4. OVERFLOW-D-Mode Operation Without Grid Movement

Two features distinguish "OVERFLOW-D-mode" from a traditional OVERFLOW run. The first is the internal generation of off-body Cartesian grids, used to fill up the computational domain not covered by the near-body grids supplied in **grid.in**. The second is the assembly of the overset grid system, including cutting holes and finding appropriate interpolation stencils from the overlapped grid boundary points ("inter-grid boundary points"). Here this is done internal to OVERFLOW using a routine called DCF (Domain Connectivity Function), rather than an external program such as PEGASUS 5 or SUGGAR.

This section describes a typical run sequence with OVERFLOW 2.3 in the OVERFLOW-D-mode without grid motion. The code goes through three main steps before reaching the flow solver. First, it generates the overlapping Cartesian off-body grid system. Second, if the run uses MPI parallelization, OVERFLOW splits grids for load balancing and then assigns groups of grids to individual MPI processes. Each group is run in a separate process, with MPI calls facilitating communication between the groups. Third, hole-cutting and interpolation for the overlapping grids are determined using the domain connectivity software DCF.

Before proceeding it would be worthwhile to specify some overset grid definitions:

- **Field points** – points on which the differential equations will be solved.
- **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. Single-fringe points have only one layer of interpolation points at a boundary. Double-fringe points have two layers of interpolated points at a boundary. Double-fringe points are the default in the code and allow 2nd- and 3rd-order flux algorithms to maintain their full five-point stencil across interpolated boundaries. The 5th-order schemes require a triple-fringe to maintain their seven-point stencil at interpolated boundaries. Reducing the number of fringe layers will result in the flux algorithm losing accuracy at interpolated boundaries.
- **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). In OVERFLOW, solution values at orphan points are set by averaging values from neighbors in the computational grid.
- **Quality** – 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.

The off-body Cartesian grid system is generated based on inputs read from **&GBRICK** and **&BRKINP**. The load-balancing of the grids for parallel processing is based on parameters from **&GROUPS**. DCF cuts holes based on the X-rays provided in **&XRINFO**. The indices and interpolation coefficients for the overlapping regions of the grid system are also determined by DCF based on the **&DCFGLB** NAMELIST input. DCF writes the grid connectivity file **INTOUT**. The code output will specify the number of points for which it cannot find donor points (orphans) for each grid. The orphan points can be visualized using "Show Fringe Points" in the **overgrid** graphical interface to Chimera Grid Tools. The goal of the grid assembly process is to minimize or eliminate orphans points. Two common reasons for orphan points are misaligned surfaces, particularly in viscous layers, and insufficient grid overlap. If the **INTOUT** file exists, the code will read the file and continue with the flow solver. Each of these steps will be examined in the following sections.

The following files are used for running in OVERFLOW-D-mode:

1. **grid.in** – grid file (required)
2. **over.namelist** – NAMELIST input file (required), described in Chapter 3
3. **xrays.in** – hole-cutting definition file (required for hole-cutting only)
4. fomoco or usurp files described in the Chapter 3 (required for force/moment integration only)
5. **Config.xml** – body properties and positioning file for Geometry Manipulation Protocol (GMP) (required if **&OMIGLB** input **I6DOF**=2). The GMP interface is described in Chapter 5.
6. **Scenario.xml** – prescribed or 6-degree-of-freedom motion file for GMP (required if **&OMIGLIB** input **I6DOF**=2 and this is a moving-body run: **DYNMCS**=.TRUE.)

4.1 Near-Body Grid Generation

The first step in preprocessing is to generate a near-body grid system. The near-body grids should completely contain the boundary layer on the body and provide sufficient overlap with the off-body grids for overset interpolation.

Ref. 1 contains suggestions for generating the near-body overset grid system. One philosophy for near-body grid generation is to grow the volume grid system out until the distance from the wall is approximately 10 times the outer cell size. It is desirable to have the outer grid cells approximately square to help match the grid resolution of the off-body Cartesian grids generated by OVERFLOW 2.3. An example near-body grid system is shown in Fig. 4.1.

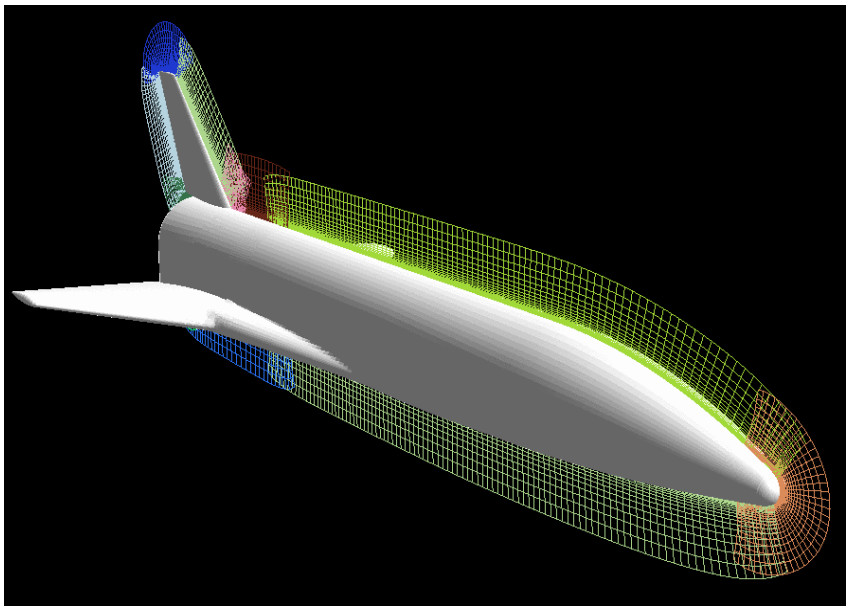


Figure 4.1 Example near-body grid system.

When running in OVERFLOW-D-mode, "data surface grids" can also be included in **grid.in**. Data surface grids are any grids with dimension *mxnx1*, and will receive interpolated data from other grids in the computational domain. These grids can be used to extract data at specific locations for post-processing applications like acoustic analysis; comparison with experimental data like velocity profiles, pressure tap data, or PIV data; or for plotting two-dimensional slices of a complex grid system. Grid motion can be controlled just like any other near-body grid. Interpolation stencils are found using DCF in the same manner as Chimera boundary points on overset grids. Flow solution data for data surface grids is automatically included in any **q.save** type file written by OVERFLOW. Data surface grids can also be explicitly written using **&SPLITM** namelists, just like any other near-body grids.

