operations:

- Initialization Case
- Moving Body Case

Demo Cases:

1) Inviscid Falling Sphere
2) Viscous Falling Sphere

3a) Planar NACA-0012

3b) …with initial static grid transformation

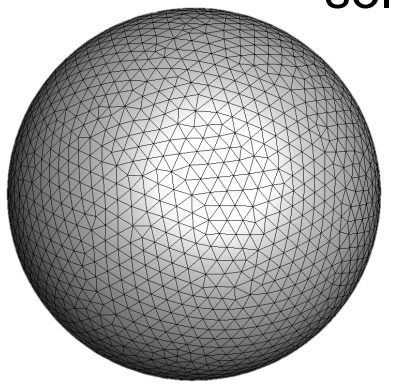
Disclaimer

Demonstration cases are meant to give examples of using the 6-DOF library and should not necessarily be considered the best practices for any given flow solution.

The rigid body motion product of 6-DOF solutions is an integration of body accelerations resulting from unsteady force and moment solutions, and by nature is very sensitive to the quality of inputs.

Demo 1: Inviscid Falling Sphere 2-DOF (1/2)

- Basic accelerating body case
  - “Gravity check”
  - Aero forces ~ zero
- Requires restart from initialized flow solution
  - Can be obtained from similar namelist
  - Example uses steady solution to initialize



invis-sphere.mapbc

```
      8
1       5000    farfield
2       5000    farfield
3       5000    farfield
4       5000    farfield
5       3000    sphere
6       3000    sphere
7       3000    sphere
8       3000    sphere
```

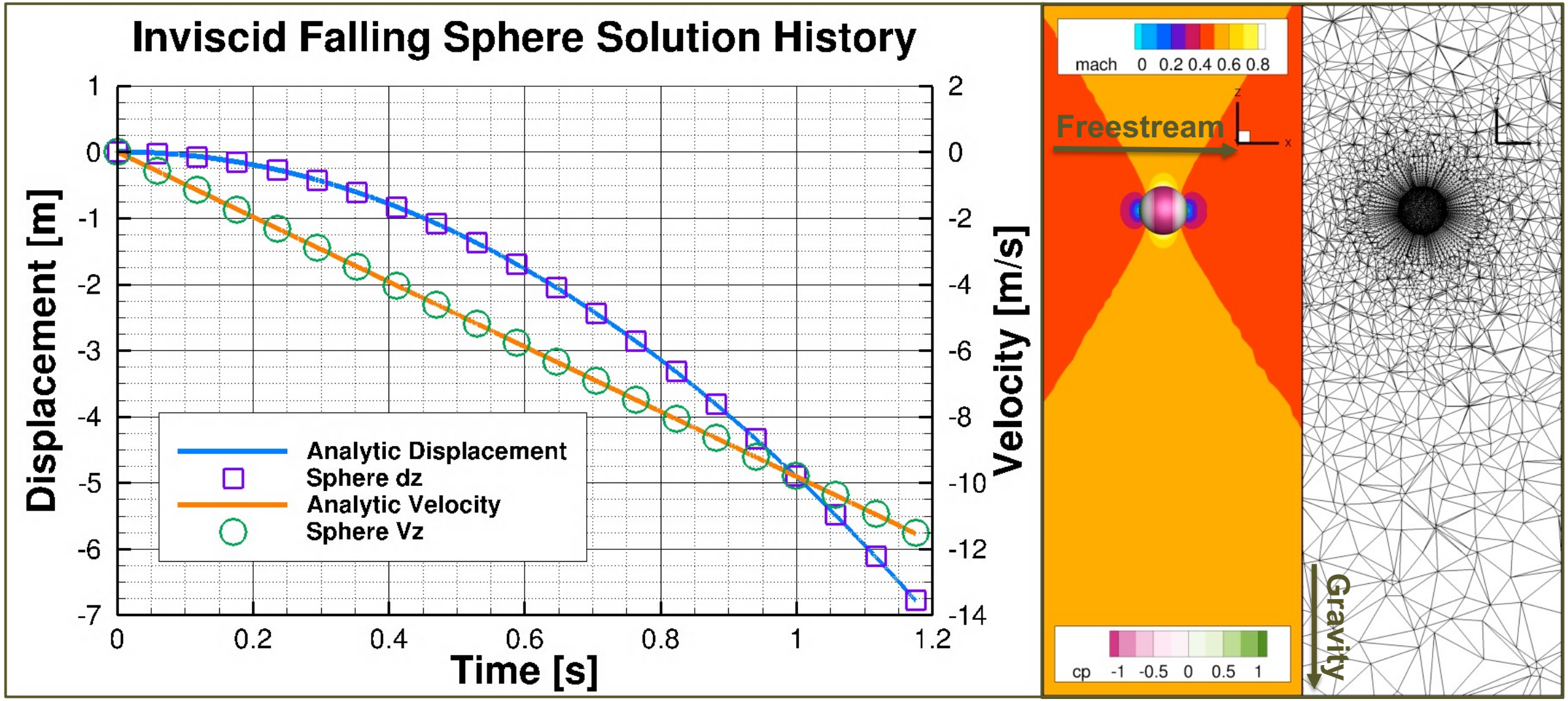
fun3d.nml

```
&project
  project_rootname = 'invis-sphere'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
  patch_lumping = 'family'
/
&force_moment_integ_properties
  area_reference  =  0.7854 ! pi*(D/2)^2
  x_moment_length =  1.0000 ! D
  y_moment_length =  1.0000 ! D
  x_moment_center =  0.0000 ! x_cg
  y_moment_center =  0.0000 ! y_cg
  z_moment_center =  0.0000 ! z_cg
/
&governing_equations
  eqn_type      = 'compressible'
  viscous_terms = 'inviscid'
/
&reference_physical_properties
  dim_input_type    = 'nondimensional'
  mach_number       = 0.5
  temperature       = 288.15
  temperature_units = 'Kelvin'
  angle_of_attack   = 0.0
  angle_of_yaw      = 0.000
/
&inviscid_flux_method
  flux_construction     = 'roe'
/
&code_run_control
  steps                 = 2000
  restart_write_freq    = 1000
  restart_read          = 'on_nohistorykept'
/
&nonlinear_solver_parameters
  time_accuracy           = '1storder'
  time_step_nondim        = 0.2
  subiterations           = 3
  schedule_iteration(1:2) = 1, 3
  schedule_cfl(1:2)       = 25.0, 25.0
  schedule_cflturb(1:2)   = 10.0, 10.0
/
&global
  moving_grid             = .true.
/
```

