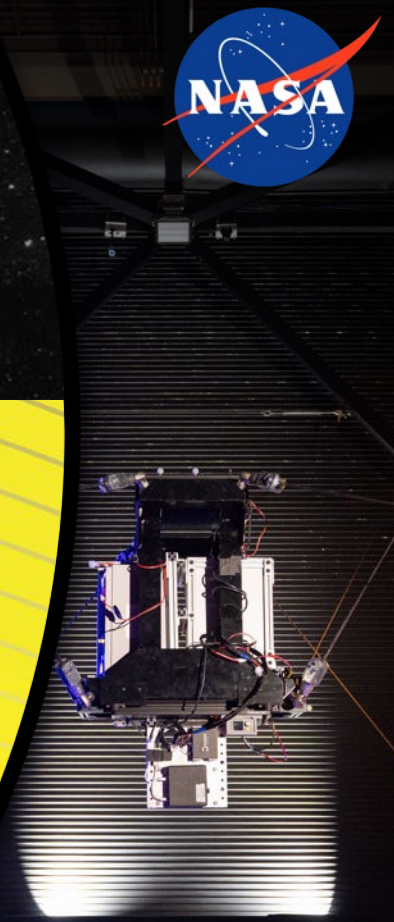
Flight Reconstruction of the Starship Landing Maneuver using OVERFLOW



Darby Vicker, Amentum
EG3 – Applied Aeroscience and CFD
NASA Johnson Space Center, Houston TX

Background

- Through the Human Landing System (HLS) program, NASA and SpaceX are collaborating on Starship launch and reentry aerodynamic and aerothermal environments
- Precision entry is necessary due to the desire to catch the Starship vehicle (both booster and the 2nd stage)
- This has lead to the need to perform high fidelity analysis of the landing maneuver
  - Unsteady flow
    - Low dissipation 5th order spatial accuracy
    - Delayed Detached Eddy Simulation (DDES)
  - Moving body
    - Overall vehicle motion
    - Gimbaling nozzles
    - Moving controls surfaces
  - Adaptative Mesh Refinement (AMR)
  - Quiescent flow
  - Variable thrust

Case Types

- Standard
  - Lagrangian frame
  - Moving fluid, fixed body
  - ”Normal” cases at fixed conditions
- Relative Motion
  - Lagrangian frame
  - Moving fluid, Main body fixed with moving control surfaces
  - Used to explore isolated control surface movement at fixed free stream conditions
- Flight reconstruction
  - Eulerian frame
  - Quiescent flow (FSMACH=0), moving body and control surfaces,
  - Used for full flight reconstruction cases where everything is changing (free stream conditions, vehicle orientation, control surface movement, etc.)

Configuration and Motion Definition

- Overflow uses the Geometry Manipulation Protocol (GMP) to describe the motion
  - Extremely flexible and powerful hierarchical framework used for component motion
  - Uses **equations** to define the component velocity
  - Incorporated into both CART3D and OVERFLOW
    - Inputs are `Config.xml` and `Scenario.xml`
- Canonical relative motion cases (e.g. just motion of the flaps) are straight forward to set up
- Previous experience on flight reconstruction-like cases is with relatively simple motion, utilizing a polynomial fit of the velocity
- That approach is not viable for this work as the vehicle and control surface motions are too complex for a simple polynomial fit
- Instead, we've used a piecewise cubic spline to fit the provided trajectory variables, resulting in a good representation of the motion

Configuration Definition

- The overall configuration is defined with `Config.xml` (partial sample shown – built by CGT scripts)

Config.xml

```
<?xml version='1.0' encoding='utf-8'?>
<Configuration AngleUnit='degree'>

     <Component Name='ship' Type='struc' Parent='ship_total'>
        <Data> Grid List=1-5,7-14,39-102,127-178 </Data>
     </Component>

     <Component Name='raptor1_total' Type='struc' Parent='ship_total'>
        <Data> Grid List=15-22 </Data>
        <Transform>
           <Translate Displacement='10,0,0' />
           <Rotate Center='0, 0, 0' Axis='0,1,0' Angle='0.0' />
           <Rotate Center='0, 0, 0' Axis='0,0,1' Angle='0.0' />
        </Transform>
     </Component>

     <Component Name='aft_flap_r_hl' Type='struc' Parent='ship_total'>
        <Data> Grid List=103,105,107,109,111-112 </Data>
        <Transform>
           <Translate Displacement='0,0,-20' />
           <Rotate Center=0, 0, 0' Axis='1, 0, 0' Angle='0.0' />
        </Transform>
     </Component>

     <Component Name='ship_total' Type='container'>
     </Component>

</Configuration>
```

Prescribed Motion Definition

- Developed a python script to automate this process that reads the trajectory, spline fits the variables (using CubicSpline from scipy), and builds the Scenario.xml file
- GMP wants velocities but you can provide this script a position and differentiate the cubic polynomial to get the velocity

```
# Both of these modules are part of our in-house aero_utils package
import cdat  # A module the reads and manipulates "columnized data"
import gmp   # A module that builds Scenario.xml files

traj = cdat.ColDat('trajectory.cdat') # Load the trajectory file

segs = '' # Initialize a string to store segments

# Build ship segments - inputs can be a specific value, or a variable name in the cdat file
segs += gmp.construct_prescribed_trajectory(cdat= traj, xmlcomp='ship_total', time='time',
     vx='vx_body',     vy='vy_body',     vz='vz_body',
     cx='body_cg_x',   cy='body_cg_y',   cz='body_cg_z',
     rx='body_rate_x', ry= body_rate_x', rz='body_rate_x',
     vref=100, frame='body')

# Build flap segments
segs += gmp.construct_prescribed_trajectory(cdat=traj, xmlcomp='aft_flap_r', time='time',
     cx=1,cy=2,cz=3,
     rg='aft_flap_r_position', axis='1,2,3'
     vref=100, rot_der=True, frame='parent')

gmp.write_scenario(segs,filename='Scenario.xml')
```

Prescribed Motion Definition

- 6-DOF Vehicle motion. Note the varying rotation center for CG movement.

Scenario.xml

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario AngleUnit='degree'>

<InitialPosition Component="ship_total" >
    <Rotate Center="10.0,0,0" Axis="0,1,0" Angle="0.54816" />
    <Translate Displacement="0,0,800" />
</InitialPosition>

