OVERFLOW 2 Training Class

Joseph M. Derlaga
NASA Langley Research Center
joseph.m.derlaga@nasa.gov

Pieter G. Buning
NASA Langley Research Center
pieter.g.buning@nasa.gov

13th Symposium on Overset Composite Grids & Solution Technology
Future of Flight Aviation Center
Mukilteo, WA
October 17-20, 2016

Class Outline

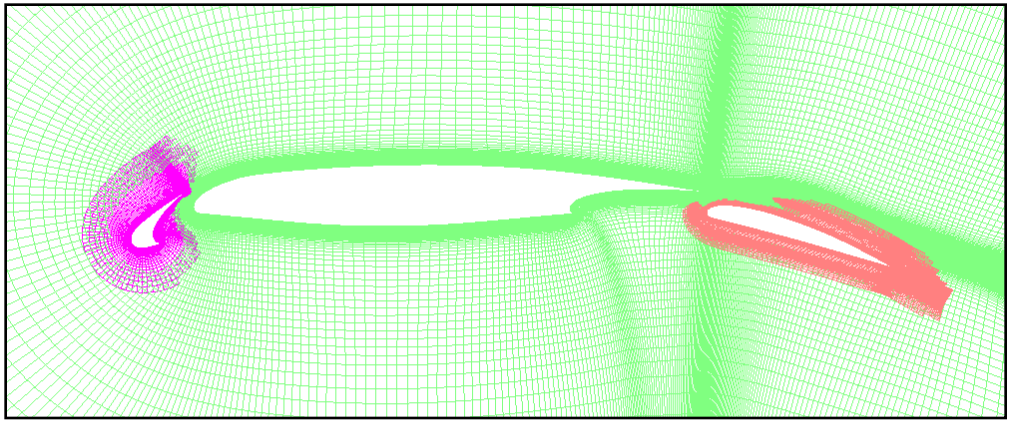
- **Introduction/review**
- **OVERFLOW-D mode prerequisites**

Introduction/Review

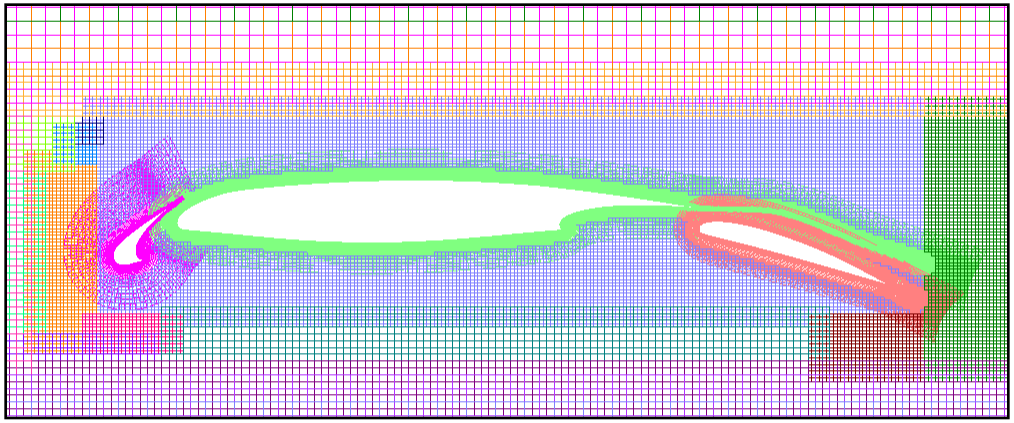
- Overset grid process
  - Compare OVERFLOW mode vs. OVERFLOW-D mode
  - Input files
  - Moving body simulation

Overset Grid Approaches: Multi-Element Airfoil

- OVERFLOW mode
  - All grids supplied
  - Grid system from Peg 5
  - Holes are cut automatically, based on comparable cell sizes
  - Better quality grid system



- OVERFLOW-D mode
  - Only near-body grids supplied
  - Distance from surfaces specified for hole cutting
  - Holes cut by DCF inside OVERFLOW
  - Hole cutting is fast enough for moving body problems



Input Files for OVERFLOW 2

What do you need to be able to run?

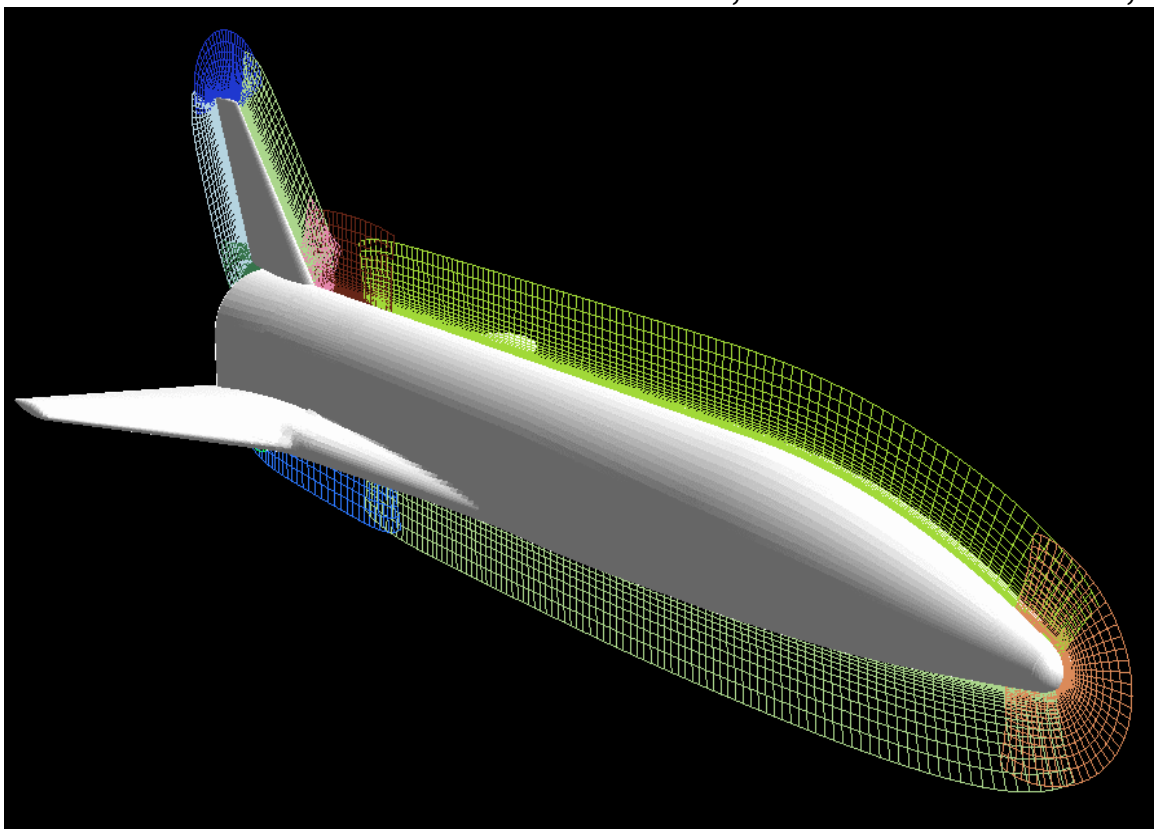
- OVERFLOW mode
  - **grid.in** (all grids)
  - **mixsur.inp** (input to force and moment preprocessor)
  - **XINTOUT** (Pegasus 5 hole cutting and interpolation stencils)
  - OVERFLOW namelist input
- OVERFLOW-D mode
  - **grid.in** (near-body grids)
  - **mixsur.inp** (input to force and moment preprocessor)
  - **xrays.in** (x-rays for hole cutting)
  - OVERFLOW namelist input
  - **Config.xml, Scenario.xml** (body properties and positioning)

Moving Body Simulation Process

- Pre-processing:
  - Near-body grid generation
  - Creating X-rays for hole cutting
- OVERFLOW grid processing:
  - Off-body grid generation
  - Hole cutting and boundary interpolation stencils
- Moving body simulation
  - Body motion (GMP interface)
- Post-processing
  - Non-trivial!

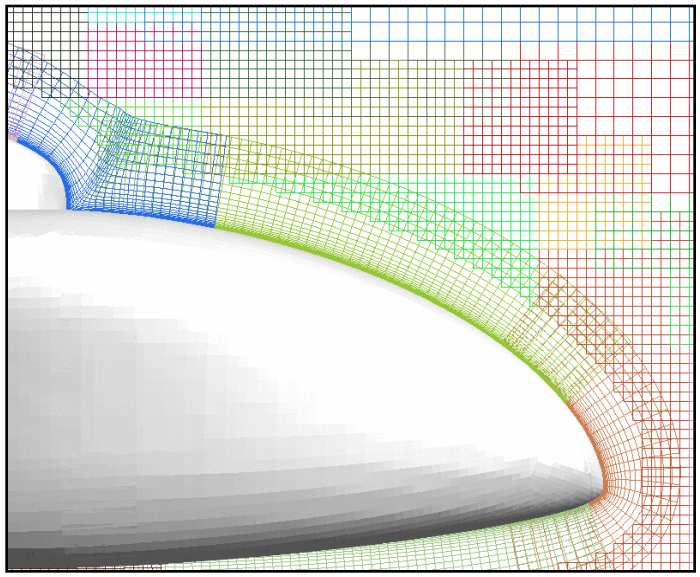
Near-Body Grid Generation

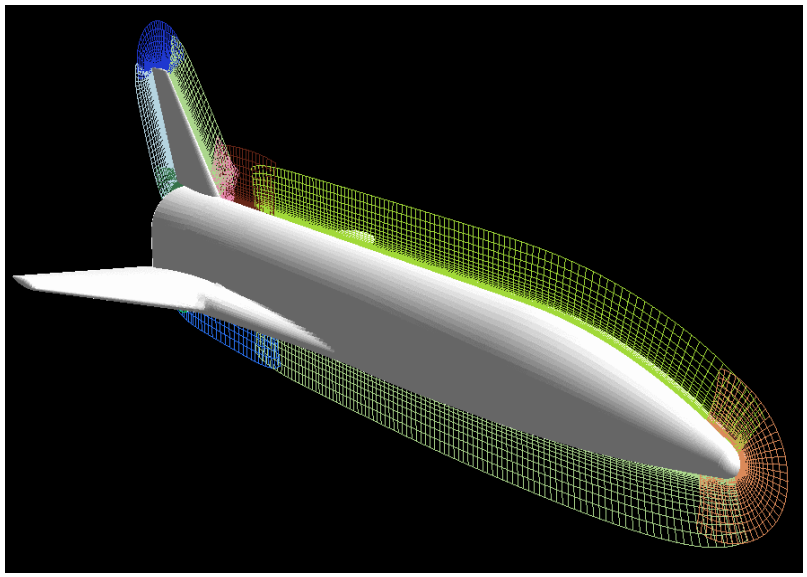
- Volume grids are generated from overset surface grids
- Use Chimera Grid Tools (CGT) or commercial package
- All near-body volume grids concatenated into **grid.in**
- Reference: *W.M. Chan, R.J. Gomez III, S.E. Rogers, and P.G. Buning, "Best Practices in Overset Grid Generation," AIAA 2002-3191, June 2002*