moving_body.input

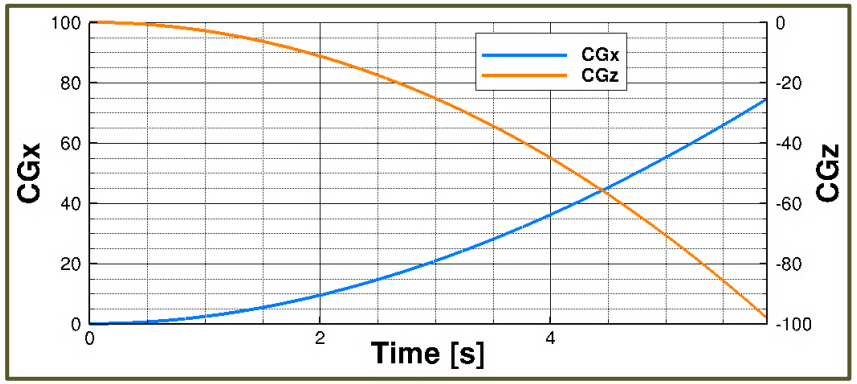
```
&body_definitions
  n_moving_bodies = 1
  relative_velocity_scaling = .false.
  dimensional_output = .false.
! Reference Dimensionalizing Properties:
  ref_velocity = 340.294
  ref_density = 1.2250
  ref_length = 1.0
! Define body:
  body_name(1) = 'sphere'
  parent_name(1) = ''
  n_defining_bndry(1) = 1
  defining_bndry(1,1) = 2
  motion_driver(1) = '6dof'
  mesh_movement = 'rigid'
  x_mc(1)  = 0.0000     ! x_mrc = x_cg
  y_mc(1)  = 0.0000     ! y_mrc = y_cg
  z_mc(1)  = 0.0000     ! z_mrc = z_cg
  s_ref(1) = 0.7854     ! A = pi*(D/2)^2
  c_ref(1) = 1.0000     ! D
  b_ref(1) = 1.0000     ! D
  move_mc(1) = 1
/
&sixdof_motion
  mass(1) =500.00       ! Body 1 mass
  cg_x(1) =  0.00       ! Body 1 x_cg
  cg_y(1) =  0.00       ! Body 1 y_cg
  cg_z(1) =  0.00       ! Body 1 z_cg
  i_xx(1) = 50.00       ! I_xx,sphere = (2/5)*m*(D/2)^2
  i_yy(1) = 50.00       ! I_yy,sphere = (2/5)*m*(D/2)^2
  i_zz(1) = 50.00       ! I_zz,sphere = (2/5)*m*(D/2)^2
  i_xy(1) =  0.00       ! I_xy,sphere = 0.00
  i_xz(1) =  0.00       ! I_xz,sphere = 0.00
  i_yz(1) =  0.00       ! I_yz,sphere = 0.00
  body_lin_vel(1:3,1) = 0., 0., 0.
  body_ang_vel(1:3,1) = 0., 0., 0.
  euler_ang(1:3,1) = 0.0, 0.0, 0.0
  gravity_dir(1:3) = 0.0, 0.0, -1.0
  gravity_mag = 9.81
  ignore_x_aeroforce(1)   = .false.
  ignore_y_aeroforce(1)   = .true.
  ignore_z_aeroforce(1)   = .false.
  ignore_x_aeromoment(1) = .true.
  ignore_y_aeromoment(1) = .true.
  ignore_z_aeromoment(1) = .true.
  print_sixdof_summary = .true.
/
```