<Prescribed Component="ship_total" Start="0.0" Duration="10">
    <Rotate    Frame="body"   Center="1.3682,-0.00058,-0.162159" Axis="1,0,0" Speed="-7.1702e-07*(t-0.0)^3+2.15e-05*(t-0.0)^2-1.7454e-05*(t-0.0)+0.016" />
    <Rotate    Frame="body"   Center="1.3682,-0.00058,-0.162159" Axis="0,1,0" Speed="8.21152e-08*(t-0.0)^3-6.26e-06*(t-0.0)^2+6.0825e-05*(t-0.0)+0.004" />
    <Rotate    Frame="body"   Center="1.3682,-0.00058,-0.162159" Axis="0,0,1" Speed="-3.4998e-07*(t-0.0)^3+9.07e-06*(t-0.0)^2+9.1236e-05*(t-0.0)+0.004" />
    <Translate Frame="body"
        Velocity="-5.6737e-07*(t-0.0)^3+1.754e-05*(t-0.0)^2-0.0001*(t-0.0)+0.0,
                  -4.8410e-07*(t-0.0)^3+2.096e-05*(t-0.0)^2+9.674e-05*(t-0.0)+0.0,
                   2.8906e-08*(t-0.0)^3-2.072e-07*(t-0.0)^2-8.733e-05*(t-0.0)+0.0"
    />
</Prescribed>
<Prescribed Component="ship_total" Start="10.0" Duration="10">
    <Rotate    Frame="body"   Center="1.368,-0.0007,-0.162017" Axis="1,0,0" Speed="-7.1702e-07*(t-10.0)^3+1.99999e-08*(t-10.0)^2+0.00019*(t-10.0)+0.0181" />
    <Rotate    Frame="body"   Center="1.368,-0.0007,-0.162017" Axis="0,1,0" Speed="8.21152e-08*(t-10.0)^3-3.79890e-06*(t-10.0)^2-3.7e-05*(t-10.0)+0.0042" />
    <Rotate    Frame="body"   Center="1.368,-0.0007,-0.162017" Axis="0,0,1" Speed="-3.4999e-07*(t-10.0)^3-1.42248e-06*(t-10.0)^2+0.00016*(t-10.0)+0.0019" />
    <Translate Frame="body"
        Velocity="-5.67e-07*(t-10.0)^3+5.25e-07*(t-10.0)^2+2.92e-05*(t-10.0)-0.00032,
                  -4.84e-07*(t-10.0)^3+6.44e-06*(t-10.0)^2+0.000370*(t-10.0)+0.00257,
                   2.89e-08*(t-10.0)^3+6.60e-07*(t-10.0)^2-8.27e-05*(t-10.0)-0.00086"
    />
</Prescribed>
<Prescribed Component="ship_total" Start="20.0" Duration="10">
   ...
</Prescribed>
```

Prescribed Motion Definition

- Raptor gimbal motion is similar but is done in only two axes about their respective gimbal points

Scenario.xml

```
<Prescribed Component="raptor1_total" Start="80.0" Duration="10">
    <Rotate    Frame="parent"   Center="2.953,0.0,0.9" Axis="0,1,0" Speed="2.983e-08*(t-80.0)^3-1.891e-07*(t-80.0)^2-1.0899e-06*(t-80.0)-4.58879e-07" />
    <Rotate    Frame="parent"   Center="2.953,0.0,0.9" Axis="0,0,1" Speed="-2.43e-08*(t-80.0)^3+1.545e-07*(t-80.0)^2+8.9230e-07*(t-80.0)-3.12062e-18" />
</Prescribed>
<Prescribed Component="raptor1_total" Start="90.0" Duration="10">
   ...
</Prescribed>


<Prescribed Component="raptor2_total" Start="80.0" Duration="10">
    <Rotate    Frame="parent"   Center="2.954,-0.779423,-0.45" Axis="0,1,0" Speed="9.13e-08*(t-80.0)^3-7.503e-07*(t-80.0)^2-4.36e-06*(t-80.0)-5.21e-07" />
    <Rotate    Frame="parent"   Center="2.954,-0.779423,-0.45" Axis="0,0,1" Speed="2.11e-07*(t-80.0)^3-1.447e-06*(t-80.0)^2-8.31e-06*(t-80.0)-1.25e-07" />
</Prescribed>
<Prescribed Component="raptor2_total" Start="90.0" Duration="10">
   ...
</Prescribed>


<Prescribed Component="raptor3_total" Start="80.0" Duration="10">
    <Rotate    Frame="parent"   Center="2.955,0.779423,-0.45" Axis="0,1,0" Speed="9.85e-06*(t-80.0)^3-5.85e-05*(t-80.0)^2-0.06*(t-80.0)-1.18942e-07" />
    <Rotate    Frame="parent"   Center="2.955,0.779423,-0.45" Axis="0,0,1" Speed="3.60e-06*(t-80.0)^3-2.15e-05*(t-80.0)^2-0.02*(t-80.0)+3.42919e-06" />
</Prescribed>
<Prescribed Component="raptor3_total" Start="90.0" Duration="10">
   ...
</Prescribed>
```

Prescribed Motion Definition

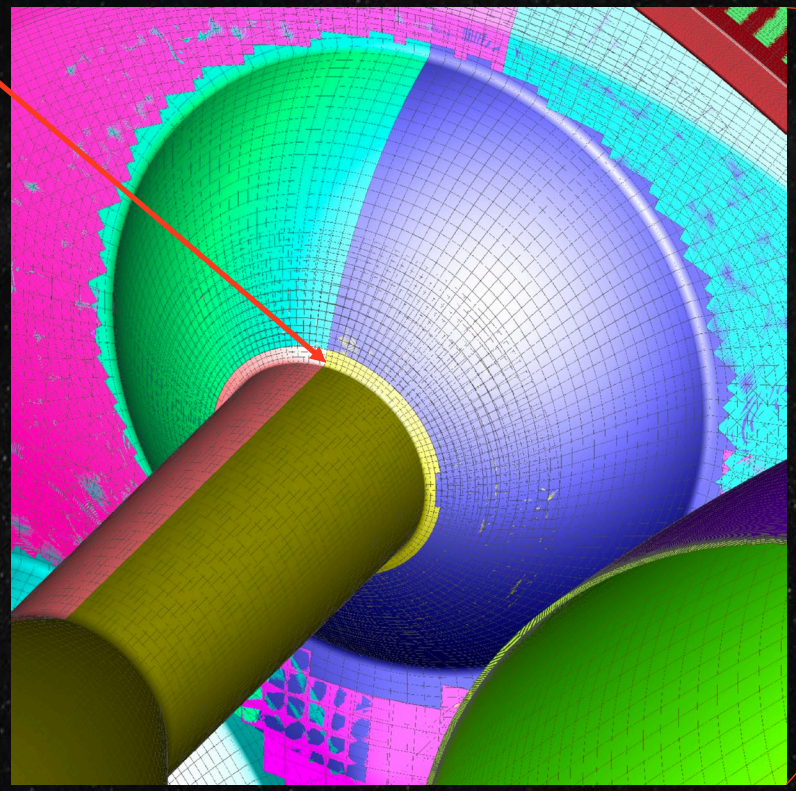
- Flap motion is similar but is done only about their hinge lines
  - Note: used position data for flaps, so this is a 2nd-order polynomial in this case

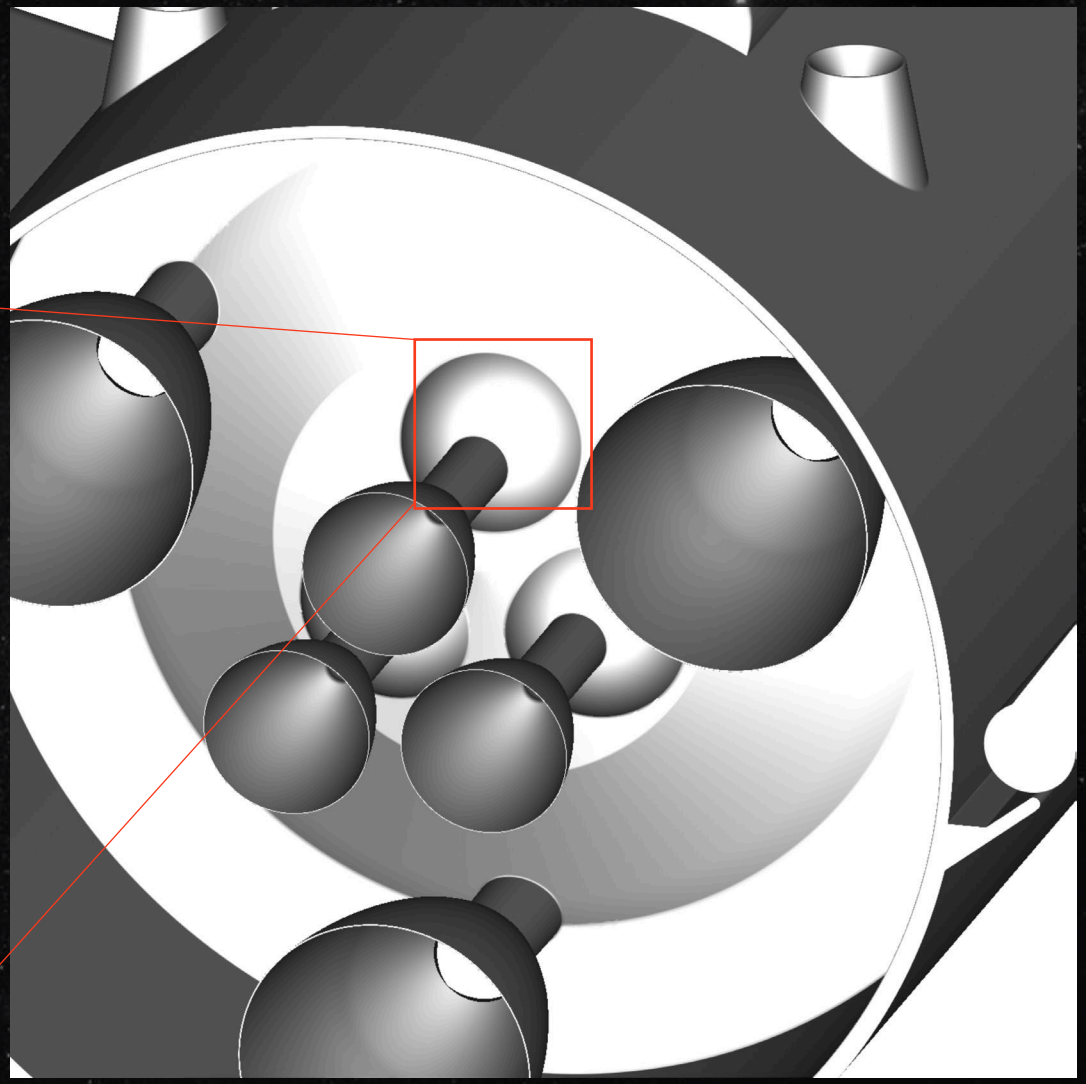
Scenario.xml