Near-Body Grid Generation

- Distance off wall (**S**) and outer grid spacing (Δ**S**) contribute to the size of the off-body grids
  - We will refer to **S** and Δ**S**, and how they affect the grid generation process, in following sections
- One philosophy:
  - Grow volume grids out until grid cells are roughly square
  - Grow out a total distance **S** which is about 10 times the outer cell size Δ**S**
  - This will determine the off-body grid spacing, and will contribute to the required X-ray spacing as well





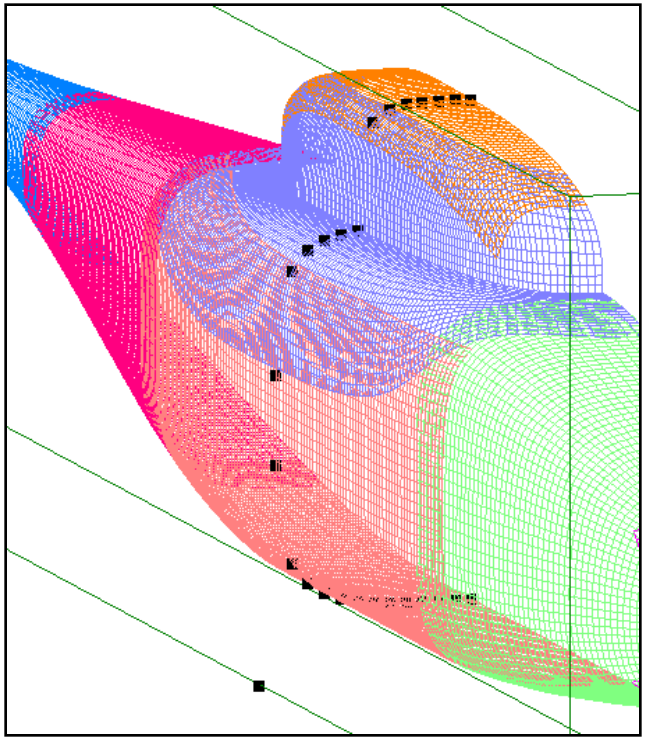
Creating X-Rays

- Creating X-rays
- Picking X-ray spacing
- Using OVERGRID to create X-rays
- X-ray number and Body ID
- Using **gen_x** to create X-rays
- Examples
- Notes and comments

Creating X-Rays

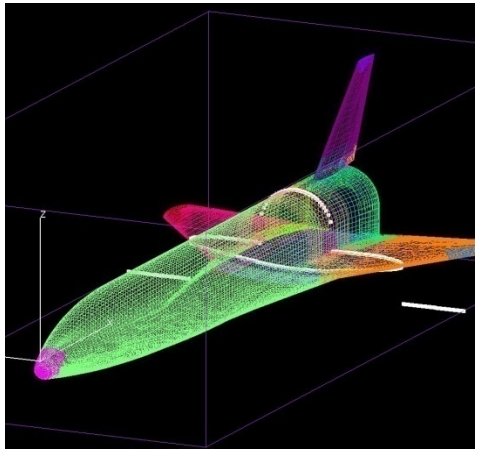
- An X-ray is an (x,y) array of z-value pierce-points of a body
  - These are used inside OVERFLOW for faster hole-cutting for grid connectivity
- Process relies (entirely) on Chimera Grid Tools (CGT)
- Create the **xrays.in** file before running OVERFLOW
- Use OVERGRID (interactive) or use **gen_x** (batch) in CGT

- Before you start, you will need to:
  - Generate a PLOT3D grid file of each body surface
  - Pick the X-ray spacing



Picking X-Ray Spacing

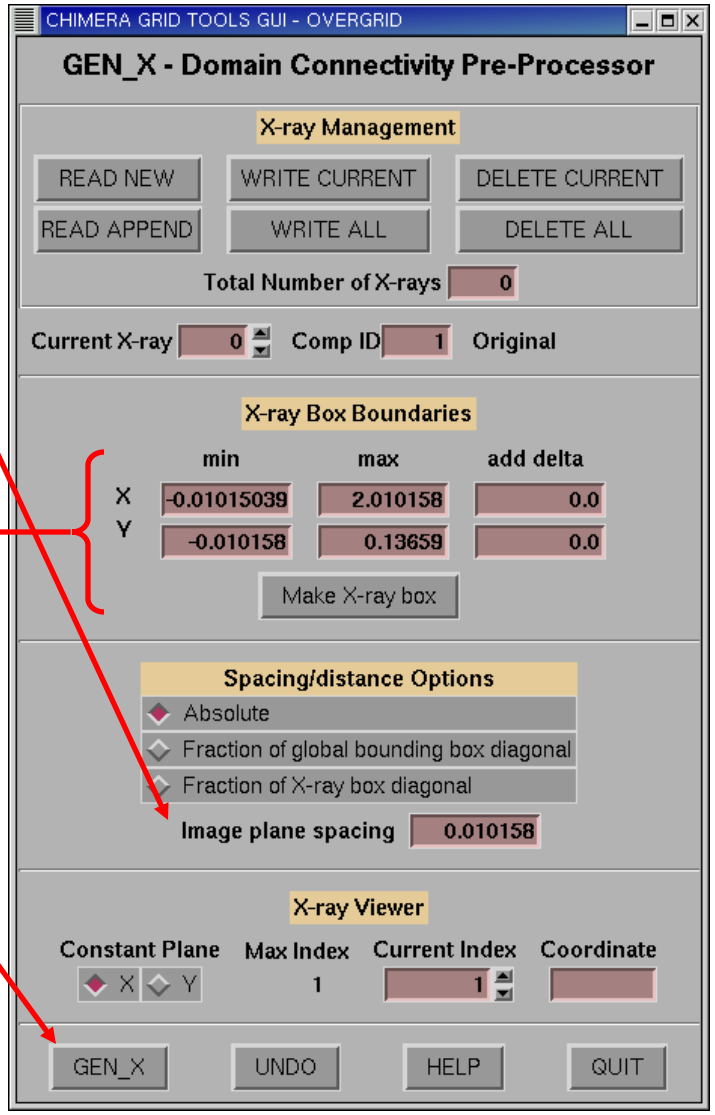
- This is the “resolution” of the body surface for the hole-cutting operation
  - The X-rays need to represent the body geometry sufficiently well to cut holes in other grids
- For single-body applications, use ½ to 1 times the outer cell size of the near-body grids (**ΔS**)
  - Too-fine X-ray spacing slows down hole-cutting (very important for moving-body problems)
  - X-rays take memory in the flow solver (proportional to spacing squared)



- For bodies in close proximity, use 0.1 to 0.2 times the distance between bodies
  - Can use different x-rays (with different spacing) for different regions
- X-rays used for collision detection also need higher resolution
  - More on this later…

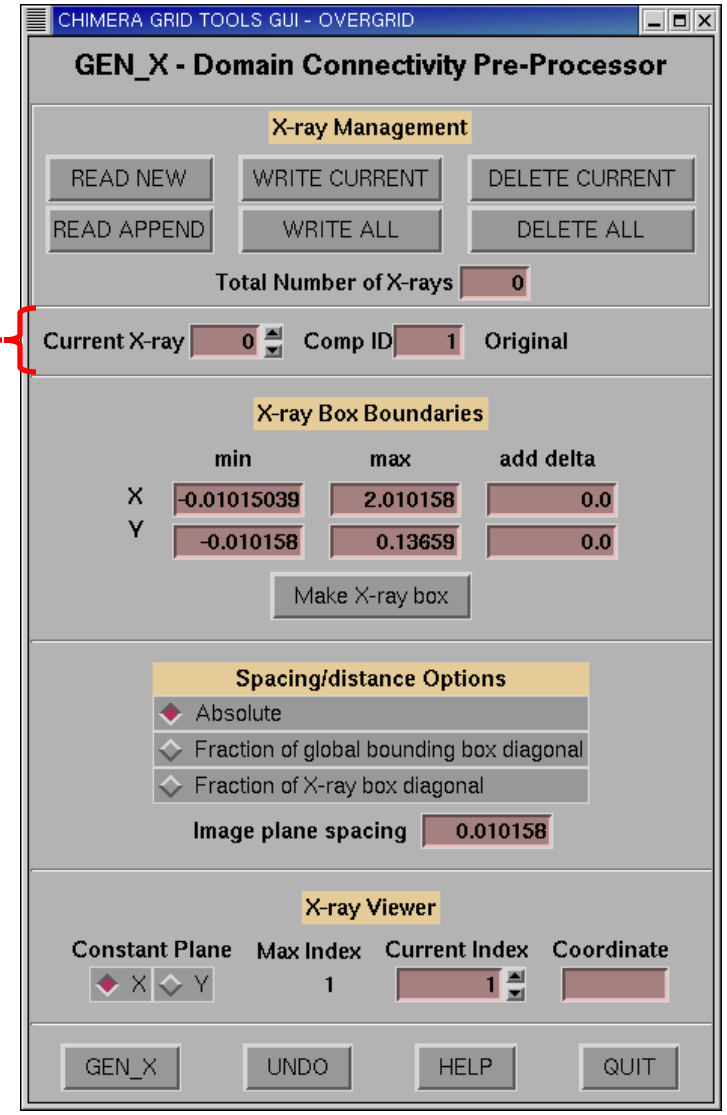
Using OVERGRID to Create X-Rays

- Enter X-ray spacing as “Image plane spacing”
  - Type <ENTER> to automatically adjust box boundaries
- Adjust box boundaries if needed
  - Ignore “add delta”
- Click “Make X-ray box”
- Click “GEN_X” to generate the X-rays
- Click “WRITE CURRENT” or “WRITE ALL” to save the X-rays to a file