Demo 1: Inviscid Falling Sphere (2/2)

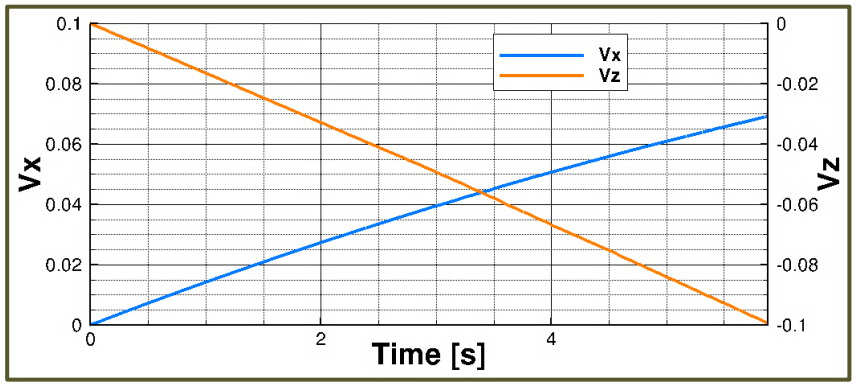


Demo 2: Viscous Falling Sphere 2-DOF

- Simple modification of inviscid sphere
  - Change to viscous solver
  - Requires viscous grid



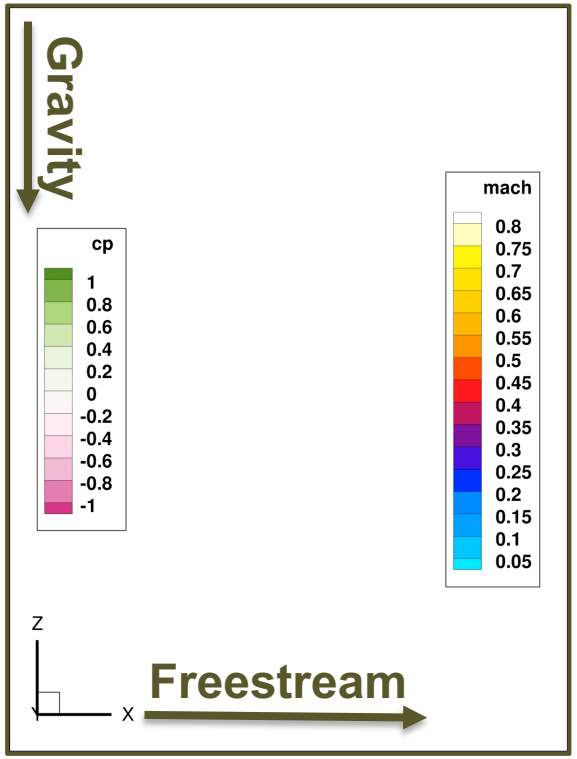
Nondimensonal Output of Position is in Grid Units