4.2 Off-Body Grid Generation

Off-body Cartesian grids are used to surround all of the near-body grid system. The methodology for off-body grid generation is described in detail in Ref. 2. Off-body grids are generated if the **&GBRICK** NAMELIST input **OBGRIDS**=.TRUE.. For single-grid cases or when outer grids are included in the **grid.in** file, no off-body grids need to be generated and **OBGRIDS** should be set to .FALSE.. If the file **brkset.restart** is present, off-body grid information is read from this file. Otherwise the off-body grids are generated and saved to **brkset.restart**. (OVERFLOW runs with grid adaption will write new off-body grid information to **brkset.save**; the **overrun** script will move **brkset.save** to **brkset.restart** before starting the run.)

The Cartesian grids that surround the near-body grids are referred to as level-1 grids. Additional regions to be covered by level-1 grids can be specified using input in NAMELIST **&BRKINP.** Level-2 and coarser grids surround level-1 grids and fill in the domain to the outer boundary. Examples of Cartesian off-body grids are shown in Fig. 4.2.

The basic controls for off-body grid generation in NAMELIST **&GBRICK** are **DS**, **DFAR**, **CHRLEN**, **OFRINGE**, **XNCEN**, **YNCEN**, and **ZNCEN**. The input **DS** is the grid 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** is the distance to (all) outer boundaries. **CHRLEN** is a characteristic body length (currently not used in the grid generation process). Generally **CHRLEN** is set to the major dimension of body; the default is 1.0. **XNCEN**, **YNCEN**, and **ZNCEN** define the center of the off-body grid system. The default is the center of the near-body grid system. The grid center inputs must be specified for moving body problems. **OFRINGE** specifies the number of fringe points to use with the off-body grids. The default **OFRINGE** is

based on the numerical scheme used for the off-body grids, double-fringe (**OFRINGE**=2) for 2nd- and 3rd-order algorithms, triple-fringe for 4th- and 5th-order schemes.

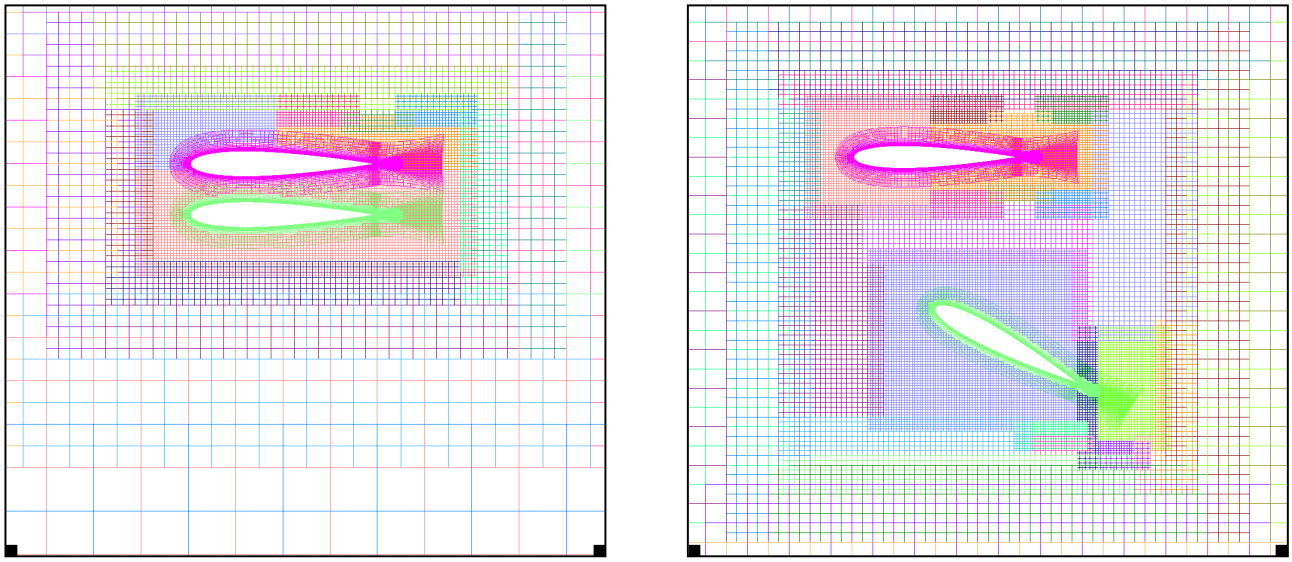


Figure 4.2 Cartesian off-body grid examples.

The **MINBUF** input in NAMELIST **&GBRICK** is used to control the rate of coarsening of the off-body grids as they move away from the level-1 grids. **MINBUF** is the minimum number of points at each grid level before switching to the next-coarser grid level. Larger values of **MINBUF** cause a more gradual coarsening of the off-body grids at the expense of adding additional points. Fig. 4.3 shows the grid system for **MINBUF**=4 (the default) and **MINBUF**=8. **MINBUF**=4 results in a 3D grid of 2 million points, while **MINBUF**=8 results in a 3D grid of 3 million points.