X-Ray Number and Body ID

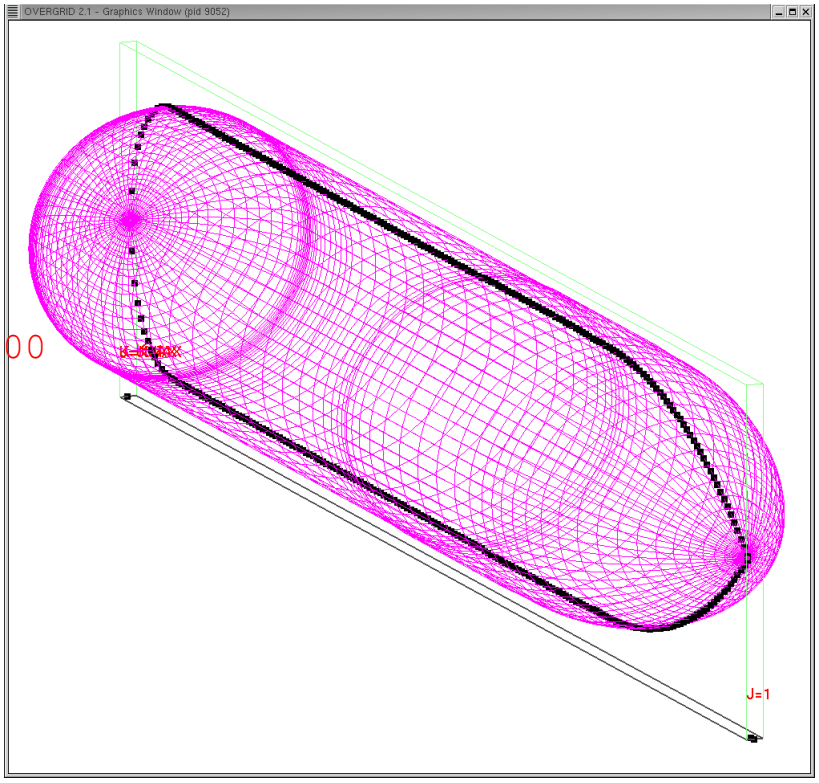
- X-rays are numbered sequentially and will be referred to by number in the OVERFLOW input
- Each X-ray is tied to a body, identified by “Comp(onent) ID” number (so when the body moves, the hole-cutting moves with it)
  - Body ID (Component ID) can be set here
  - Body ID=**n** refers to the **n**th component defined in the **Config.xml** file (discussed later)
- A text-input utility **xrayed** (part of CGT) allows manipulation of X-ray files
  - Combining X-ray files
  - Splitting files
  - Duplicating X-rays
  - Changing body IDs

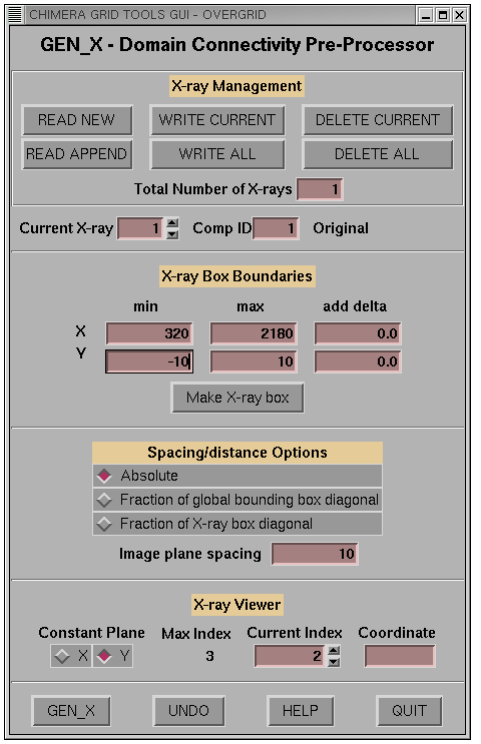


Example: Axisymmetric External Tank

- For 2D or axisymmetric geometries, X-rays only need to bound the center (y=0) grid plane
  - Create the surface grid to represent the geometry within ± the X-ray spacing of y=0
  - Set the X-ray bounding box y limits to ± the X-ray spacing
  - Comparable **gen_x** input:

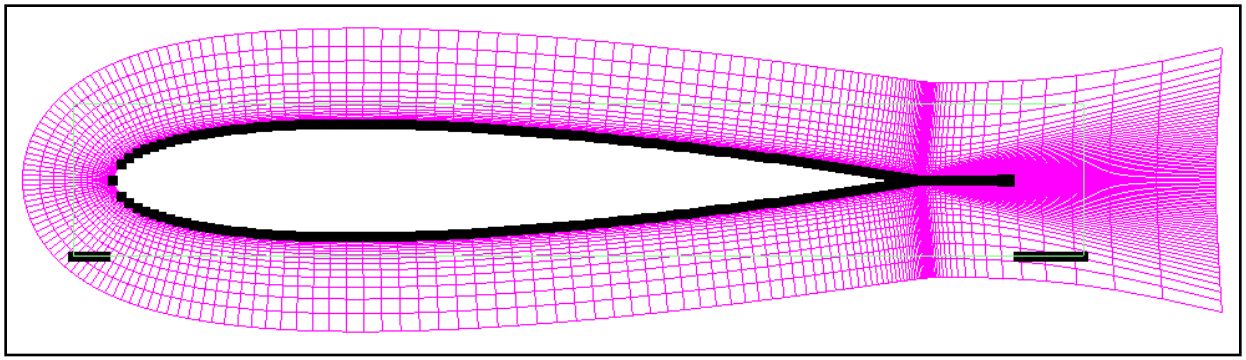
```
et.srf
1   ISOPT(1/2/3)
10  DS
0   DELTA
1   NCROPS
1   IDBODY
320,2180,-10,10,-200,200
```



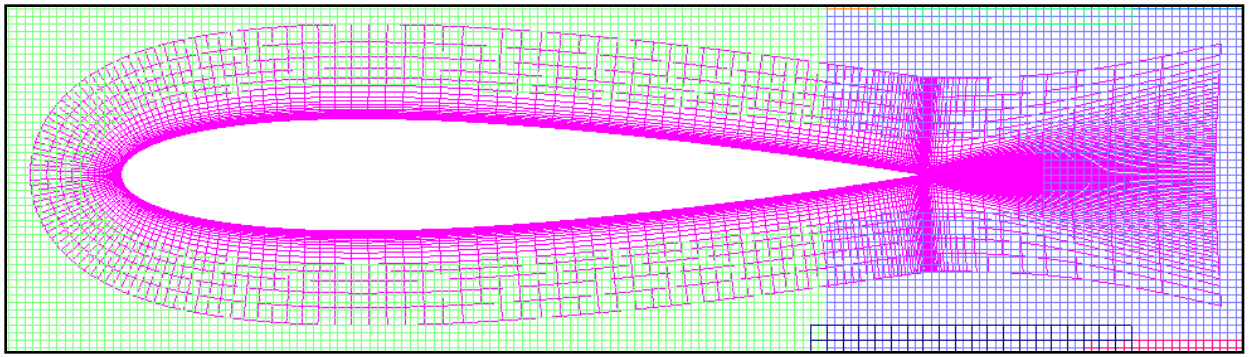


Example: 2D Airfoil

- For airfoils and wings, include a thin section of the C-grid wake with the surface grid
  - Use L=2 (or K=2) surface for finite thickness wake
  - Allows X-ray to cut other grids out of refined wake region



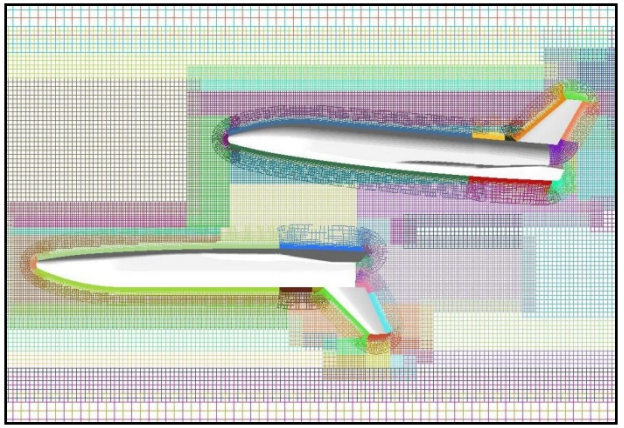
Airfoil grid and X-ray, showing extension into wake



Resulting hole in off-body grids

Notes and Comments

- "Duplicated" X-rays are useful in some cases
  - For example when multiple bodies are different only in position
  - Special format in X-ray file does not take additional space
  - X-rays can be duplicated using **xrayed** utility