Nondimensonal Output of Velocity follows Model Conventions

fun3d.nml

```
&governing_equations
  ...
  viscous_terms = 'turbulent'
  ...
/
&turbulent_diffusion_models
  turbulence_model       = 'sa'
/
&reference_physical_properties
  ...
  reynolds_number = 1e7
  ...
/
```

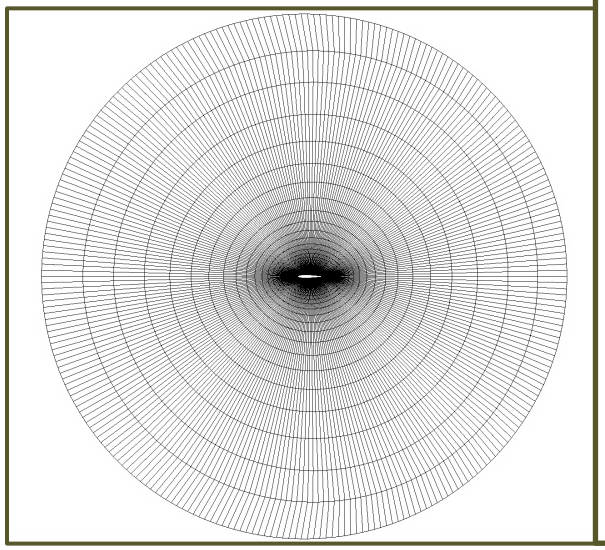


Solver Screen Output w/ sixdof summary true

```
*********************
*  Problem Summary  *
*********************
  Library Defaults
    Gravity On
    Gravity Magnitude: 9.810000
    Gravity Direction: 0.000000, 0.000000, -1.000000
    Time = 4.390909
    Time Step = 0.000588
**************************************
  Body 1 - sphere
**************************************

    Mass: 500.000000
    Center Of Gravity:
      CGx = 0.000000 CGy = 0.000000 CGz = 0.000000
    Moments Of Inertia:
      Ixx = 50.000000, Iyy = 50.000000, Izz = 50.000000
    Products Of Inertia:
      Ixy = Iyx = 0.000000
      Iyz = Izy = 0.000000
      Izx = Ixz = 0.000000

    Total Applied Forces:
      Fx = 1791.470770 Fy = 0.000000 Fz = 2163.671256
    Total Applied Moments:
      Mx = 0.000000 My = 0.000000 Mz = 0.000000

    Linear Velocity:
      u = 18.603576  v = 0.000000 w = -24.985485
    Angular Velocity:
      p = 0.000000 q = 0.000000 r = 0.000000

    Transformation Matrix:
      [ 1.000000e+00, 0.000000e+00, 0.000000e+00, 4.315152e+01 ]
      [ 0.000000e+00, 1.000000e+00, 0.000000e+00, 0.000000e+00 ]
      [ 0.000000e+00, 0.000000e+00, 1.000000e+00, -5.429049e+01 ]
      [ 0.000000e+00, 0.000000e+00, 0.000000e+00, 1.000000e+00 ]

    Quaternion:
      e0 = 1.000000 e1,e2,e3 = 0.000000, 0.000000, 0.000000
    Euler Angles:
      yaw = 0.000000, pitch = 0.000000, roll = 0.000000

    Linear Displacements:
      dx =    43.1515      dy =    0.00000      dz =  -54.2905

    Checks of body 1 motion data extracted from transform matrix:
      yaw, pitch, roll    0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
      u, v, w             0.186035752286E+02  0.000000000000E+00 -0.249854842169E+02
      p, q, r             0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
```

Demo 3a: Planar NACA0012 3-DOF (1/2)

- Infinite wing case
  - Unit span section
  - Nose Heavy
  - 5deg AoA
- Unconstrained planar motion with planar flow solution