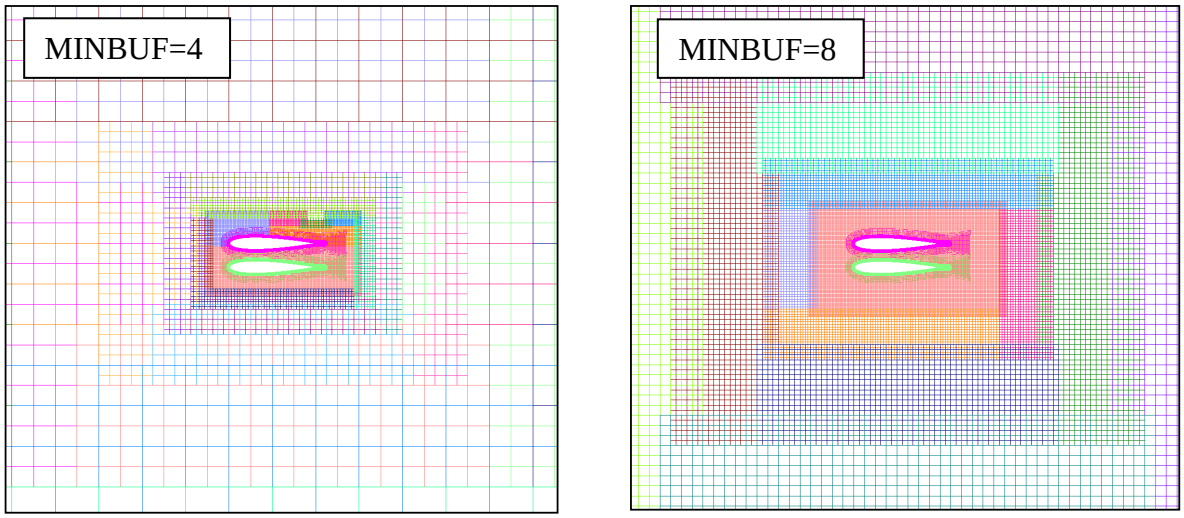


Figure 4.3 Effect of **MINBUF**=4 vs. 8 on off-body grids.

The NAMELIST **&GBRICK** inputs **I_XMIN**, **I_XMAX**, **I_YMIN**, **I_YMAX**, **I_ZMIN**, and **I_ZMAX** can be used to specify *x*-, *y*-, or *z*-constant planes for special cases such as ground planes or symmetry planes. The default value for each of these parameters is 0 and causes the far-field planes to be placed a distance **DFAR** from the grid

center. If one of these parameters is set to 1, then a plane will be placed at the location specified using the NAMELIST inputs **P_XMIN**, **P_XMAX**, **P_YMIN**, **P_YMAX**, **P_ZMIN**, and **P_ZMAX**. Only one of each *x*, *y*, *z* pair may be specified for a given problem (you cannot create a "channel" using **P_XMIN** and **P_XMAX** at the same time**)**. Two examples for these inputs are given in Fig. 4.4 and Fig. 4.5. In Fig. 4.4 a supersonic inlet plane is specified at *x*=-20. In Fig. 4.5 a symmetry plane is specified at *y*=0.

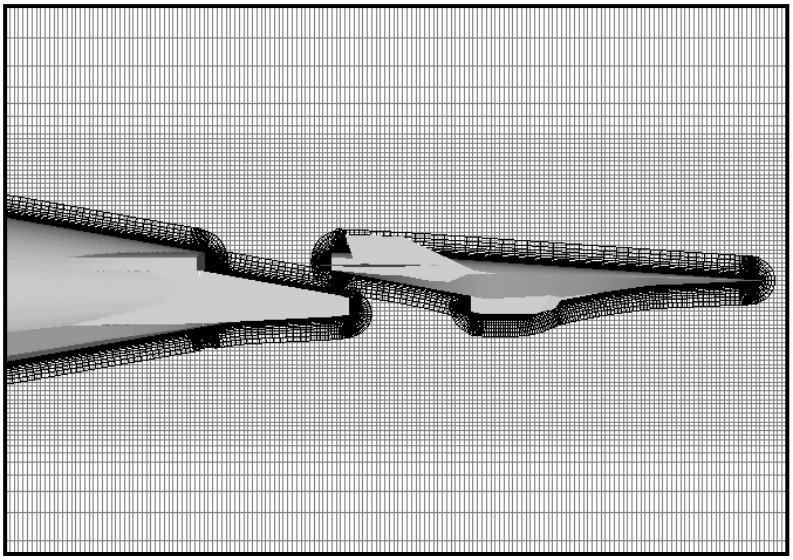


Figure 4.4 Hyper-X supersonic inflow plane.

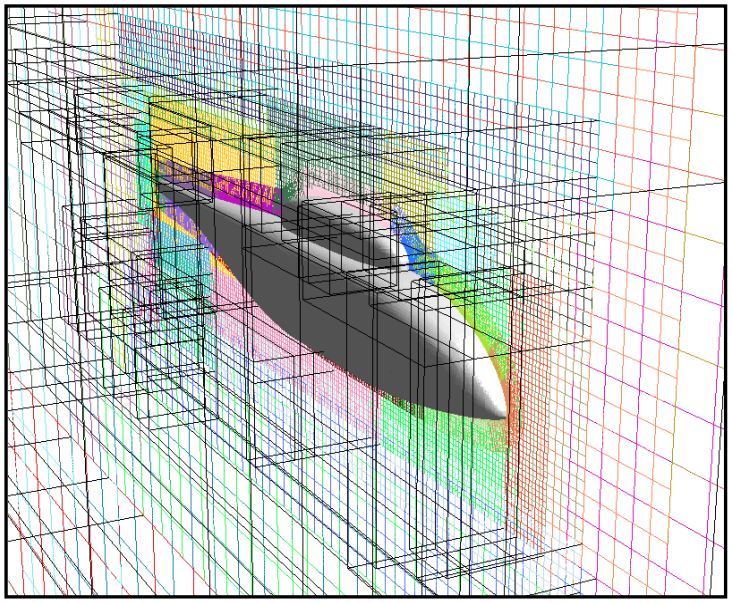


Figure 4.5 Helicopter symmetry plane.

Far-field boundary conditions for the off-body grids may be specified using the **&OMIGLIB** NAMELIST inputs **IBXMIN**, **IBXMAX**, **IBYMIN**, **IBYMAX**, **IBZMIN**, and **IBZMAX**. The boundary condition choices are

- Inflow/outflow conditions: BC types 30,35,37,40,41,47,49
- 2D or axisymmetric condition (y only): BC types 21,22
- Axis condition (z only, and combined with axisymmetric in y): BC type 16
- Symmetry conditions: BC types 11,12,13,17
- Inviscid wall: BC type 1

The default boundary condition is the Riemann characteristic inflow/outflow BC type 47. The following is the input required for the helicopter symmetry plane shown in Fig. 4.5.