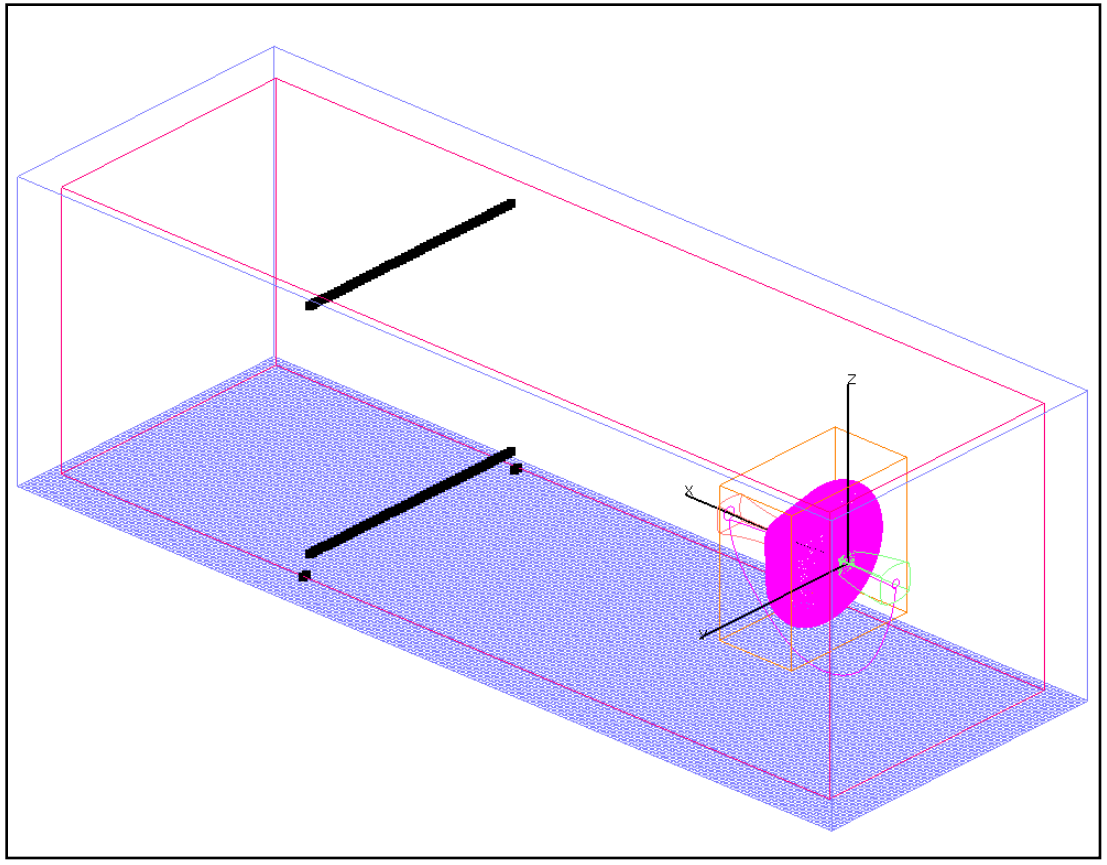


- Remember that when creating X-rays, surface grids for different bodies have to be in different files. Resulting X-ray files then have to be merged (again, using **xrayed**).

Notes and Comments

- If a user-generated box grid is added, an X-ray must be generated to cut off-body grids from the inside of the box
  - A surface grid file must be created using interior surfaces of the box grid, for example constant planes of J,K,L=8 and -8

Sample capsule plus wake box, with X-rays for the wake box. Capsule will cut a hole in the wake box; wake box will cut a hole in off-body grids.

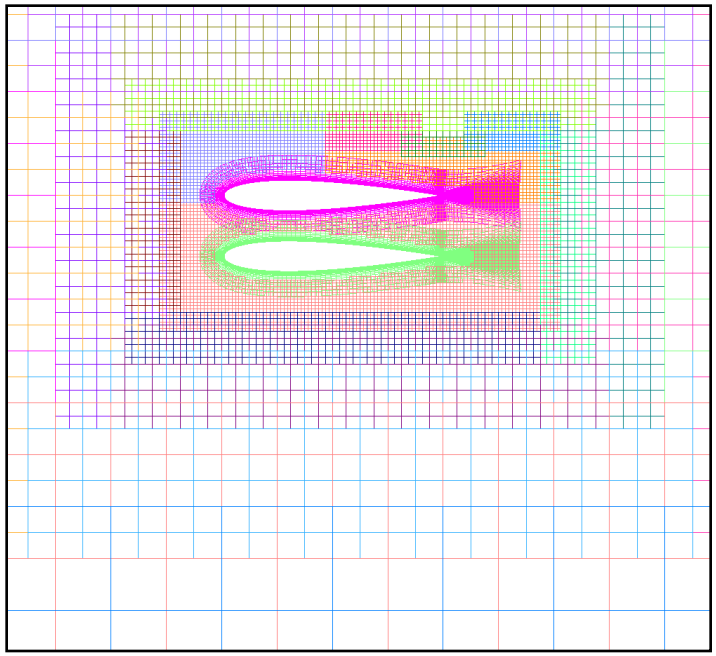


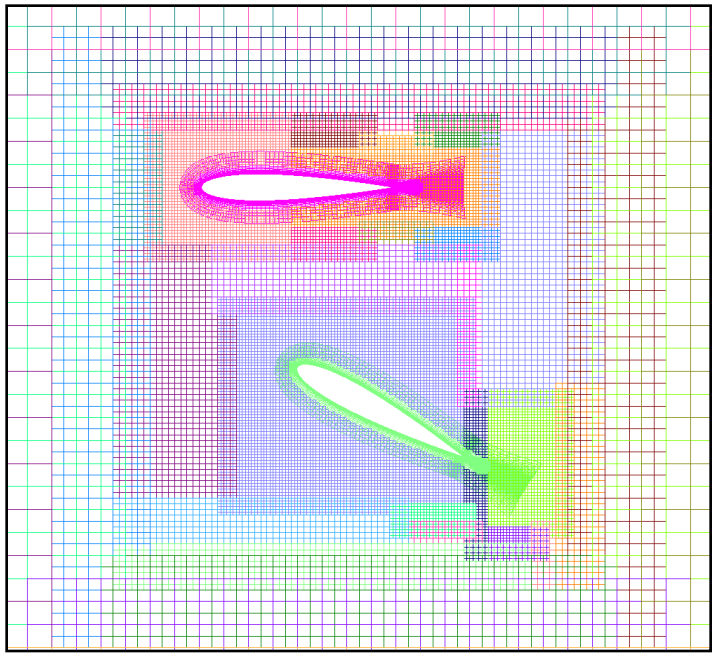
Automatic Off-Body Grid Generation

- Function of off-body grids
- Basic controls
- Matching near-body and off-body grid spacing
- Specifying additional refined regions
- Controlling the rate of grid coarsening
- Specifying symmetry planes, ground planes, etc.
- Far-field boundary conditions
- Examples
- Notes and comments

Function of Off-Body Grids

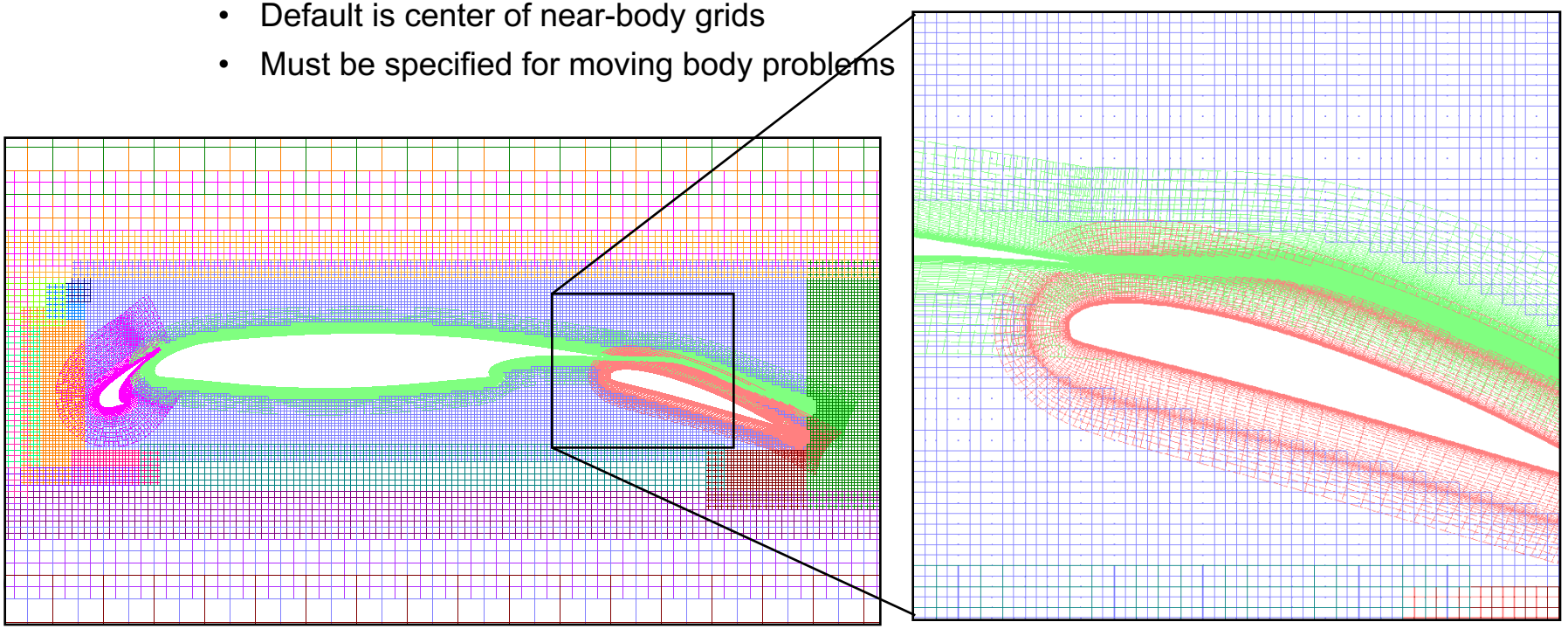
- Level-1 (finest) off-body grids:
  - Surround (all) near-body grids
  - Fill user-specified regions
  - Solution adaption (if used)
- Level-2 and coarser grids fill in to the far-field boundary