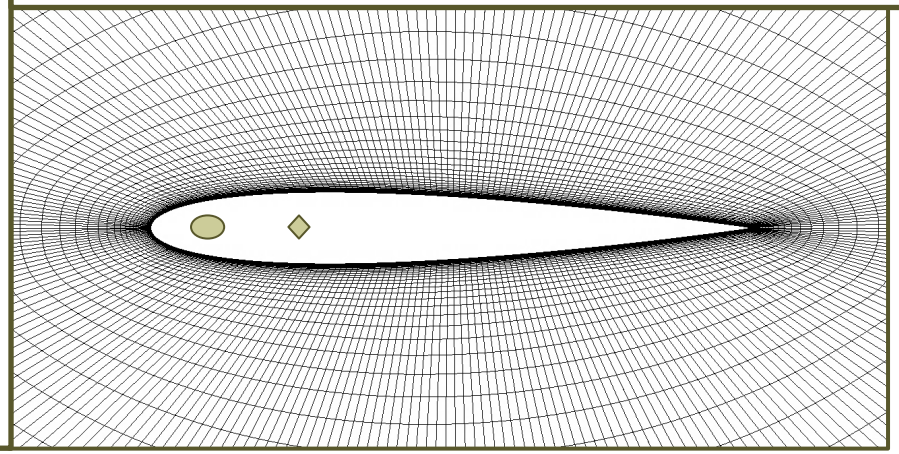
fun3d.nml

```
&project
  project_rootname = 'naca0012'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
/
&force_moment_integ_properties
  area_reference  =  1.0000 ! C*B
  x_moment_length =  1.0000 ! C
  y_moment_length =  1.0000 ! B
  x_moment_center =  0.2500 ! x_cg
  y_moment_center =  0.5000 ! y_cg
  z_moment_center =  0.0000 ! z_cg
/
&governing_equations
  eqn_type      = 'compressible'
  viscous_terms = 'turbulent'
/
&turbulent_diffusion_models
  turbulence_model      = 'sa'
/
&reference_physical_properties
  dim_input_type    = 'nondimensional'
  mach_number       = 0.25
  reynolds_number   = 1e7
  temperature       = 288.15
  temperature_units = 'Kelvin'
  angle_of_attack   = 5.0
  angle_of_yaw      = 0.000
/
&inviscid_flux_method
  flux_construction     = 'roe'
/
&code_run_control
  steps                 = 5000
  restart_read          = 'on_nohistorykept'
/
&nonlinear_solver_parameters
  time_accuracy           = '2ndorder'
  time_step_nondim        = 0.2
  subiterations           = 10
  schedule_iteration(1:2) = 1, 10
  schedule_cfl(1:2)       = 25.0, 25.0
  schedule_cflturb(1:2)   = 10.0, 10.0
/
&global
  moving_grid             = .true.
/
```

moving_body.input

```
&body_definitions
  n_moving_bodies = 1
  relative_velocity_scaling = .false.
  dimensional_output = .false.
! Reference Dimensionalizing Properties:
  ref_velocity = 340.294
  ref_density = 1.2250
  ref_length = 1.0
! Define body:
  body_name(1) = 'airfoil'
  parent_name(1) = ''
  n_defining_bndry(1) = 1
  defining_bndry(1,1) = 2
  motion_driver(1) = '6dof'
  mesh_movement = 'rigid'
  x_mc(1)   = 0.2500      ! x_mrc = x_cg
  y_mc(1)   = 0.0000      ! y_mrc = y_cg
  z_mc(1)   = 0.0000      ! z_mrc = z_cg
  s_ref(1) = 1.0000       ! C*B
  c_ref(1) = 1.0000       ! C
  b_ref(1) = 1.0000       ! B
  move_mc(1) = 1
/
```

```
&sixdof_motion
! Mass properties:
  mass(1) =150.00        ! Body 1 mass
  cg_x(1) =  0.10        ! Body 1 x_cg
  cg_y(1) =  0.00        ! Body 1 y_cg
  cg_z(1) =  0.00        ! Body 1 z_cg
  i_xx(1) =100.00        ! I_xx
  i_yy(1) =  5.00        ! I_yy
  i_zz(1) =100.00        ! I_zz
  i_xy(1) =  0.00        ! I_xy
  i_xz(1) =  0.00        ! I_xz
  i_yz(1) =  0.00        ! I_yz
! Start from "rest" i.e. zeroed out ICs:
  body_lin_vel(1:3,1) = 0., 0., 0.
  body_ang_vel(1:3,1) = 0., 0., 0.
! Gravity vector:
  gravity_dir(1:3) = 0.0, 0.0, -1.0
  gravity_mag = 9.81
! 2D flow, full motion (3-DOF)
  ignore_x_aeroforce(1)   = .false.
  ignore_y_aeroforce(1)   = .true.
  ignore_z_aeroforce(1)   = .false.
  ignore_x_aeromoment(1)  = .true.
  ignore_y_aeromoment(1)  = .false.
  ignore_z_aeromoment(1)  = .true.
! Print libmo summary:
  print_sixdof_summary = .true.
/
```