Example Input 4.1

```
&OMIGLB   IBYMIN=17,
/
&GBRICK
      DS = 1, DFAR = 1000, CHRLEN = 100,
      XNCEN = 50, YNCEN = 0, ZNCEN = 0,
      I_YMIN = 1, P_YMIN = 0,
/
&BRKINP /
```

Typically "proximity regions," which must be covered by level-1 off-body grids, are determined from the boundaries of near-body grids which represent geometry. Additional regions can be specified using NAMELIST **&BRKINP**. This is often useful to refine the grid system to capture wakes or other flow features. This is often a more economical method of locally improving the grid resolution in terms of the number of grid points over use of the **&GBRICK MINBUF** input described earlier. The number of regions specified is given by **NBRICK**. If **NBRICK** is positive the new regions are added to the proximity regions derived from the near-body grids; if **NBRICK** is negative, the specified regions replace the existing regions. **XBRKMIN**, **XBRKMAX**, **YBRKMIN**, **YBRKMAX**, **ZBRKMIN**, and **ZBRKMAX** are used to define the extent of each of the proximity regions. **IBDYTAG** is used to associate the region with a body so that the region will move with the body during the simulation. **IBDYTAG**=0 causes the proximity region to remain fixed during the simulation. Fig. 4.6 shows a proximity region used in the wake region of a body. The proximity region causes the level-2 grids to be moved farther from the body in the wake region to provide better resolution of the wake. In this example it would be desirable for the proximity region to move with the body.

Again, note that these user-specified "bricks" are only regions to be covered by level-1 grids, not specifically level-1 grids themselves. Thus the actual resulting level-1 grids often extend somewhat beyond the specified regions.

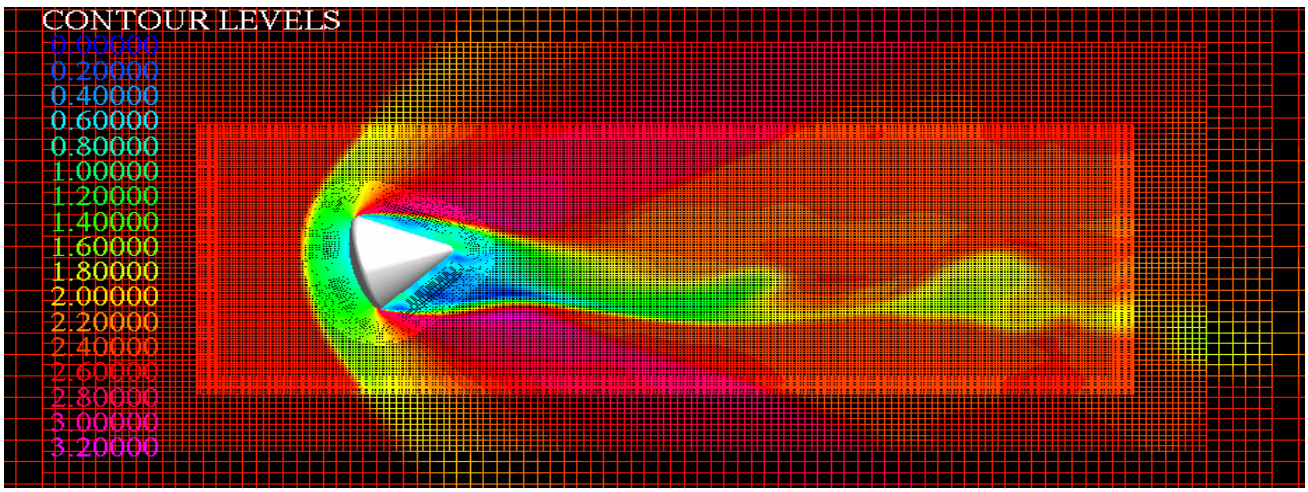


Figure 4.6. Proximity region used to calculate the wake from a body.

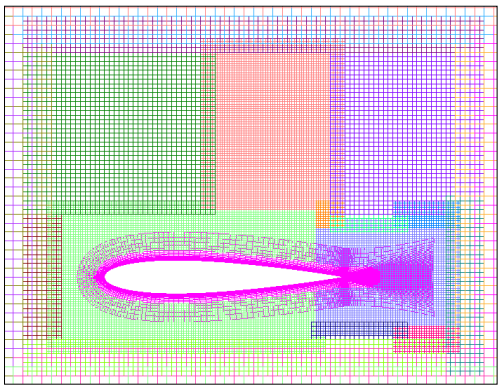
The following are two example inputs for user defined proximity regions. The first example uses a proximity grid to increase the resolution in the vicinity of a shock wave on the upper surface of the airfoil. The second example uses a proximity grid to move the level-2 grids farther away from the airfoil so that the off-body grids will not have to

be regenerated during the moving body simulation. In the first example the proximity region moves with the airfoil since it is used to capture the shock. In the second example the proximity region remains fixed since it is used to move the level-2 grid farther away from the airfoil and avoid off-body grid regeneration when the airfoil moves. The second example shows the grid system for the pitching airfoil with and without the proximity region specified.

Example Input 4.2

- 2D airfoil, chord=1, far-field at 100 chords
  - Use **$BRKINP** to add a refined level-1 region for shock
  - Since **IBDYTAG=1**, this region is tied to motion of the airfoil

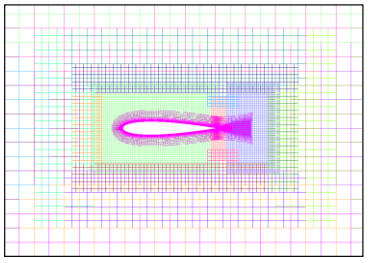
```
$OMIGLB  IBYMIN=21, …  $END
$GBRICK
    DS=0.01, DFAR=100, CHRLEN=1,
    XNCEN=0.5, YNCEN=0, ZNCEN=0,
    $END
$BRKINP
    NBRICK=1,
    XBRKMIN=0.5, XBRKMAX=0.9,
    YBRKMIN=0, YBRKMAX=0,
    ZBRKMIN=0, ZBRKMAX=1,
    IBDYTAG=1,
    $END
```



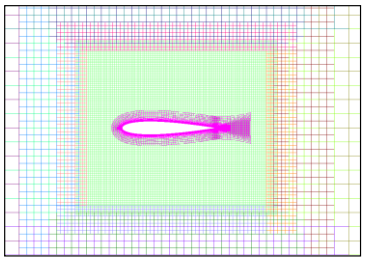
Example Input 4.3

- Airfoil forced oscillation problem
  - Use **$BRKINP** to make level-1 region big enough to capture expected body motion, so that off-body grids will not need to be regenerated during moving-body run

```
$OMIGLB  IBYMIN=21, …  $END
$GBRICK
    DS=0.01, DFAR=100, CHRLEN=1,
    XNCEN=0.5, YNCEN=0, ZNCEN=0,
    $END
$BRKINP
    NBRICK= -1,
    XBRKMIN= -0.3, XBRKMAX= 1.5,
    YBRKMIN= 0, YBRKMAX= 0,
    ZBRKMIN= -0.8, ZBRKMAX= 0.8,
    IBDYTAG= 0,
    $END
```