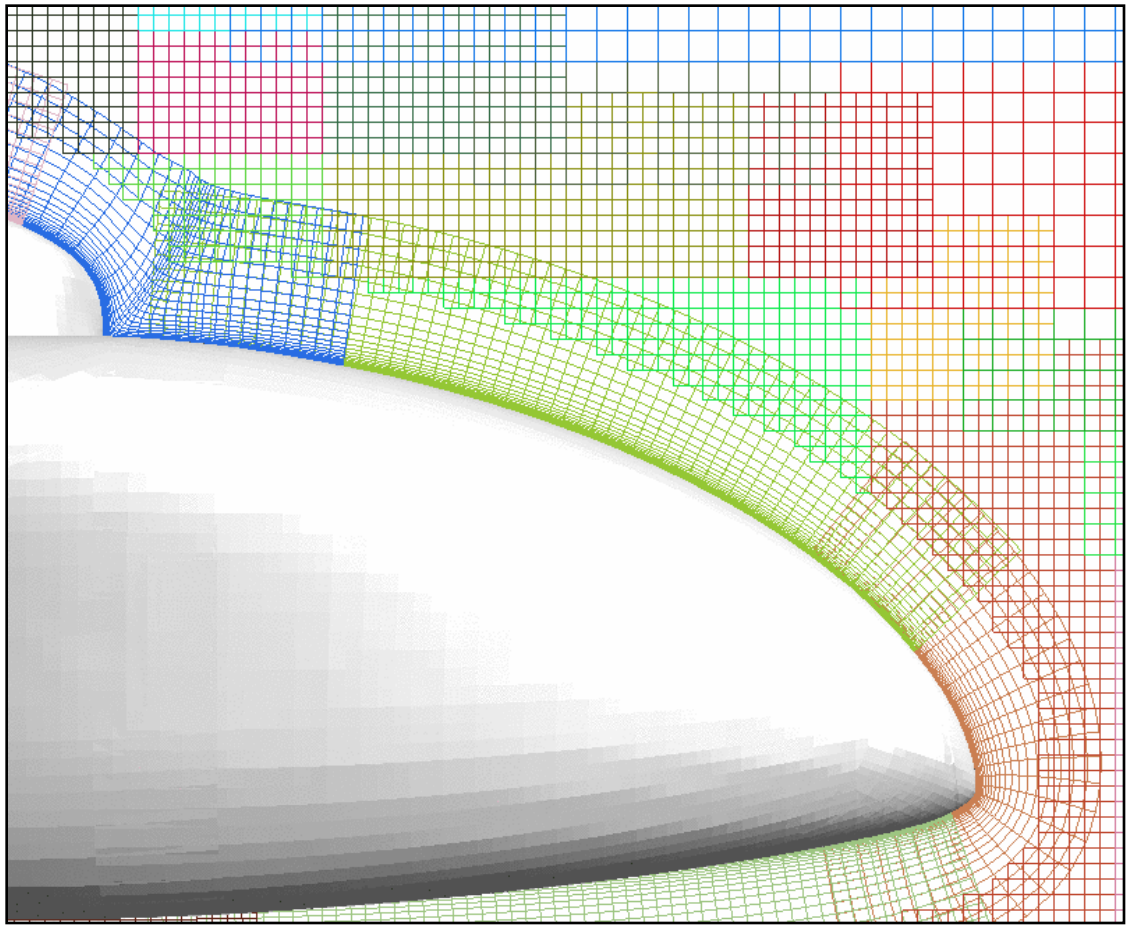
Basic Controls

- Basic controls (input parameters in **$GBRICK**):
  - **DS** – spacing for level-1 (finest) off-body grids
    - This parameter is critical for (a) proper communication with near-body grids, (b) resolving off-body flow gradients, and (c) controlling overall number of grid points.
  - **DFAR** – distance to (all) outer boundaries
  - **CHRLEN** – characteristic body length (*no longer used*)
    - Default is 1, use (major) dimension of body
  - **XNCEN,YNCEN,ZNCEN** – center of off-body grid system
    - Default is center of near-body grids
    - Must be specified for moving body problems



Matching Near-Body and Off-Body Grid Spacing

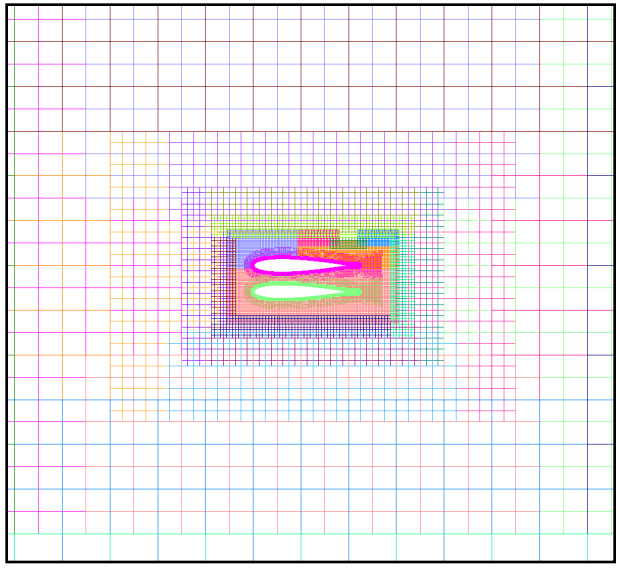
- How to pick **DS** (or, how far to grow near-body grids)?
  - **DS** should match **ΔS** (outer boundary spacing of near-body grids)
  - **DS** (and **ΔS**) should be sized to resolve off-body flow gradients
  - Near-body grids should extend out about **10xDS** from the body surface



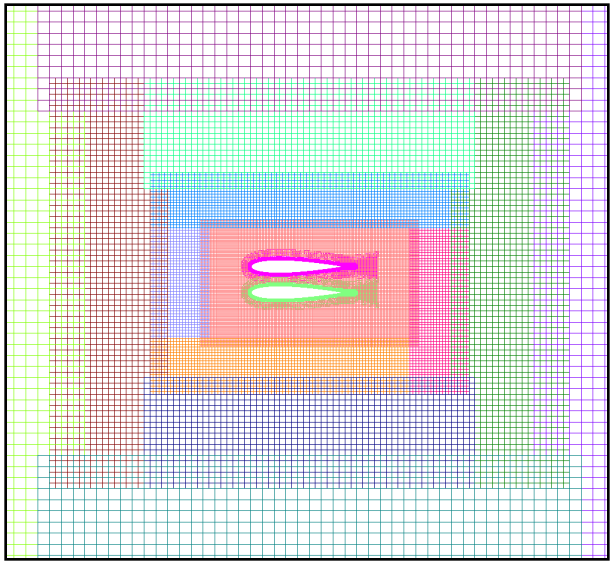
Controlling the Rate of Grid Coarsening

- Effect of **MINBUF** (in **$GBRICK**):
  - Default **MINBUF=4** gives minimum overlap between successively coarser off-body grids
  - Larger values give more gradual coarsening, but use more grid points
  - 2-airfoil example:
    - **MINBUF=4** (if geometry were 3D, off-body grids would have 2 million points)
    - **MINBUF=8** (3D off-body grids would have 3 million points)

MINBUF=4

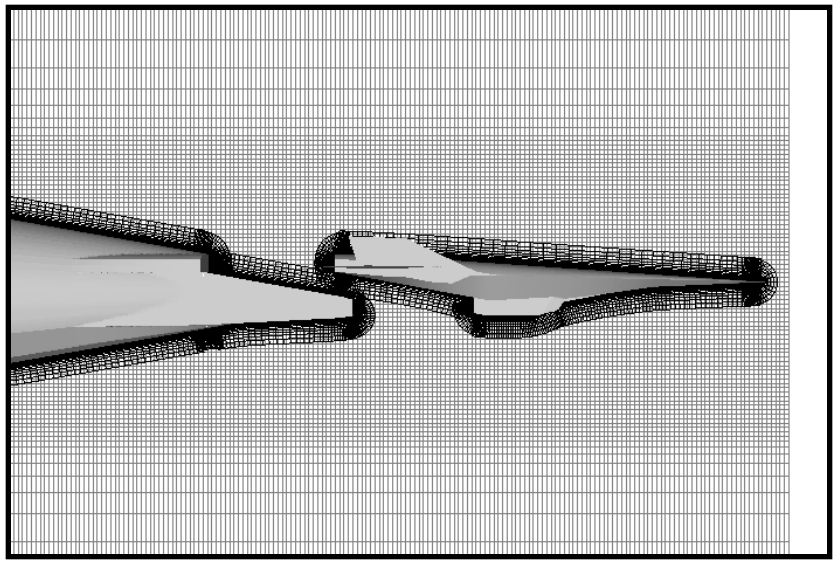


MINBUF=8

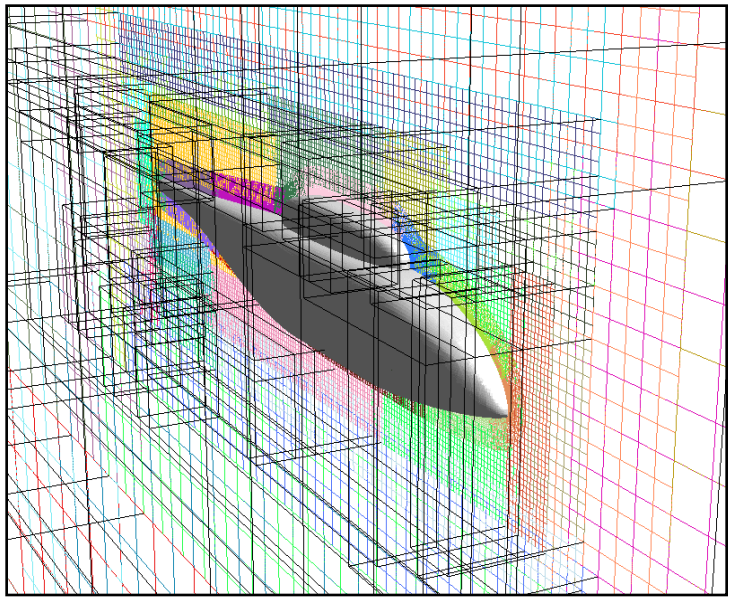


Specifying Symmetry Planes, Ground Planes, etc.