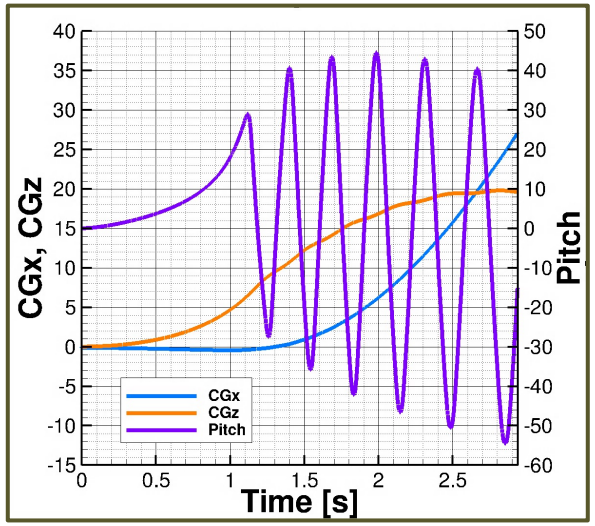
C.G.

MRC (C/4)

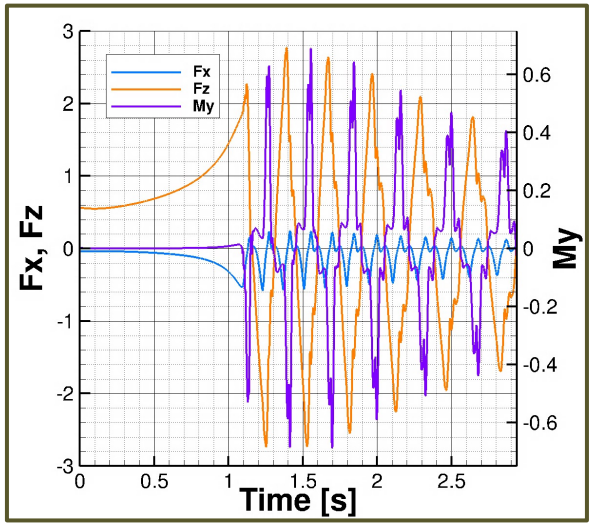
Demo 3a: Planar NACA0012 3-DOF (2/2)

- Case tailored to show variety of motion ranges
- Airfoil is released from a static condition, stable ascending flight
- After ~1s simulated time, reaches stall condition which induces deceleration (x-dir) with pitch (y-axis) - plunge (z-dir) motion.