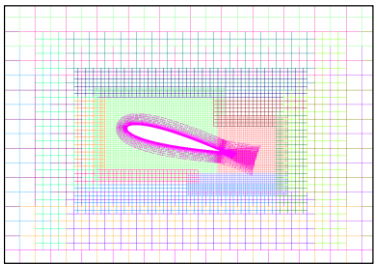
Without **$BRKINP**



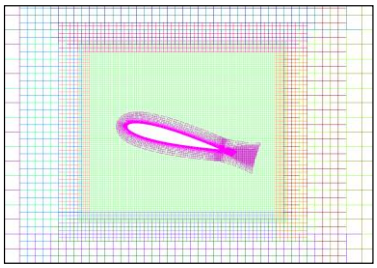
With **$BRKINP**

- Airfoil forced oscillation problem
  - Use **$BRKINP** to make level-1 region big enough to capture expected body motion, so that off-body grids will not need to be regenerated during moving-body run

```
$OMIGLB  IBYMIN=21, …  $END
$GBRICK
    DS=0.01, DFAR=100, CHRLEN=1,
    XNCEN=0.5, YNCEN=0, ZNCEN=0,
    $END
$BRKINP
    NBRICK= -1,
    XBRKMIN= -0.3, XBRKMAX= 1.5,
    YBRKMIN= 0, YBRKMAX= 0,
    ZBRKMIN= -0.8, ZBRKMAX= 0.8,
    IBDYTAG= 0,
    $END
```



Without **$BRKINP**



With **$BRKINP**

4.3 X-Ray Specification

Hole-cutting in OVERFLOW 2.3 is accomplished using Robert Meakin's X-ray method[3]. An X-ray is an (*x*,*y*) array of *z*-value pierce-points of a closed surface. A pre-processing step is required to generate the **xrays.in** file used by OVERFLOW during the DCF process. The Chimera Grid Tools (CGT) graphical interface **overgrid** should be used to generate the **xrays.in** file. Alternatively, the file can be generated using the stand-alone program **gen_x** (also part of CGT). The format of the **xrays.in** file is given in Appendix A. The terminology used for generating an **xrays.in** file include:

surface	A closed, watertight surface grid geometry must be specified to be X-rayed. The surface normals must all point in a consistent direction (all in or all out).
spacing	Spacing is used to define the refinement used by the X-ray projection scheme to define the hole.
bounding box	The X-ray bounding box should encompass the surfaces to be X-rayed. The bounding box is automatically determined in **overgrid**, but the ranges can be edited on-screen.
Body ID	X-rays are associated with a given body by specifying a Body ID (or Component ID). This is critical for moving body grid motion since the X-ray will move with the body. For body information specified using GMP (i.e., **I6DOF**=2), the Body ID is the numerical order of the components defined in the **Config.xml** file. For body information specified using **&SIXINP** (**I6DOF**=0 or 1), the Body ID is the **IBLINK** specified for each grid in NAMELIST **&SIXINP**.

The spacing defines the resolution of the body surface for the hole-cutting operation. For single-body applications the spacing size should be ½ to 1 times the outer cell size of the near body grid. For bodies in close proximity (such as weapons on a pylon) need to use a spacing of 0.1 to 0.2 times the minimum distance between the bodies. X-rays used for collision detection need to have higher resolution. The smaller the spacing the tighter the resolution, but the increased resolution comes at the expense of increasing the X-ray size and slowing the hole-cutting process in DCF. Hence it is best to use the maximum spacing possible for moving body applications.

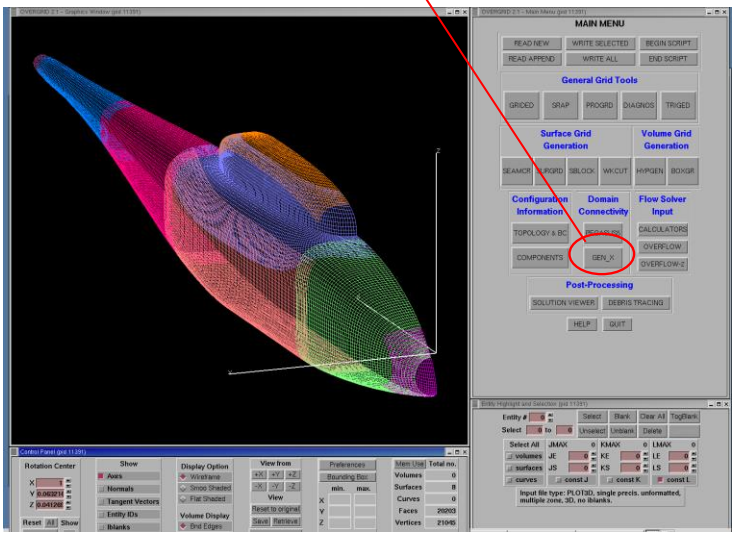
Note that different X-rays can be used for different reasons. For example, a coarser-spaced X-ray can be used for hole-cutting, while a finer-spaced X-ray is used for collision detection. Another example is the use of fine X-rays in the region of close proximity between bodies where precise geometry representation is important, and coarser X-rays elsewhere.

Before beginning the X-ray generation process, surface grids should be extracted from the volume grid. This can easily be done using **overgrid**. The surface grid should be a closed, watertight surface grid with all of the surface normals pointed in the same direction. The surface grid may contain overlapping sub-surfaces. It is easiest to deal with one X-ray at a time in **overgrid**. The individual X-rays can be concatenated using **overgrid**, or using the **xrayed** utility

in CGT. The X-rays may also be visualized in **overgrid**. The following steps are used to generate X-rays using **overgrid**.