- Special planes (input parameters in **$GBRICK**):
  - Used to set a ground plane, symmetry plane, inflow plane, etc.
  - **I_XMIN=1** – use value of **P_XMIN** as off-body grid X(minimum)
  - **I_XMIN=0** – default is to use **DFAR** to set X (minimum)
  - Same for **I_XMAX, I_YMIN,I_YMAX, I_ZMIN,I_ZMAX,** and **P_XMAX, P_YMIN,P_YMAX, P_ZMIN,P_ZMAX**
  - Can only set one out of each (x,y,z) pair of values



Hyper-X supersonic inflow plane:
I_XMIN=1, P_XMIN=-20



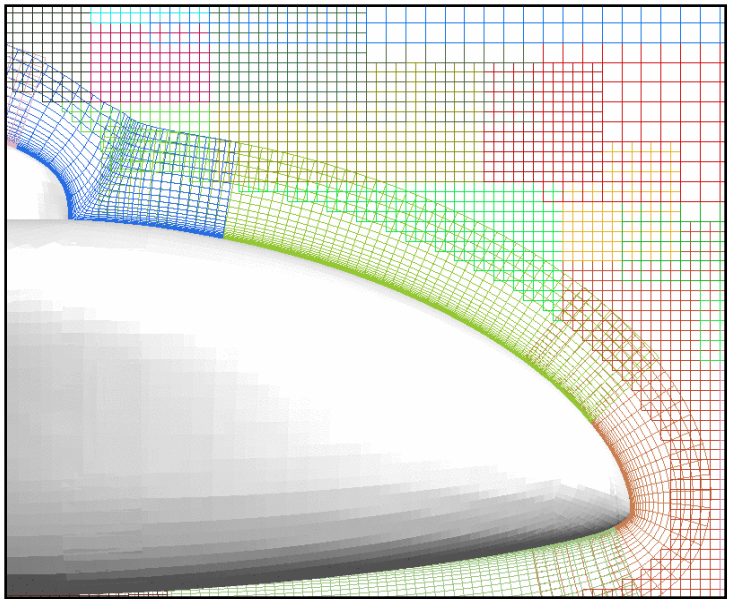
Symmetry plane for helicopter fuselage:
I_YMIN=1, P_YMIN=0

DCF: Hole Cutting and Grid Assembly

- Using X-rays to cut holes
- Choosing **XDELTA**
- Orphan points and donor quality
- Double fringe interpolation
- Viscous stencil repair
- Examples
- Notes and comments

Using X-Rays to Cut Holes

- Specifying X-ray cutters (input parameters in multiple **$XRINFO**):
  - **IDXRAY** – X-ray number
  - **IGXLIST** – list of grids to be cut
    - Special grid number "-1" refers to (all) off-body grids
  - Or use **IGXBEG,IGXEND** – starting/ending grids to be cut
  - **XDELTA** – offset of hole from body surface
- Example: $XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.05 $END
  - Use the first X-ray in **xrays.in** file, cut a hole in the off-body grids, 0.05 grid units off the X-ray surface:



Using X-Rays to Cut Holes

- Example: multi-element airfoil

  $XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.02 $END

  $XRINFO IDXRAY=2, IGXLIST=-1, XDELTA=0.02 $END

  $XRINFO IDXRAY=3, IGXLIST=-1, XDELTA=0.02 $END

  – Slat, main, and flap X-rays (X-rays 1,2,3) cut holes in off-body grids

  $XRINFO IDXRAY=1, IGXLIST=2, XDELTA=0.005 $END

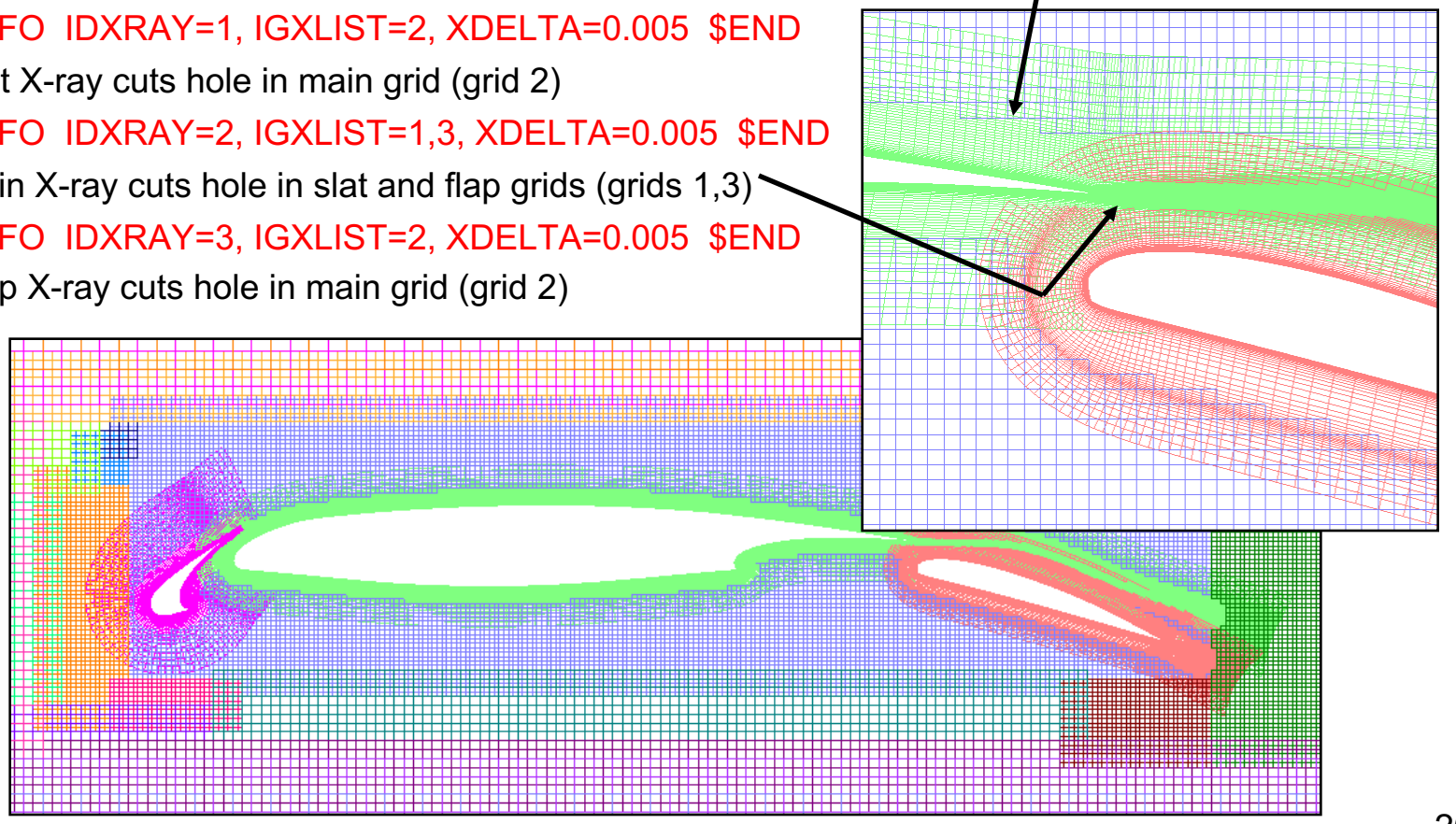
  – Slat X-ray cuts hole in main grid (grid 2)

  $XRINFO IDXRAY=2, IGXLIST=1,3, XDELTA=0.005 $END

  – Main X-ray cuts hole in slat and flap grids (grids 1,3)

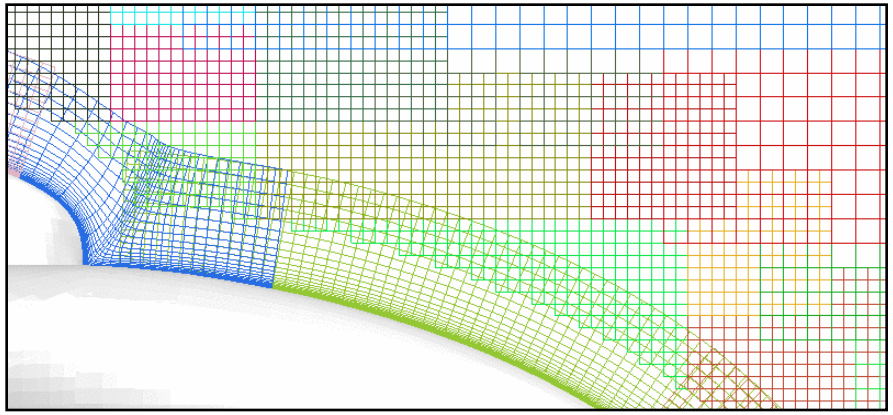
  $XRINFO IDXRAY=3, IGXLIST=2, XDELTA=0.005 $END

  – Flap X-ray cuts hole in main grid (grid 2)



Choosing **XDELTA**

- Holes should be cut to keep coarser grids out of high-gradient regions (such as boundary layers)
- Holes should be cut so that grids have similar resolution in overlap regions, and have sufficient overlap for interpolation of boundary data
- When cutting holes in off-body grids, choose **XDELTA** to be 5 times **DS**, *in from the outer boundary* of the near-body grids, or **XDELTA = S – 5xDS**
  - This is often about half the distance to the surface



- When cutting holes in nearby bodies, **XDELTA** must be less than half the expected minimum distance between the bodies to avoid orphan points
  - Can use different values for different cutters

Orphan Points and Donor Quality

- Some overset grid definitions (thanks to Ralph Noack):
  - **Blanked-out points** – points inside bodies or holes, where the solution is not computed or is ignored
  - **Fringe points** – inter-grid boundary points where solution values are obtained via interpolation from another grid
  - **Donor points** – points contributing to interpolation stencils
  - **Orphan points** – fringe points without valid donors; resulting from hole cutting failure (no possible donor) or only poor quality donors are available (insufficient overlap)
- Donor stencil quality (input parameter in **$DCFGLB**):
  - "Quality" of the donor stencil refers to how much of the interpolated information has to come from donor points that are interior to the flow solution, i.e., not fringe points themselves
  - **DQUAL=1** – donor stencils must consist of only field points (default)
  - **DQUAL=0** – stencils which include all fringe points may be accepted
    - This is *not* a good idea—the simulation may simply pass boundary data back and forth between grids
  - **DQUAL=0.1** is generally acceptable

Viscous Stencil Repair

- Viscous stencil repair (input parameters in **$DCFGLB**):
  - **MORFAN** – enable/disable viscous stencil repair (1/0)
  - **NORFAN** – number of points above a viscous wall subject to viscous stencil repair
  - Viscous stencil repair is needed to handle bad interpolations when overlapping surface grids lie on the same curved surface. If not corrected, this can result in orphan points (convex surfaces) or interpolations too high in the boundary layer (concave surfaces).
  - WARNING: Interpolation stencils for boundary points within NORFAN points of a viscous surface will be modified, using the assumption that all viscous walls have the same grid distribution in the normal direction. QUALITY OF REPAIRED STENCILS IS NOT CHECKED.
  - A better scheme is needed!