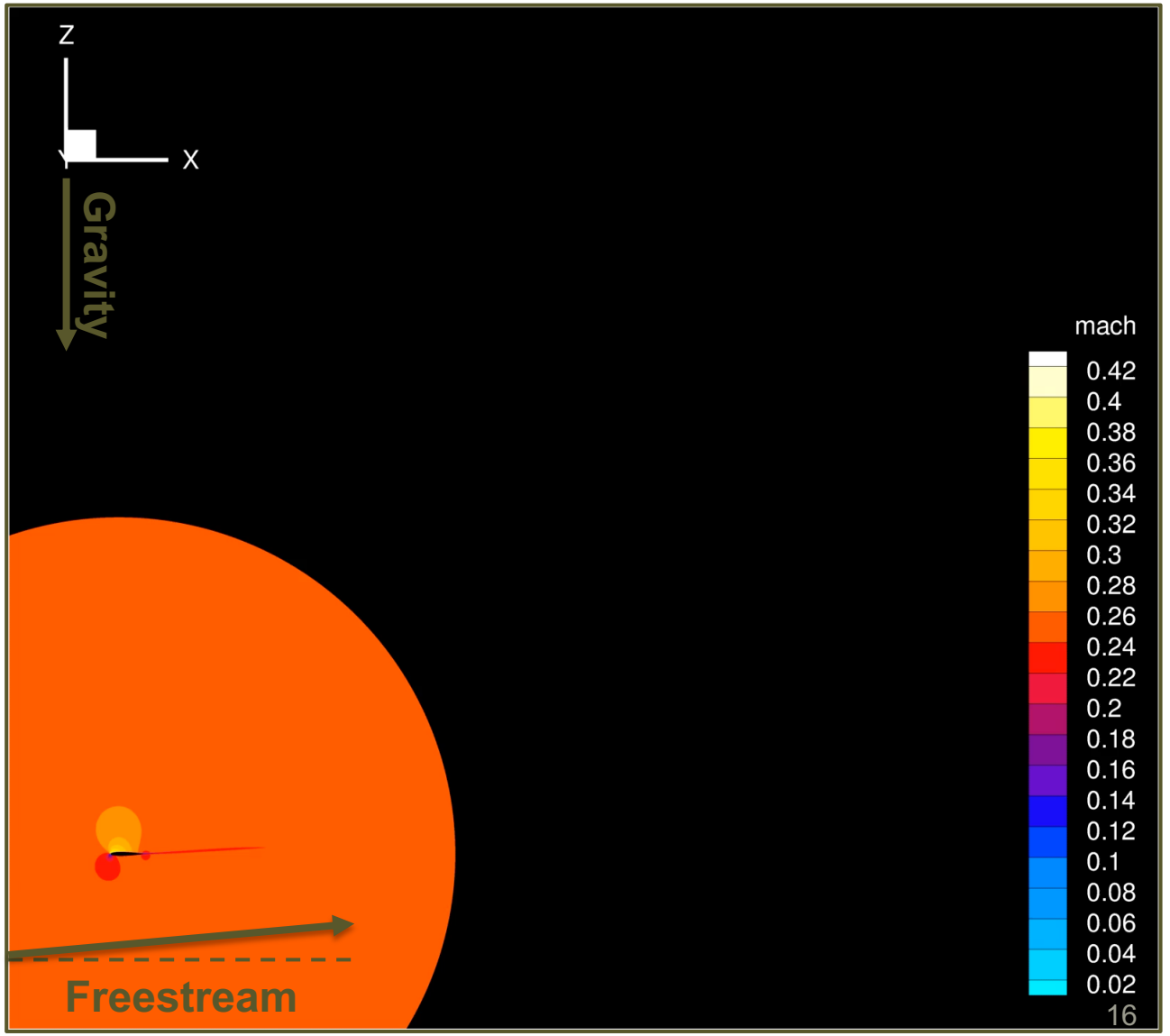
Body Axes Position (left) and Aero-Force-Moment (right) Outputs



Nondimensional Output of Position in Grid Units (linear) and Degrees (angular)



Nondimensional Output of Forces and Moments follows Model Conventions



Demo 3b: Planar NACA0012 3-DOF

- Same as before, but rotated starting position about the CG by 45 degrees
- Starts body at an incidence angle (heavy stall)
- Caution: Stacking Reference Frames!
  - At t=0, AoA = 50deg (Flow: 5 + Body: 45)