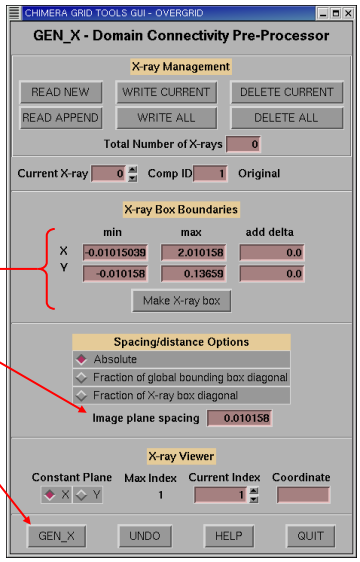
Step 1:

Start OVERGRID with the surface grid file
Click “GEN_X” under “Domain Connectivity”



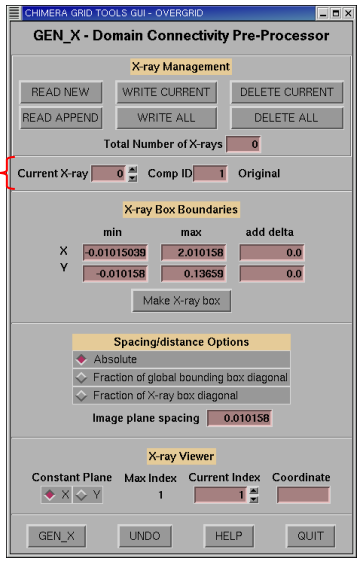
Step 2:

- Adjust box boundaries to be multiples of X-ray spacing
  - Ignore “add delta”
- Click “Make X-ray box”
- Enter X-ray spacing as “Image plane spacing”
- Click “GEN_X” to generate the X-rays
- Click “WRITE CURRENT” or “WRITE ALL” to save the X-rays to a file



Step 3:

- 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



X-rays may also be generated using the **gen_x** utility:

- **gen_x** is a text-input utility in Chimera Grid Tools (CGT)
- Documentation is included with CGT (excerpted here)
  - Input files:
    - PLOT3D surface grid file
    - Input parameter file:

```
surface_grid_filename
1                ISOPT(1/2/3)
0.01             DS
0                DELTA
1                NCROPS
1                IDBODY
-0.1, 1.1, 0, 1.5, -1, 1          XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX
```

(Ignore DELTA parameter)

  - Output files:
    - X-ray file **gen_x.xry**
    - Output messages
  - Execution:
    - gen_x < input_parameter_file > output_messages_file
- X-rays can be read into OVERGRID for viewing

Each **&XRINFO** NAMELIST is referred to as an X-ray cutter. The NAMELIST controls how an X-ray is used to cut holes, or for collision detection. Separate **&XRINFO** NAMELISTs are supplied for different X-rays, but the same X-ray may also be used in multiple cutters. The **IDXRAY** input identifies which X-ray in the **xrays.in** file to use for cutting a hole. The grids to be cut are specified using **IGXLIST** or the **IGXBEG**, **IGXEND** NAMELIST inputs. The off-body Cartesian grids are identified by the special grid number “-1”. **XDELTA** is the offset of the hole from the body surface defined by the X-ray, and can be used to expand the hole to move the interpolation points away from the body surface. As an example,

```
&XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.05, /
```

causes DCF to use the first X-ray in the **xrays.in** file to cut a hole in all of the off-body grids 0.05 units off of the X-ray surface, as shown in Fig. 4.7.

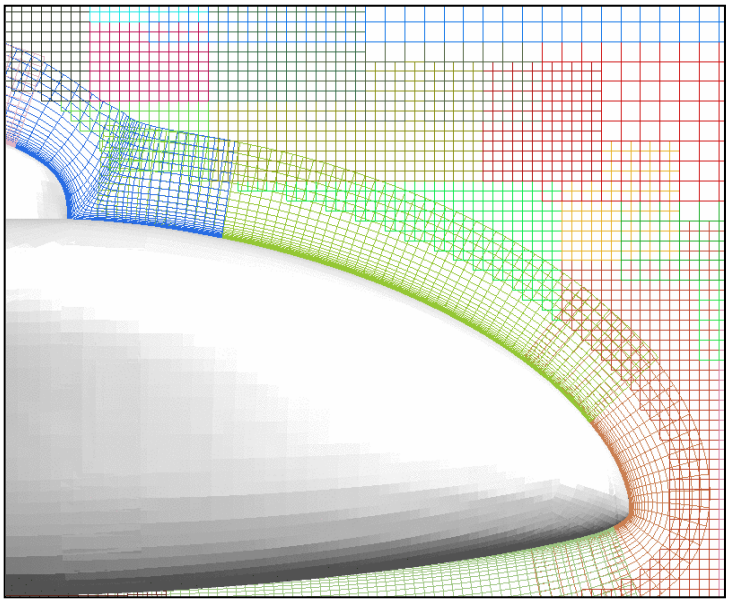


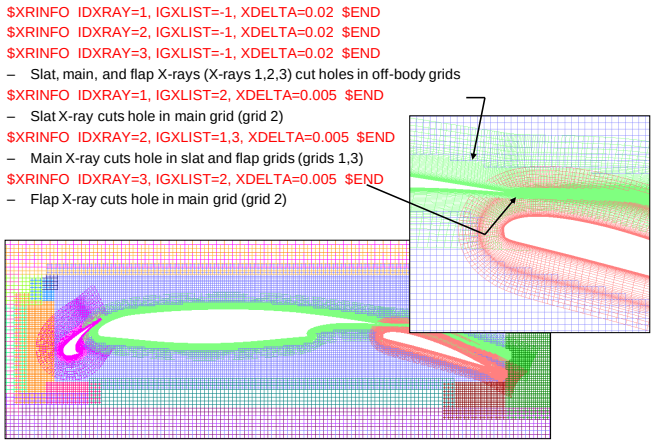
Figure 4.7 Hole cut in off-body grids.

The same X-ray may be reused to cut a different size hole in a different mesh as shown below.

```
&XRINFO IDXRAY=1, IGXLIST=-1, XDELTA=0.05, /
&XRINFO IDXRAY=1, IGXLIST=1,2, XDELTA=0.5, /
```

This input will cause DCF to use the first X-ray in the **xrays.in** file to cut a hole in all of the off-body grids 0.05 units off of the X-ray surface. The same X-ray will then be used to cut a larger hole (0.5 grid units off the surface) in grids 1 and 2. The following example shows the X-ray specifications for a multi-element airfoil case.