DCF Output

- Output from DCF process indicates the number of double fringe and viscous stencils repaired, and the final number of orphans

```
..............    START DCFCRT    ..................


  WARNING:  USING VISCOUS STENCIL REPAIR WITHIN     6 POINTS OF A WALL.
    Interpolation stencils for boundary points within NORFAN points of a
    viscous surface will modified, using the assumption that all viscous
    walls have the same grid distribution in the normal direction.
  WARNING:  QUALITY OF REPAIRED STENCILS IS NOT CHECKED.

  WARNING:        278 viscous stencils/orphans repaired in DCFCRT
  NO orphan points found in DCFCRT

   ORPHAN POINT SUMMARY:
     *Numbers are approximate due to grid splitting.
      Points in overlap region may be counted twice.

            Initial  Visc Stencils   Visc Orphans  Double Fringe     Final
    Grid    Orphans    Repaired        Repaired   Orphs Repaired    Orphans
    ------------------------------------------------------------------------
       1*        0        214              0              0              0
       2*       36         28             36              0              0

..............     END DCFCRT     ..................
```

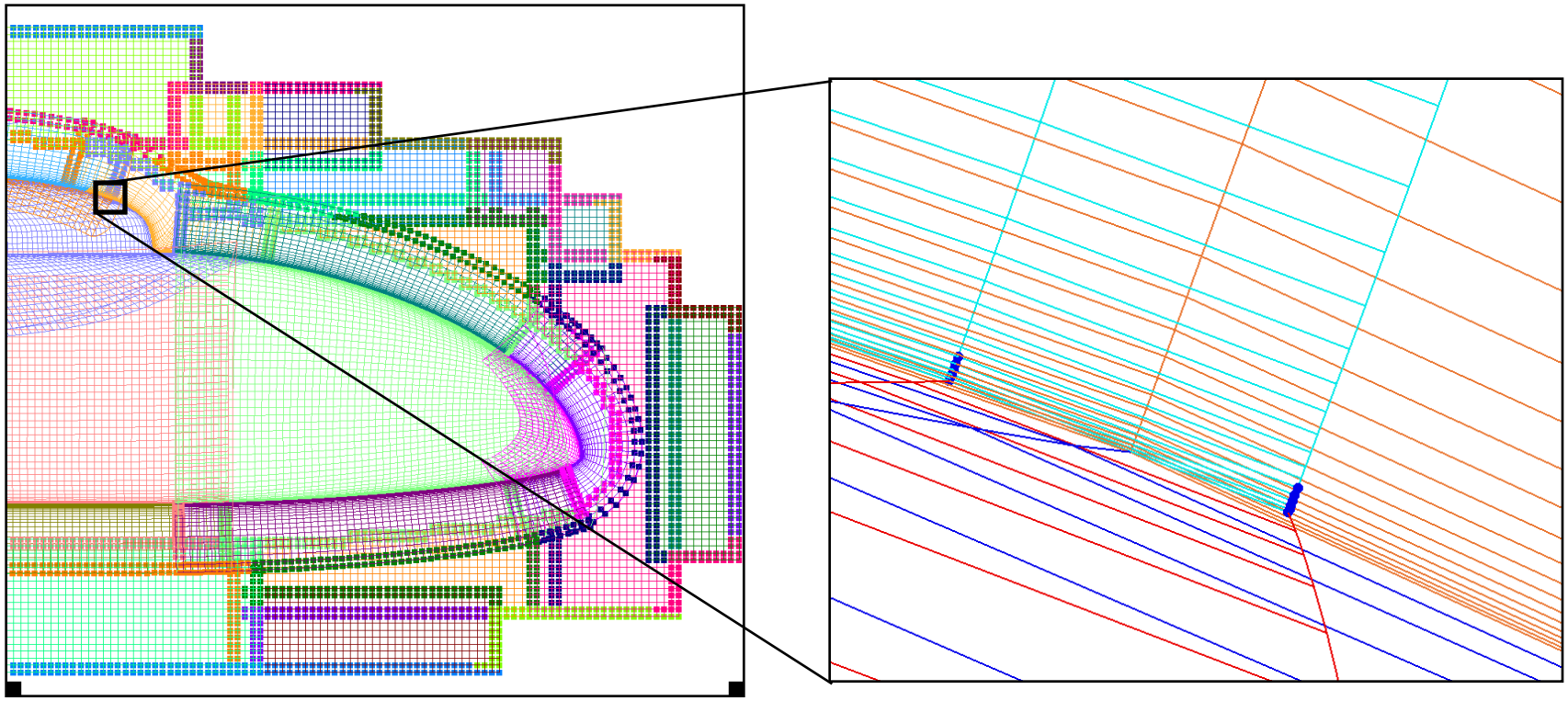
Example 1

- Helicopter fuselage

  $OMIGLB LFRINGE=2, … $END

  $DCFGLB DQUAL=0.3, MORFAN=1, NORFAN=6, $END

  $XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.035, $END

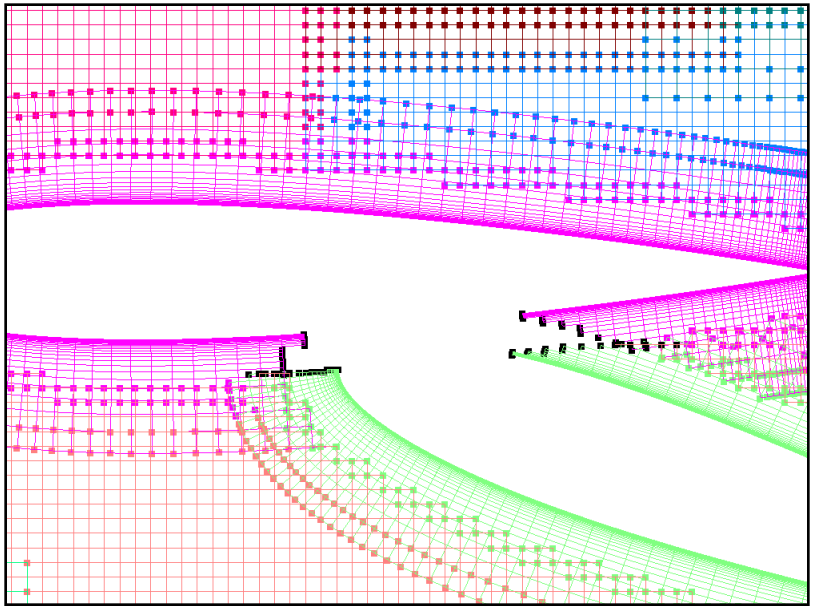


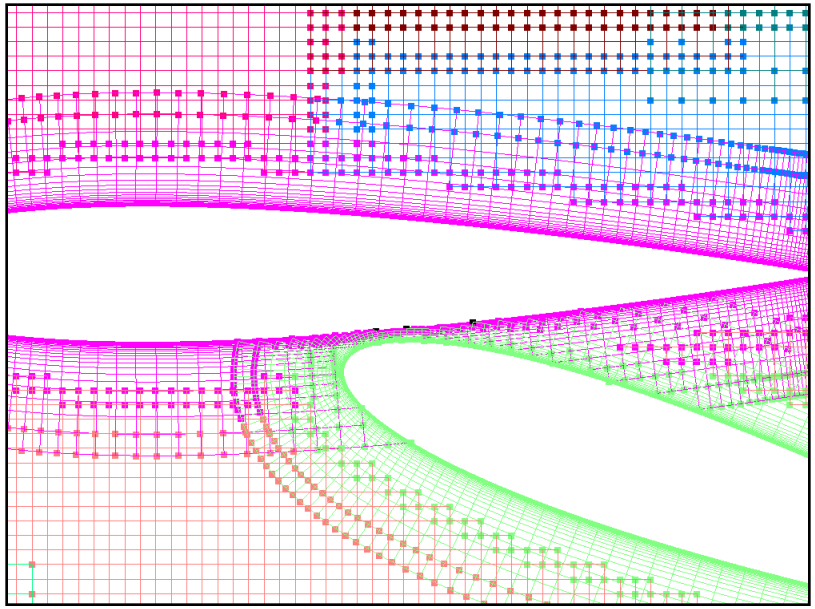
Example 2

- Airfoil drop
  - For bodies that are very close to each other, very small values of **XDELTA** may be needed

```
$OMIGLB  LFRINGE=2, …  $END
$DCFGLB  DQUAL=0.3,  $END
$XRINFO  IDXRAY=1, IGXLIST=2,-1, XDELTA=0.04,  $END
$XRINFO  IDXRAY=2, IGXLIST=1,-1, XDELTA=0.04,  $END
```

```
$OMIGLB  LFRINGE=2, …  $END
$DCFGLB  DQUAL=0.3,  $END
$XRINFO  IDXRAY=1, IGXLIST=-1, XDELTA=0.04,  $END
$XRINFO  IDXRAY=2, IGXLIST=-1, XDELTA=0.04,  $END
$XRINFO  IDXRAY=1, IGXLIST=2, XDELTA=0.0,  $END
$XRINFO  IDXRAY=2, IGXLIST=1, XDELTA=0.0,  $END
```





Notes and Comments

- It is OK to have “some” orphan points
  - But you should understand why, and where they are in the grid system
  - Be careful of compromising grid quality because you don’t want to refine the off-body grids, or don’t want to fix the near-body grids
- Orphan points become much harder to control in moving body problems
  - Have to anticipate grid movement
- OVERFLOW “fills” orphan points (and *all* hole points) with average of neighboring point values
- Input parameter **IRUN** in **$OMIGLB** allows test run of DCF:
  - **IRUN=1** – just do off-body grid generation (write **x.save** file)
  - **IRUN=2** – do off-body grid generation *and* DCF (write **x.save**)
  - **IRUN=0** – do a complete run, including flow solver
  - When changing inputs, be sure to delete **brkset.restart** and **INTOUT**, or OVERFLOW will not rerun these steps