```
<Prescribed Component="aft_flap_l_hl" Start="0.0" Duration="10">
    <Rotate    Frame="parent"   Center="1,3,0" Axis="-1.0,0.0,0.0" Speed="-1.07453796e-05*(t-0)^2+1.86907592e-04*(t-0)-4.47063586e-02" />
</Prescribed>

<Prescribed Component="aft_flap_r_hl" Start="0.0" Duration="10">
    <Rotate    Frame="parent"   Center="1,-3,0" Axis="1.0,0.0,0.0" Speed="-2.50171346e-05*(t-0)^2-3.38657308e-04*(t-0)+3.72719103e-03" />
</Prescribed>

<Prescribed Component="fwd_flap_l_hl" Start="0.0" Duration="10">
    <Rotate    Frame="parent"   Center="50,2,1" Axis="-0.9,0.3,0.1" Speed="7.44706556e-06*(t-0)^2+5.30586889e-05*(t-0)+2.12164710e-02" />

<Prescribed Component="fwd_flap_r_hl" Start="0.0" Duration="10">
    <Rotate    Frame="parent"   Center=”50,-2,1 Axis="0.9,0.3,-0.1" Speed="-1.76152573e-06*(t-0)^2+1.52305145e-05*(t-0)-2.61774350e-02" />