Example Input 4.3



Holes should be cut to keep coarser grids out of high-gradient regions (such as boundary layers). Holes should also 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*–5***DS**, where *S* is the distance from the surface to the outer boundary of the near-body grid. This is often about half the distance *S*. **XDELTA** must be less than half the expected minimum distance between bodies to avoid orphan points when cutting holes in nearby bodies. Different values of **XDELTA** may be used for different cutters.

4.4 Grid Assembly Using DCF

The DCF grid assembly code in OVERFLOW 2.3 can be run once the **grid.in** and **xrays.in** input files are completed and boundary conditions have been applied to the grids. All near-body grid boundaries that do not have a boundary condition specified are assumed to be interpolated boundaries. The input **IRUN** in NAMELIST **&OMIGLB** controls the execution of the grid assembly process. If **IRUN**=1 the code will stop after the off-body grids are generated and write the **brkset.restart** file and **x.save** file (with no IBLANK). IF **IRUN**=2 the code will stop after the grid assembly is completed and write **brkset.restart**, **INTOUT** and **x.save** files. The **x.save** file is a PLOT3D-format grid file that includes the IBLANK information and the off-body grid system. The **x.save** file is only used for post-processing. If **IRUN**=0 (default) the code will stop after the flow solver is finished. If the **brkset.restart** file is not present, the code will generate the off-body grids. If either **brkset.restart** or **INTOUT** is not present the code will perform the grid assembly. The **x.save** file is not written when **IRUN**=0. If the **brkset.restart** and **INTOUT** files are present the off-body grid generation and the grid assembly will not be performed and the files will be used to initialize the run.

Note: this means that the **brkset.restart** and **INTOUT** files must be manually deleted if a new set of Cartesian off-body grids or a new interpolation file is desired.

It is desirable that the interpolation boundaries have enough layers to maintain the full stencil of the flux algorithm used for the solution. Two layers of fringe points are required for 2nd- or 3rd-order spatial algorithms. The 4th- and 5th-order algorithms need three layers of fringe points. The algorithms will reduce to 1st-order on the interpolated boundaries if only a single fringe layer is present. The **LFRINGE** parameter in NAMELIST **&OMIGLB** is used to set the number of fringe points for near-body grids and hole boundaries. The default is based on the numerical scheme used by near-body grids. Any fringe points beyond the first that are orphans will revert to field points, unless **LFRINGE** is set to a negative number. The **OFRINGE** parameter in NAMELIST **&GBRICK** is used to specify the number of fringe layers for the off-body grids. The default is based on the off-body numerical scheme. Note that if the numerical schemes change on a restart, **LFRINGE** can be changed, but **OFRINGE** cannot, since this would imply changing the off-body grid system.

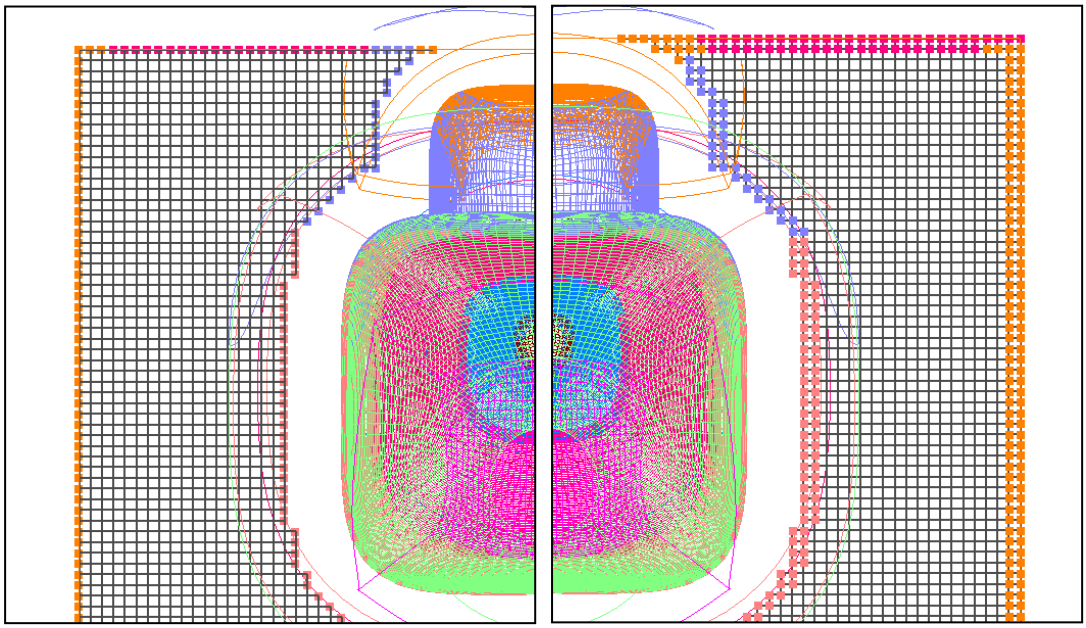


Figure 4.8 Example of single- and double-fringe layers.

The NAMELIST **&DCFGLB** input variable **DQUAL** specifies the acceptable level of interpolation stencil quality during the grid assembly process. **DQUAL** varies between 0 and 1. For **DQUAL**=1, donor stencils must consist of only field points (this is the default). If **DQUAL**=0, stencils which include all fringe points may be accepted.

This is *not* a good idea since the simulation may simply pass boundary data back and forth between grids. **DQUAL**=0.1 is generally acceptable.

Viscous stencil repair is needed to handle inaccurate interpolation stencils that can be created when overlapping surface grids lie on the same curved surface. If not corrected, these can result in orphan points (common near convex surfaces) or interpolations too high in the boundary layer (common near concave surfaces). The input **MORFAN**=1 in NAMELIST &**DCFGLB** turns viscous stencil repair on. The input parameter **NORFAN** specifies the number of points above a viscous wall subject to viscous stencil repair. The 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. The quality of repaired stencils is not checked.

The code standard output includes a summary of the orphan points generated in the grid assembly process. The orphan points may be visualized by importing the **x.save** file into **overgrid** and using the diagnostic functions. An example of the standard output summary is shown in Fig. 4.9.

```
..............  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  ..............
```

Figure 4.9 Example of OVERFLOW 2.3 standard output orphan summary.