Data Surface Grids

- Can be used to extract acoustic data surfaces, velocity profiles, pressure tap locations, 2D slices, etc.
- Any “1D” or “2D” (*m*x1x1 or *m*x*n*x1) grid in the **grid.in** file will be treated by DCF as a “data surface grid”
  - Flow solution at all points will be interpolated from other grids
  - Grid and solution will be saved in usual files (**x.save**, **q.save**)
  - Can also write these out explicitly using **$SPLITM**

OVERFLOW-D Mode With Grid Motion

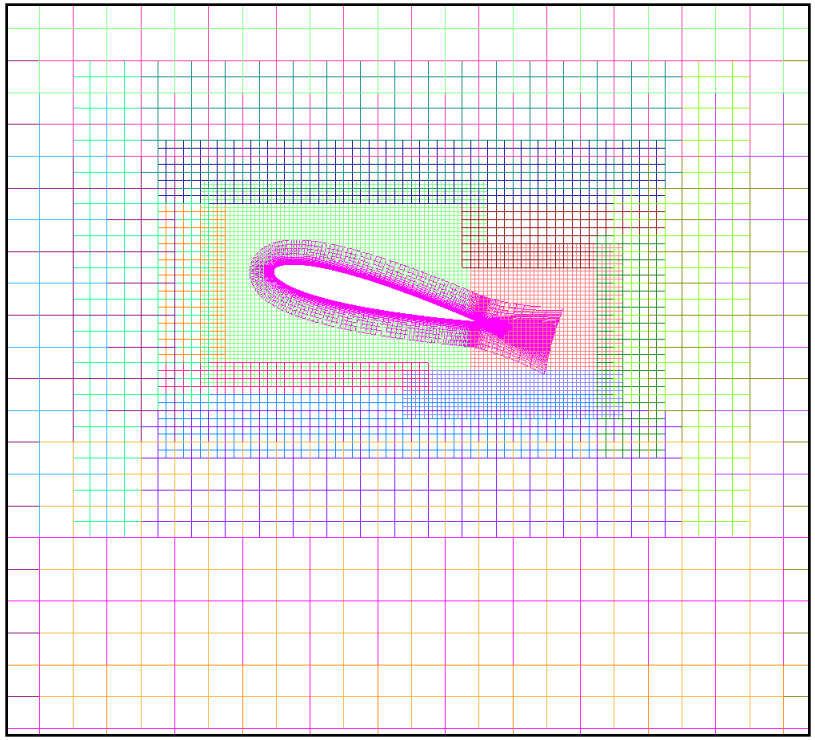
- General moving body process
- Off-body grid adaption to geometry
- GMP files **Config.xml** and **Scenario.xml**
- Non-dimensionalization of dynamics quantities
- Time step specification
- Simulating collisions
- Output information for moving body problems
- Visualizing body motion in OVERGRID
- Some references

General Moving Body Process

- Current recommendation: **run OVERFLOW in double precision**
  - Quaternion variables need to be stored as 64-bit (most, but not all complete)
- General process (input parameters in **$OMIGLB**):
  - **DYNMCS=.TRUE.** – enable body dynamics (default is FALSE)
  - **I6DOF=2** – Prescribed and/or 6-DOF motion for different components. Specified via the GMP interface (**Config.xml** and **Scenario.xml** files) (**$SIXINP** is ignored). **This is the recommended (and supported) option for moving body problems.**
    - **I6DOF=1** – 6-DOF body motion, specified via **$SIXINP** namelist input.
    - **I6DOF=0** – User-specified motion, controlled by user-supplied USER6 subroutine.
  - **NADAPT** – number of steps between adaption (regeneration) of the off-body grid system
    - **NADAPT=-n** – off-body grids adapt to geometry only
    - **NADAPT=0** – off-body grids will not be regenerated during solution process
    - **NADAPT=n** – off-body grids adapt to geometry and flow solution (see next section)

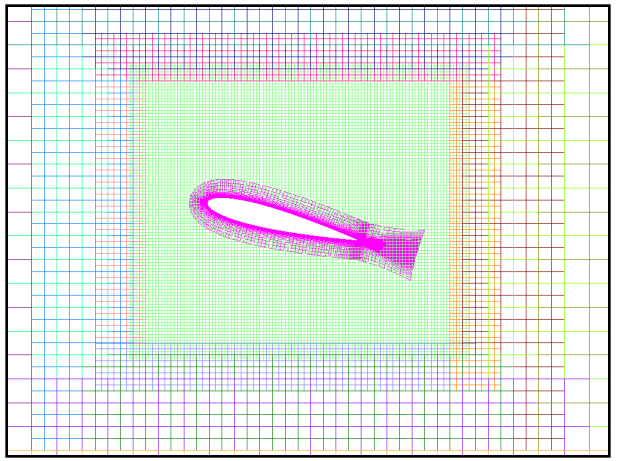
General Moving Body Process

- DCF (hole-cutting and interpolation stencil-finding) is done every step
  - Want interpolation stencils to change less than one cell per step for time accuracy
  - Estimate maximum velocity of fringe points and compare to donor grid cell size
  - This sets maximum desired physical time step
- Look at a simple example:
  - We have level-1 boxes
  - We have a near-body grid inside the boxes
  - The body is moving, the boxes are not
- What happens in OVERFLOW?
  - Body motion is computed
  - Body is moved
  - DCF is performed
  - Flow solution is advanced



Off-Body Grid Adaption to Geometry

- As body moves, near-body grid gets close to the edge of the level-1 boxes
- Off-body grids must be regenerated, and the flow solution transferred (interpolated) to the new off-body grids
- **NADAPT=-n** gives the number of time steps between off-body grid adaption
  - Usually every 20-50 steps (based on time step **DTPHYS** and **MINBUF**)
    - Check this by running sample cases
  - In some cases we can avoid this by creating a larger level-1 grid (e.g., for pitching airfoil problem)



Geometry Manipulation Protocol Files: Config.xml

- Defines body (component) names and the associated grids
- Specifies any initial body transforms to assemble components into their starting positions
- Components and their transforms can be defined hierarchically
- Example:

```
<?xml version='1.0' encoding='utf-8'?>
<Configuration AngleUnit="degree">
   <Component Name="wing" Type="struc">
      <Data> Grid List=1-5 </Data>
   </Component>
   <Component Name="aileron" Parent="wing" Type="struc">
      <Data> Grid List=6,7 </Data>
      <Transform>
         <Rotate Center="0.7,0,0" Axis="0,-1,0" Angle="10"/>
      </Transform>
   </Component>
</Configuration>
```

GMP Files: **Scenario.xml**

- Prescribed motion: specify
  - Start time and duration
  - Translation and rotation rates
- Example (prescribed motion):

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario Name="Forced Oscillation" AngleUnit="degree">
   <Prescribed Component="aileron" Start="0", Duration="0">
      <Rotate Center="0.7,0,0" Axis="0,-1,0"
       Speed="20.*2.*pi/100.*cos(2.*pi/100.*t+pi/2.)"/>
   </Prescribed>
</Scenario>
```

GMP Files: **Scenario.xml**

- 6-DOF motion: specify
  - Start time and duration
  - Component inertial properties
  - Applied forces
  - Motion constraints
- Example (constrained 6-DOF motion):

```
<?xml version='1.0' encoding='utf-8'?>
<Scenario Name="Constrained Motion" AngleUnit="degree">
   <Aero6dof Component="aileron" Start="0", Duration="0">
      <InertialProperties Mass="1.0" CenterOfMass="0.7,0,0"
       PrincipalMomentsOfInertia="0,2,0"/>
      <Constraint Rotate="1,0,1" Frame="body" Start="0"/>
      <Constraint Translate="1,1,1" Frame="body" Start="0"/>
   </Aero6dof>
</Scenario>
```

GMP Files: **Config.xml** and **Scenario.xml**

- For 6-DOF problems, GMP component names *must* match component names in **mixsur.inp** (force and moment calculation)
- GMP files can be created in a text editor or using OVERGRID
  - Be careful that the motion illustrated in OVERGRID is the same as that in OVERFLOW (should be OK with CGT 2.1)
- Some GMP capabilities do not work in OVERFLOW:
  - Principal axes not aligned with the original (x,y,z) axes in **grid.in**
  - Moments of inertia do not change with moving parts

Non-Dimensionalization of Dynamics Quantities

- This is critical!
- Non-dimensionalizations in the flow solver are easy (free-stream density $\rho^*_\infty=1$, free-stream speed-of-sound $c^*_\infty=1$)
- Non-dimensionalizations for all dynamics and time-accurate information are based on V_{ref} rather than c_∞
  - V_{ref} is defined as V_{ref}=**REFMACH***c_∞
  - V_{ref} is the same as V_∞ if **REFMACH** is not explicitly specified in **$FLOINP**
    - **REFMACH** defaults to **FSMACH**
    - **REFMACH** may be different from **FSMACH**, for example for hover problems (**FSMACH=0**)
  - This includes **DTPHYS**; all quantities in GMP files (or **$SIXINP**); and output forces and moments, velocities and angular rates
  - Some quantities can be very large (or small)

Non-Dimensionalization of Dynamics Quantities

- Non-dimensionalizations of dynamic quanities are thus based on
  - Length: L=1 grid unit
  - Time: L/V_{ref}
  - Mass: $\rho_\infty L^3$
- Indicating non-dimensional quantities with a *:
  - Length: len* = len / L
  - Mass: m* = $m / (\rho_\infty L^3)$
  - Velocity: V* = V / V_{ref}
  - Time: t* = $t (V_{ref}/L)$
  - Acceleration: a* = $a (L/V_{ref}^2)$
  - Force: F* = $F / (\rho_\infty V_{ref}^2 L^2)$
  - Moment of inertia: I* = $I / (\rho_\infty L^5)$
  - Angular velocity: ω* = $\omega (L/V_{ref})$
  - Moment: M* = $M / (\rho_\infty V_{ref}^2 L^3)$