</Prescribed>
```

Raptor Gridding Details

- The base of the raptors is a cylinder and interfaces with a spherical surface
- A collar at the base of the raptor slides along the spherical surface, cutting holes dynamically (xrays)

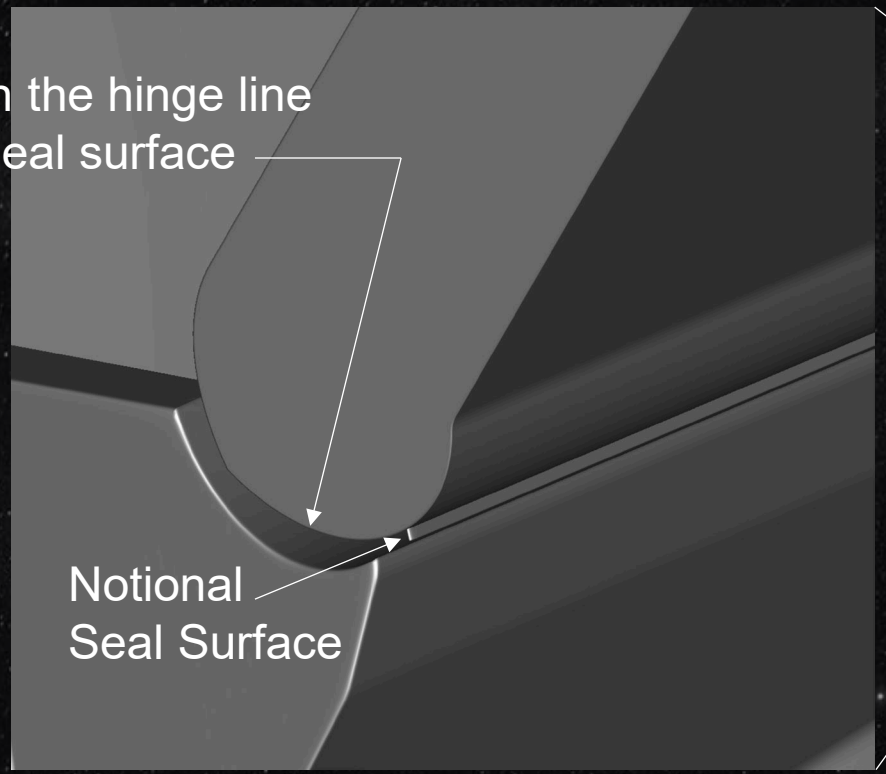


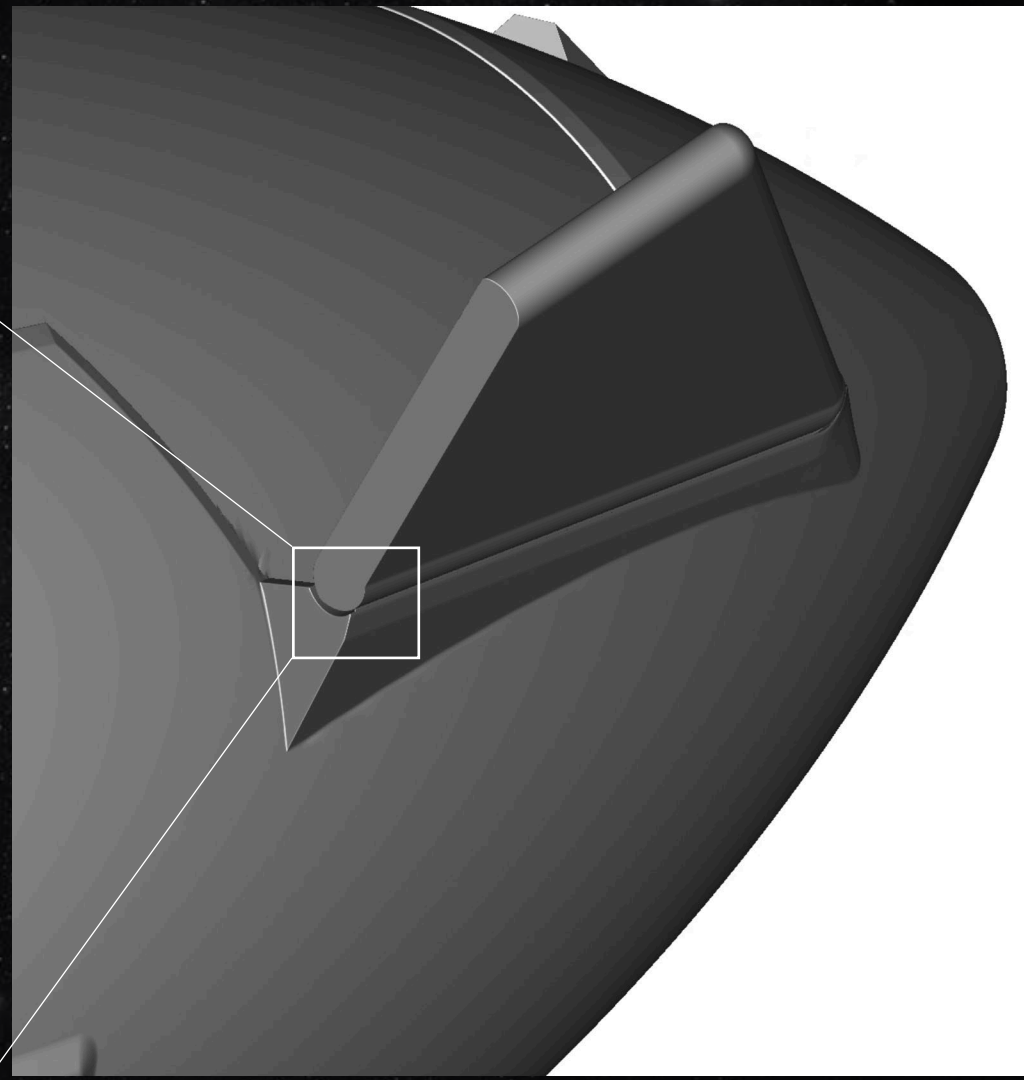


Flap Gridding Details

- The flaps “float” next to the main body with a small gap
- A small seal surface prevents flow through the gap
- Again, each flap has its own xray and cuts holes dynamically

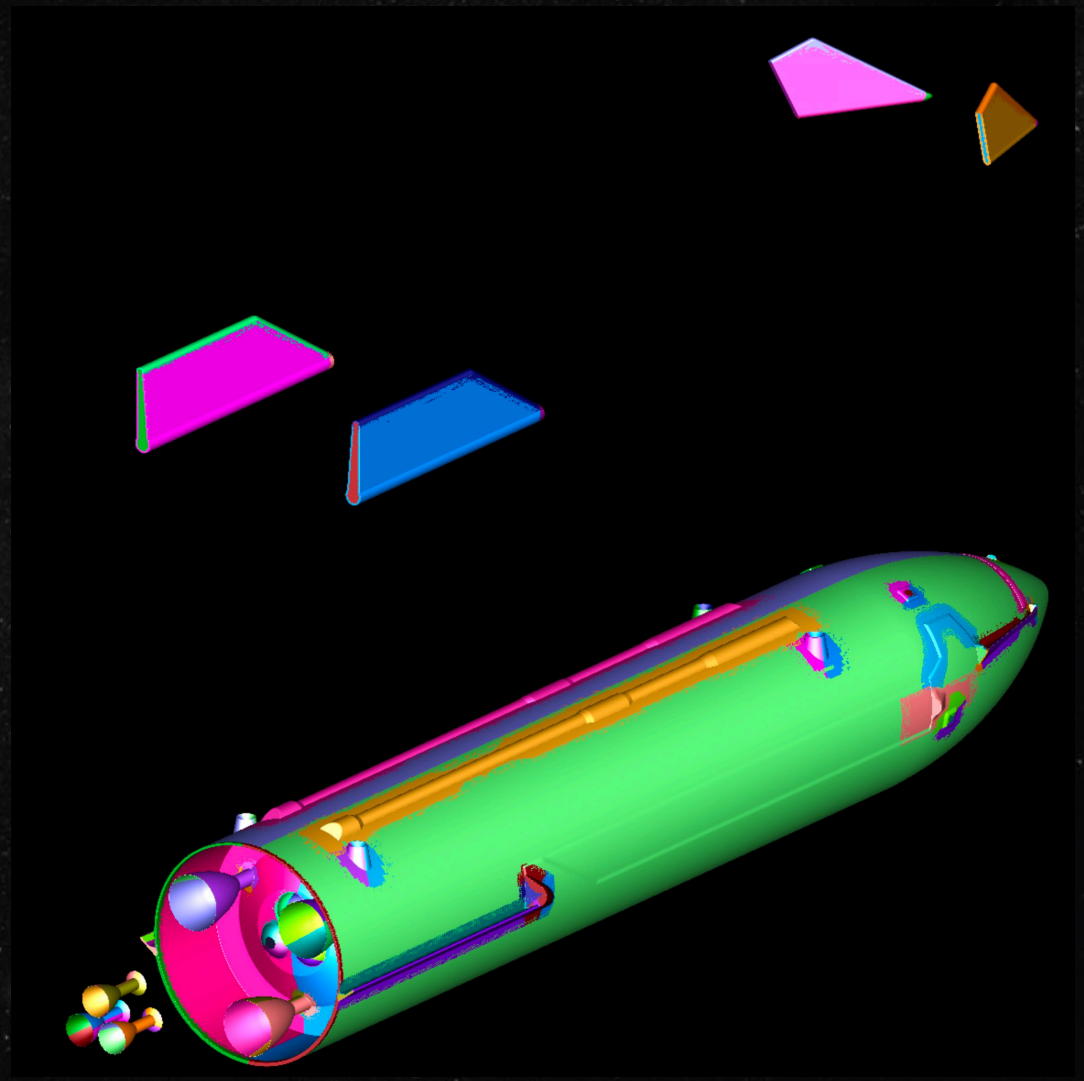
This surface is conical with the hinge line and stays fixed WRT the seal surface



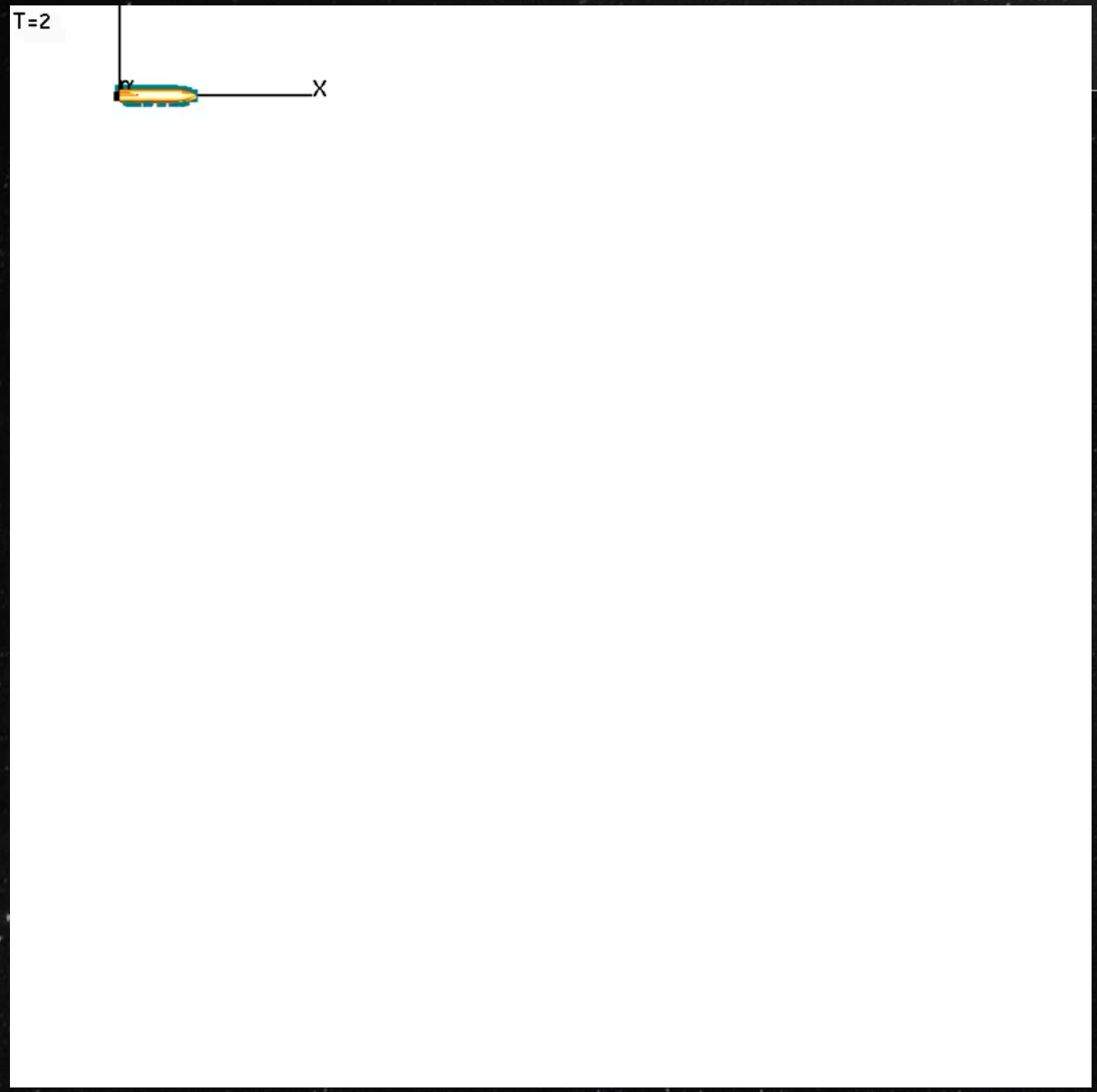


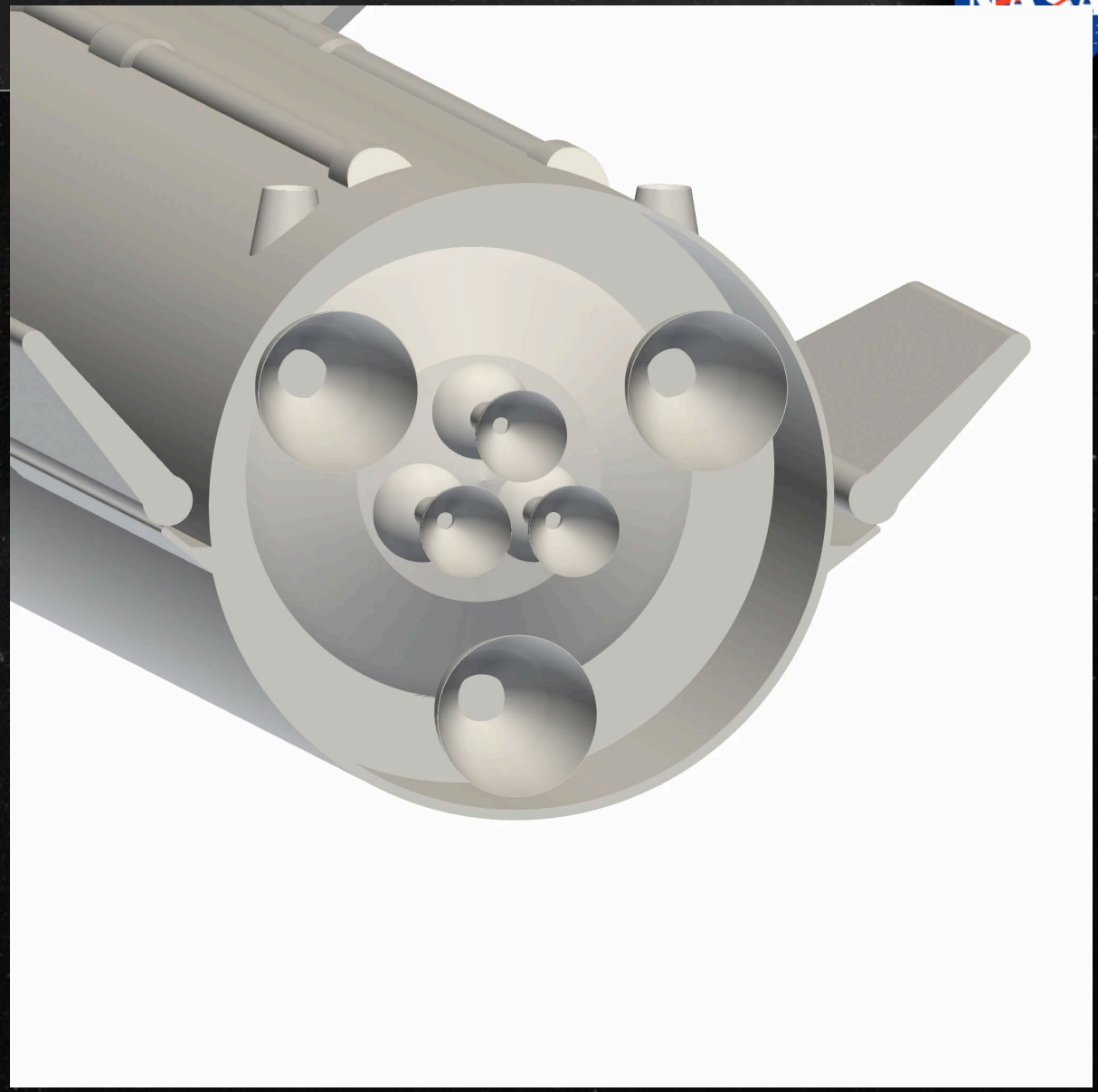
Hybrid Pegasus/DCF

- OVERFLOW's hybrid Pegasus/DCF mode is utilized