fun3d.nml

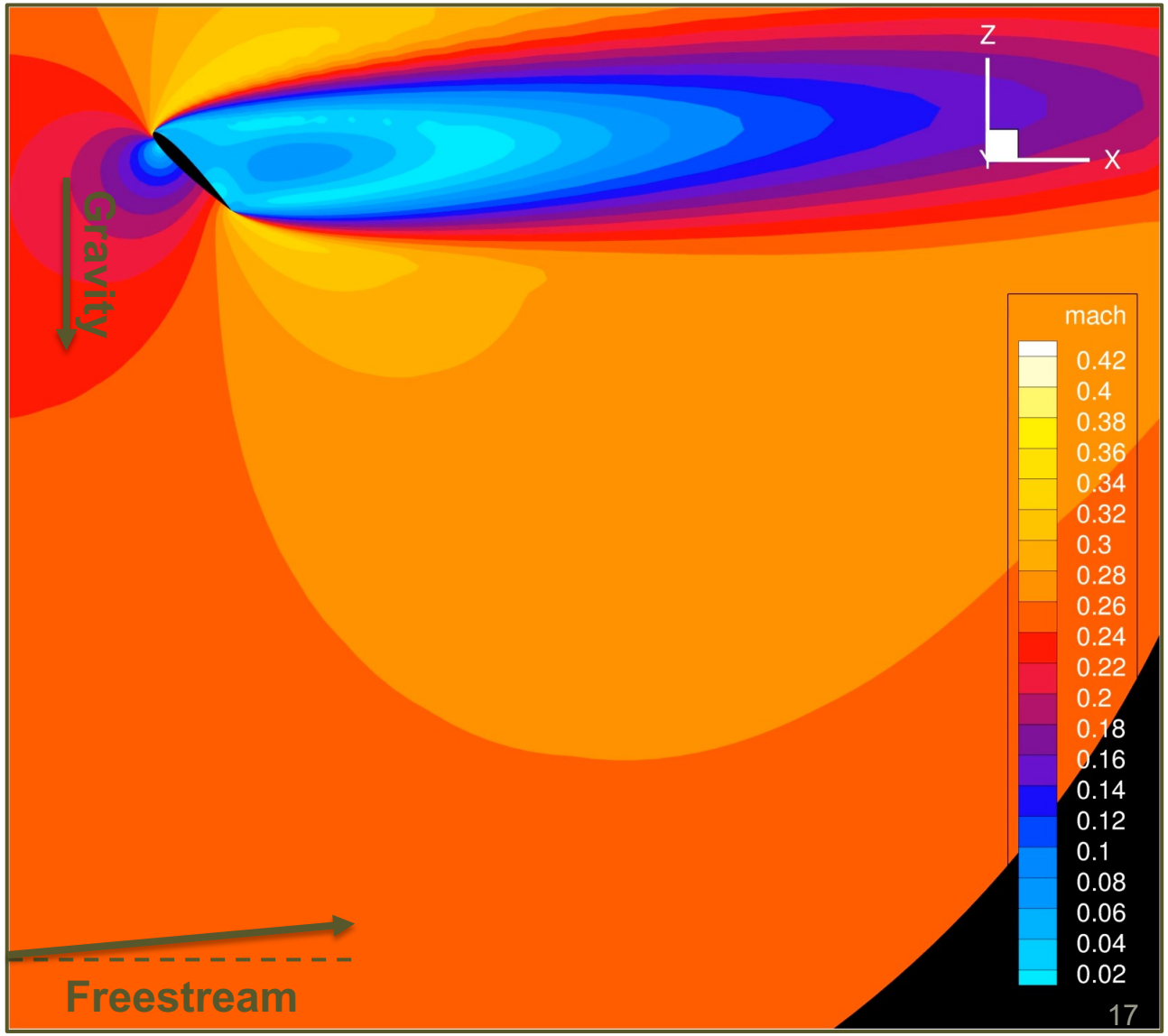
```
&grid_transform
  n_transforms = 1
  theta = 45
  tx = 0
  ty = 1
  tz = 0
  x0 = 0.1
  y0 = 0
  z0 = 0
  echo_transform = .true.
/
```

moving_body.input

```
&body_definitions
  ...
  body_axes_theta(1) = 45
  body_axes_tx(1)    = 0
  body_axes_ty(1)    = 1
  body_axes_tz(1)    = 0
  body_axes_x0(1)    = 0.1
  body_axes_y0(1)    = 0
  body_axes_z0(1)    = 0
  ...
/
```

screen output

```
...(grid transform output)...
 Grid static transform parameters
  Transform number 1 (applied to grid points with imesh = 0)
   Displacement magnitude:     0.00000000E+00
   Displacement vector:        0.10000000E+01   0.00000000E+00   0.00000000E+00
   Rotation angle (deg):       0.45000000E+02
   Rotation vector:            0.00000000E+00   0.10000000E+01   0.00000000E+00
   Scale factor:               0.10000000E+01
...(sixdof output)...
Euler Angles:
      yaw = 0.000000, pitch = 45.000000, roll = 0.000000
```



Considerations & Failure Modes

Errors will integrate, and therefore propagate, in time

- 6-DOF is a coupled Flow & Motion solution
- Each solution is only as good as the other
- Users should be considerate of temporal quality, (e.g., validity of initial state, subiteration convergence)

Stacking of Reference Frames

- Easy for errors to occur if user is not careful
- Verify your body axes are where you want them
- Forced motion can be a good building block

Solver Output Is Your Friend

- Runtime outputs will often point to the source of problems

Many application specific problems may not be obvious without case-specific knowledge

What We Learned

- How to run a 6-DOF simulation
  - Input options pertinent to body motion
    - Overview of the `&sixdof` namelist
  - Tips and tricks for debugging cases
  - Common outputs from solutions
- Considerations when running
  - Treat 6-DOF solutions as an initial value problem
  - Time-accurate simulation best practices should be followed
- More Questions?
  - Public Community: fun3d-users@lists.nasa.gov
  - Private/Proprietary: fun3d-support@lists.nasa.gov