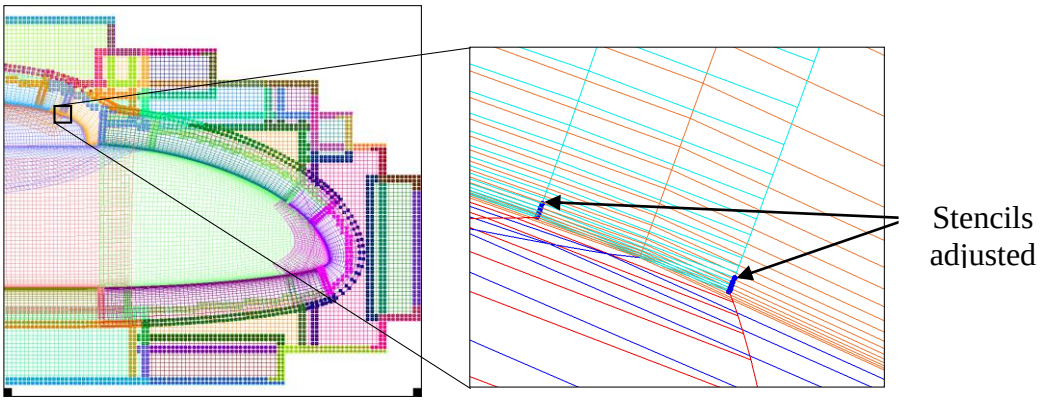
The number of orphans should be minimized during the grid assembly process. This may require refining near-body and off-body grids to improve the overlap. It is acceptable to have some orphan points, but the user needs to understand why the orphan points occurred and where they are in the grid system. Orphan points can be much harder to control during moving body problems. OVERFLOW 2.3 "fills" orphan and hole points with the average of the neighboring point values. This reduces the error caused by orphan points and allows hole points that are uncovered during moving body simulations to enter the domain with reasonable starting values.

The following are two examples of grid assembly problems. The first, Example Input 4.4, shows boundary points whose interpolation stencils are adjusted due to the viscous interpolation problem. The second example, Example Input 4.5, shows the need to use small values of **XDELTA** when bodies are in close proximity.

Example Input 4.4

- Helicopter fuselage

```
$OMIGLB  LFRINGE=2, …  $END
$DCFGLB  DQUAL=0.3, MORFAN=1, NORFAN=6,  $END
$XRINFO  IDXRAY=1, IGXLIST=-1, XDELTA=0.035,  $END
```

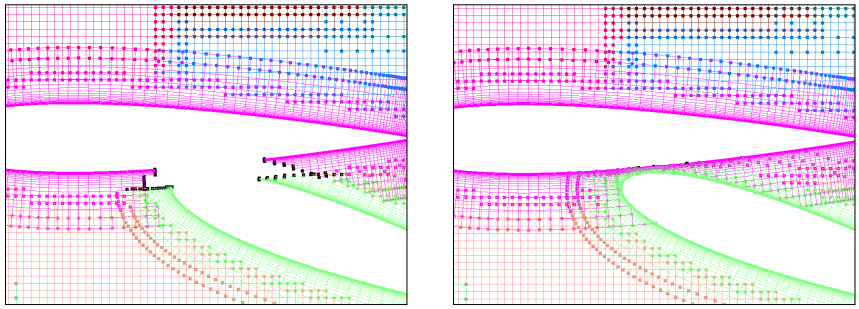


Example Input 4.5

- 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
```



4.5 Adaptation to Solution Error

OVERFLOW 2.3 can adapt both the near- and off-body grid system to solution error and geometry. The original adaptation process is described in Refs. 4 and 5. The solution adaption capability was significantly improved in OVERFLOW 2.2, while keeping much of the same off-body grid generation mechanics. The main input parameters that control grid adaption are in NAMELIST **&OMIGLB**. Off-body grids that result from solution adaption may be finer than level-1. These are labeled level-(-1), (-2), etc., each finer than level-1 by an additional factor of 2. The number of refinement levels allowed is controlled by **NREFINE** and **NBREFINE**. See Refs. 6-7 for a more complete description of the adaption process and input controls.

The frequency of grid adaptation is determined by the NAMELIST input **NADAPT** in **&OMIGLB**. If **NADAPT**=0 no adaptation occurs. If **NADAPT** is set to a positive integer *n*, the grid is adapted every *n* time steps. Adaptation often results in more grid points and may require more groups to fit the problem in memory. Adaptation also requires that the new off-body grid system interpolate starting values from the solution on the old grid system. The input **SIGERR** specifies the order of error for adaptation. The error is estimated at each point in the off-body grid

system by comparing the square of the difference between the local value and the average of the adjacent values. This error is compared to a tolerance defined as

$$\text{Tolerance} = QREF_N * DS^{SIGERR} \quad (4.1)$$

where DS is the grid spacing for the off-body mesh, and $QREF_N$ is the reference value for the conserved variable. If the local error is greater than the tolerance, the off-body grid is marked for refinement. **SIGERR** is normally set to 2.0.

It should be noted that this adaption capability is limited in several ways. First, no off-body grid will have a finer spacing than the level-1 grids. Second, the grid refinement only adds or subtracts regions of level-1 grids. It does *not*, for example, cause any level-3 grid regions to become level-2, or vice-versa.

References

1. Chan, W.M., Gomez III, R.J., Rogers, S.E., and Buning, P.G., "Best Practices in Overset Grid Generation," AIAA-2002-3191, June 2002.
2. Meakin, R.L., "Automatic Off-Body Grid Generation for Domains of Arbitrary Size," AIAA-2001-2536, June 2001.
3. Meakin, R.L., "Object X-Rays for Cutting Holes in Composite Overset Structured Meshes," AIAA-2001-2537, June 2001.
4. Meakin, R.L., "An Efficient Means of Adaptive Refinement Within Systems of Overset Grids," AIAA-95-1722, June 1995.
5. Meakin, R.L., "On Adaptive Refinement and Overset Structured Grids," AIAA-97-1858, June 1997.
6. Buning, P.G., and Pulliam, T.H., "Cartesian Off-Body Grid Adaption for Viscous Time-Accurate Flow Simulations," AIAA-2011-3693, June 2011.
7. Buning, P.G., and Pulliam, T.H., "Near-Body Grid Adaption for Overset Grids," AIAA-2016-3326, June 2016.