- Most hole cutting is done in Pegasus and takes advantage of its easier inputs and level 2 overlap minimization
- Moving components are translated away from each other initially
  - Needed so holes in these components can be computed dynamically
  - Config.xml moves them back into their base positions
  - Scenario.xml moves them according to their prescribed schedules



Prescribed Motion Results





Time Varying Thrust

- Easily handeled with recent BC's in overflow (BC 153)

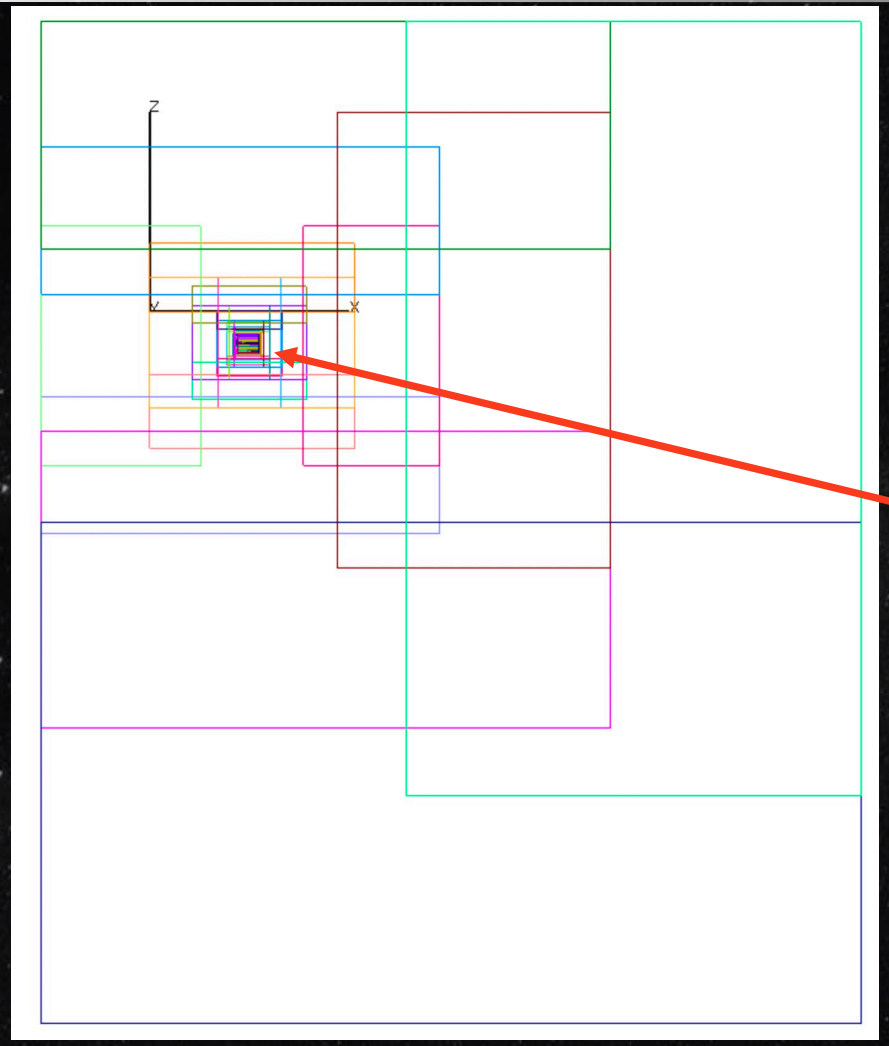
```
$BCINP
    IBTYP =     5,  153,
    IBDIR =     3,    1,
    JBCS  =     1,    1,
    JBCE  =    -1,    1,
    KBCS  =     1,    1,
    KBCE  =    -1,   -1,
    LBCS  =     1,    1,
    LBCE  =     1,   -1,
    BCPAR1(2) = 1,
    BCPAR2(2) = 500,
    BCFILE(2) = 'raptor1.dat',
    $END
```

overflow.inp

```
100.0 10.0                    ! Total Pressure and Temperature
0.0000 1.0000                 ! Species fractions
raptor1_inflow                ! Mixsur inflow component
114                           ! Number of conditions in thrust table
     0   0.01                 ! Time vs thrust table
    10   0.01
    20   0.01
    30   0.01
    40   0.01
    50   0.01
    60   0.01
    70   0.01
    80   0.01
    90   0.01
   100   0.01
   110   0.0380493
   120   0.0772846
   130   0.805694
   140   0.955731
   150   1.02775
```

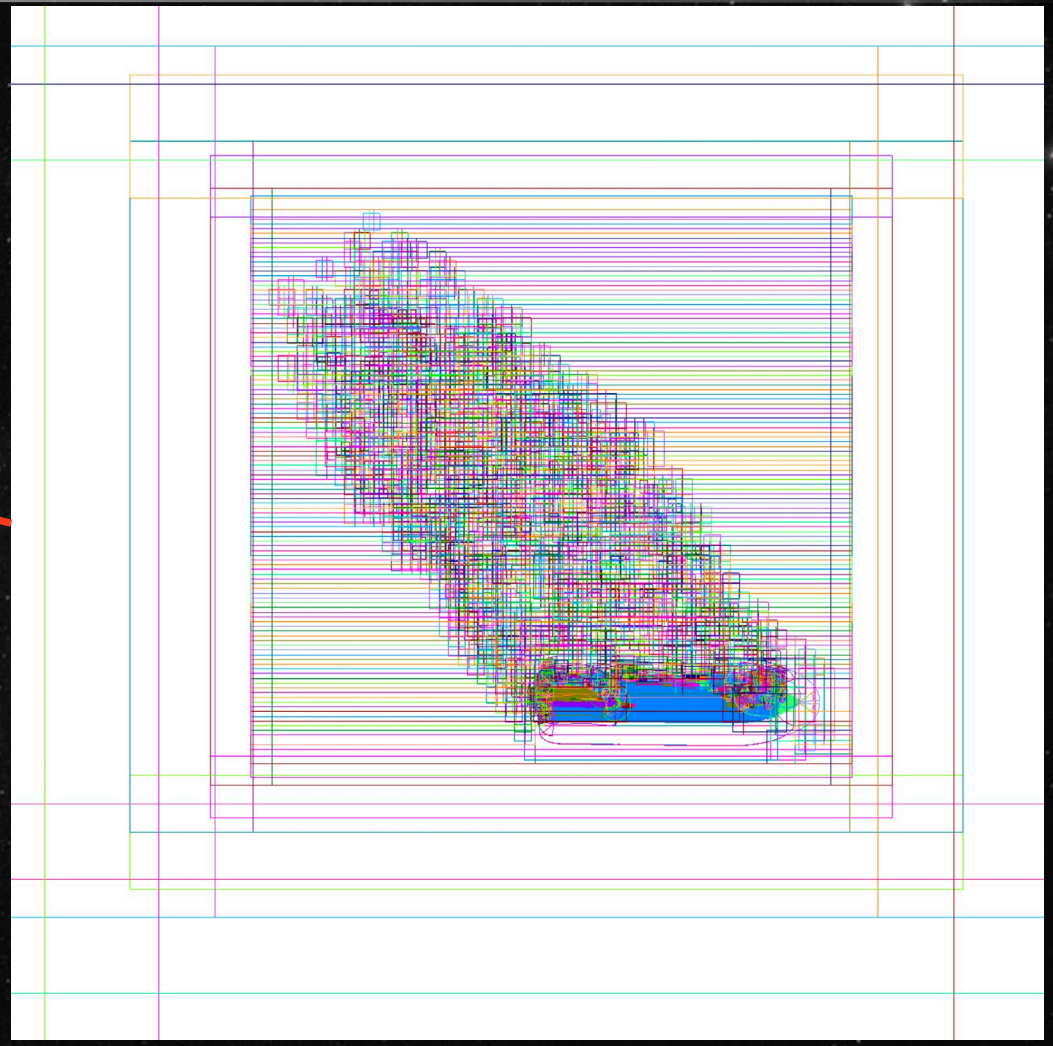
raptor1.dat

Domain Size For Reconstruction Cases

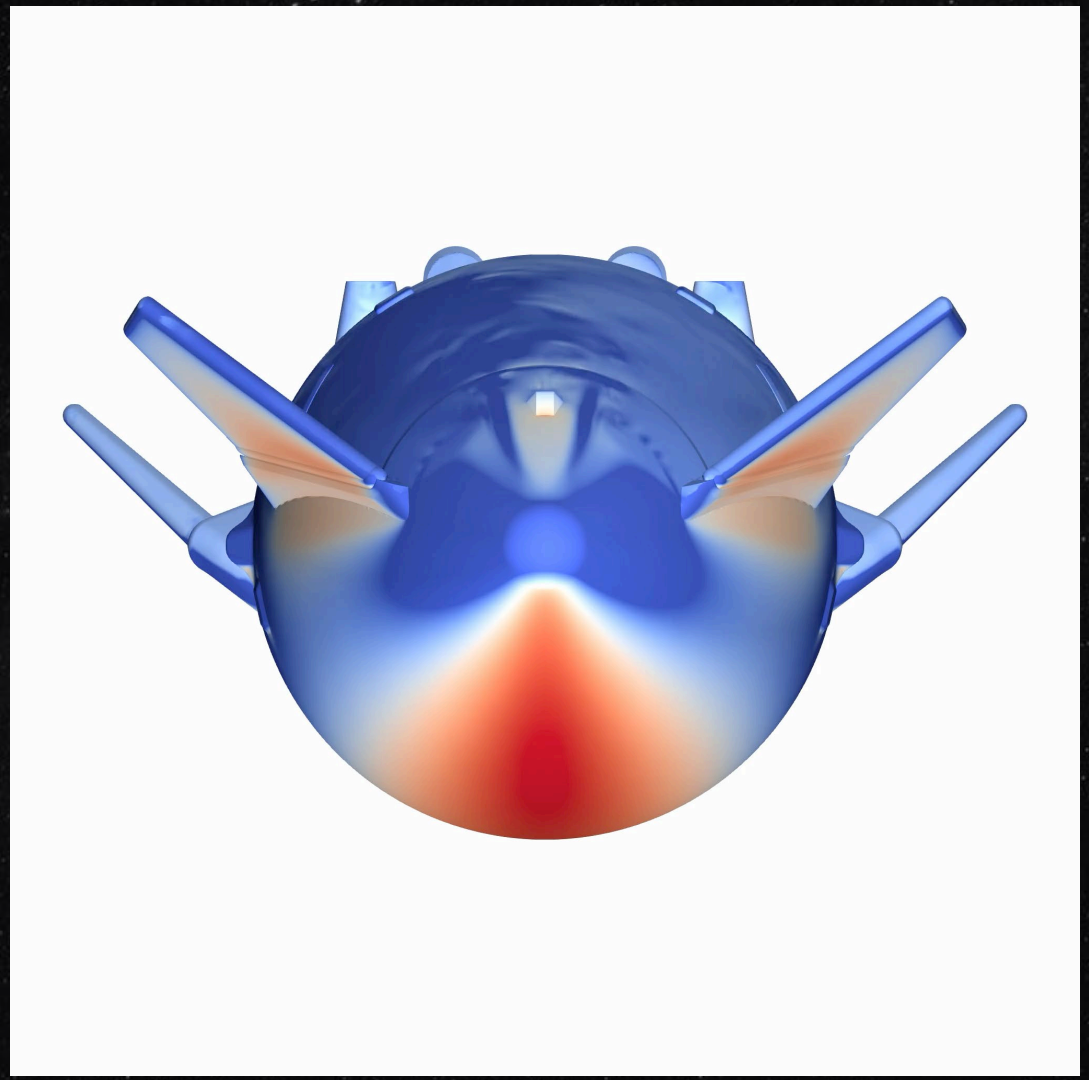


5.6 km

4.6 km

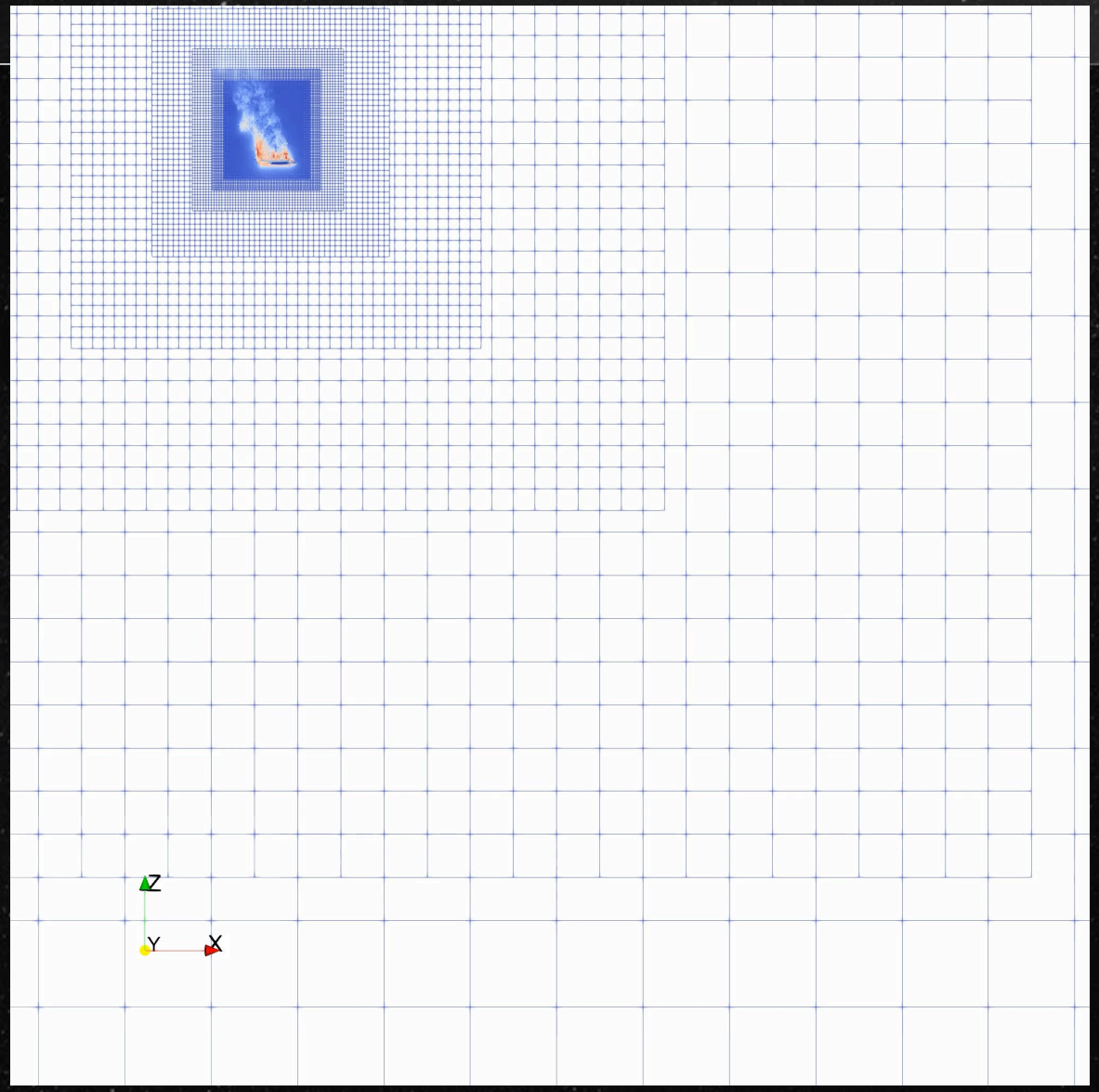


Results:
Relative Motion

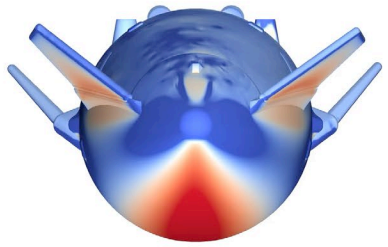


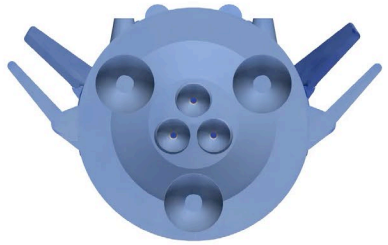
Results: Flight Reconstruction

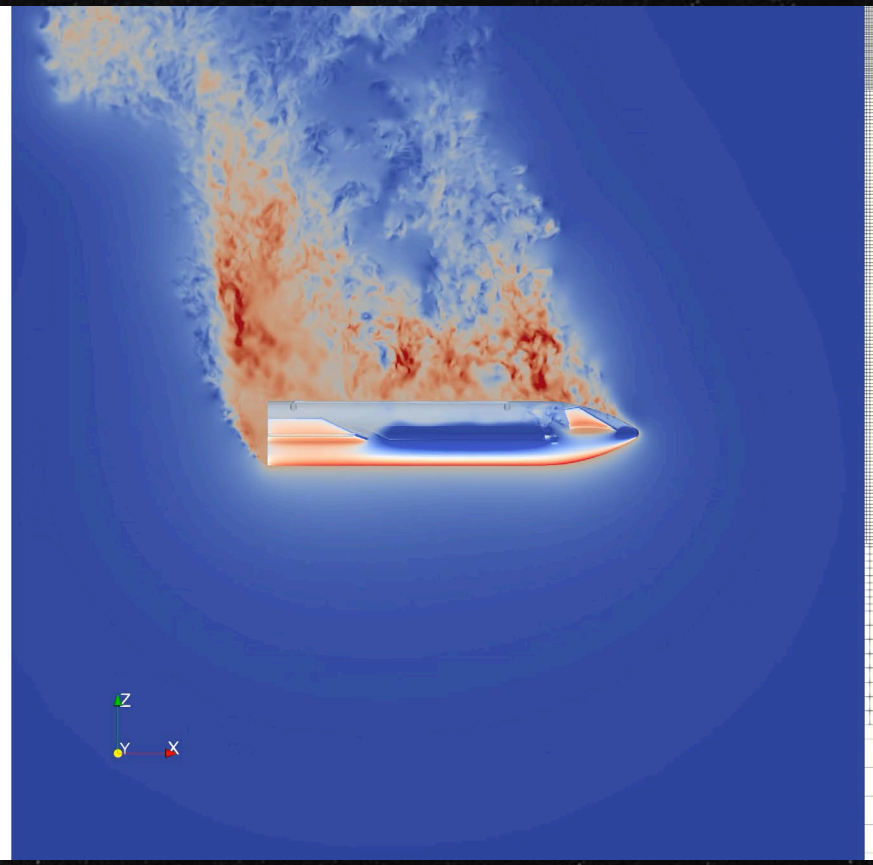
Note, I plan to replace this with the complete flight 13 movie

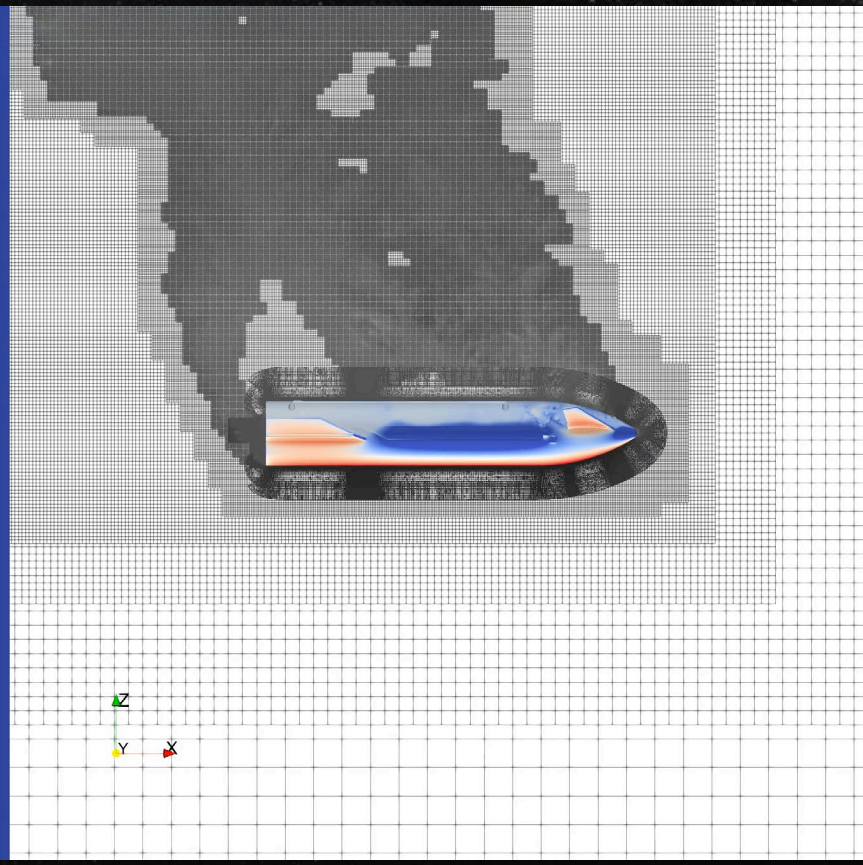


Results:
Flight Reconstruction









Note, I plan to replace this with the complete flight 13 movie

Statistics

- Base Grid – 178 grids, 150 million grid points
- Adapted grid – 5000+ grids, 300 million grids points (limited by MAX_SIZE)
- Time step = ~1e-4 seconds (10000 iterations per second)
- Physical time simulated ranging from 5 to 25 seconds
- Using 1535 to 3072 cores on NAS's Athena, we can achieve about 1 second of physical time per day
  - **AMD Turin** EPYC 9745 CPU, 256 cores per node
  - Cray Slingshot interconnect, 400 Gbps per node
- Post processing
  - Full q file every 100 iterations written as double precision, later converted to single
    - generates 50+ TB per run
    - Used for full visualization
  - Surface subsets every 5 iterations, written directly as single precision (`IPRECIS=1` in `$SPLITM`)
    - generates ~1 TB per run
    - Used for post processing of line loads, pressures, etc.

Conclusion

- Canonical cases (isolated control surface movement) have been useful in building aerodynamic models
  - characterizing time-dependent aerodynamic responses via parametric variation of control surface rates and amplitudes
- Flight reconstruction cases have been useful in understanding flight behavior
- Both have been useful in designing the flight control system and meeting the landing accuracy requirements needed for a ship tower catch

National Aeronautics and Space Administration

Questions?

NASA is grateful to SpaceX for permission to publish this work