OVERSET MESH RUNS with NSU3D

NSU3D supports the use of multiple overset meshes, when linked to the TIOGA overset mesh library. In this mode, NSU3D supports overset meshes either in steady-state or time-dependent modes, and either using static meshes (no relative motion) or dynamically moving overset meshes. For steady-state problems, the overset mesh IBLANK and fringe interpolation patterns are computed once at startup, whereas for time-dependent problems with dynamically moving meshes these are recomputed at each time step.

Enabling Overset Mesh runs in the NSU3D input file:

Overset mesh runs are enabled by specifying a mesh file name in the NSU3D input file that ends with the suffix **filename.replicate** as shown in the example below:

```
------------------------------------------------------------------------
MACH      Z-ANGLE   Y-ANGLE   RE       RE_LENGTH
0.5        0.00       0.0     3000000.    1.0
------------------------------------------------------------------------
FORCE/MOMENT COEFFICIENT PARAMETERS
REF_AREA  REF_LENGTH XMOMENT  YMOMENT   ZMOMENT  ISPAN    REF_MACH
72700.     141.2     154.245  0.0         0.0     2.0      -1.0
------------------------------------------------------------------------
MESH DATA FILE
mesh_file.replicate
------------------------------------------------------------------------
```

Instead of specifying a partitioned grid file, the **mesh_file.replicate** file refers to a Fortran namelist file that specifies the individual (partitioned) component meshes of the overset mesh system along with parameters for positioning these meshes into the current initial position for the problem setup. When read in, each component overset mesh is placed on its own distinct set of processors (i.e. a given processor only holds one partition of a single overset component mesh). Therefore, it is up to the user to pre-partition the individual overset meshes using the appropriate number of partitions based on the mesh size to ensure that the overall simulation is well load balanced when running in parallel. Furthermore, when running an overset mesh case, the total number of processors specified for mpirun must correspond to the sum of all partition numbers specified in the mesh_file.replicate file. Otherwise, NSU3D will stop and output the required number of processors for this run based on the information read in from the mesh_file.replicate file. Note that the mesh motion input file is also copied to the ./WRK directory at run time in order to produce an archival record of all inputs for the current run.

An example of a *mesh_file.replicate** input file for two grid system for an overset mesh problem consisting of a wing and aileron is given below:

```
!-----------------------------------------------------------
!Overset mesh input file (mesh_replicate)  for WINGCS case
!-----------------------------------------------------------
&number_of_meshes_def
  nreplicate        = 2
  order             = 1 !displace,scale
! order             = 2 !scale,displace
/
&mesh_replicate_def
  mesh_file_name    = '../grids/NACA0012_AR10_1.10Mn.merged.part.16'
  xscale            = 1.0
  xdisplace(1:3)    = 0.0,0.0,0.0
  rotation(1:4)     = 1.0,0.0,0.0,0.0  !Axis and magnitude in degrees
```

```
  rot_center(1:3)   = 0.0,0.0,0.0
  final_trans(1:3)  = 0.0,0.0,0.0
/
!------------------------------------------------
&mesh_replicate_def
  mesh_file_name    = '../grids/NACA0012_Aileron_0.48Mn.merged.part.8'
  xscale            = 1.0
  xdisplace(1:3)    = 0.0,0.0,0.0
  rotation(1:4)     = 0.0,1.0,0.0,10.  !Axis and magnitude in degrees
  rot_center(1:3)   = 0.0,0.0,0.0
  final_trans(1:3)  = 0.715,2.5,0.0
/
!------------------------------------------------
```

The **mesh_file.replicate** file contains two types of sections delineated by the key words **number_of_meshes_def** and **mesh_replicate_def**.

- **number_of_meshes_def**: This section comes first and defines the number of component meshes and the order used for scale/displace operations as detailed below. Note that this section is optional as the reader will scan the **mesh_file.replicate** file to determine the nnumber of meshes if **nreplicate** is not specified. However, specifying **nreplicate** will override the default.
  - **nreplicate**: Number of overset component meshes to be read in (optional).
  - **order**: Determines the order in which scale/displace transformations are applied to each mesh.
    - **order=1**: Displace first, then scale (see next section)
    - **order=2**: Scale first, then displace (see next section)
- **mesh_replicate_def**: For each component mesh, this section defines the mesh file to be read in and parameters for positioning the mesh from its original coordinate system into the proper position in the final composite overset mesh coordinate system. The parameters for defining the mesh positioning are over-specified, in that two translations are possible, one before the rotation is applied, and one translation after rotation is applied. This is to enable more intuitive options for the user in positioning these component meshes. Additionally, the order of scale/displace (i.e. first translation) is also user configurable through the **order** parameter in the previous section.
  - **mesh_file_name**: Name of partitioned NSU3D mesh file to be read in for this mesh component
  - **xscale**: Scaling factor applied to original component mesh coordinates
  - **xdisplace**: Initial translation applied to component mesh coordinates
  - **rotation**: First 3 numbers define rotation axis, fourth number defines rotation magnitude in degrees applied to scaled/displaced component mesh coordinates
  - **rot_center**: Rotation center for above rotation
  - **final_trans**: Final translation to be applied to scaled/displaced(translated)/rotated mesh coordinates

Note that the same mesh may be replicated more than once in different positions, for example when simulating a rotor or propellor with multiple identical blades. The example below illustrates the setup for a 4-bladed propellor taken from the Workshop for Integrated Propulsion Prediction (WIPP) test case:

```
!---------------------------------------
!Overset mesh input file for WIPP case
!---------------------------------------
&number_of_meshes_def
```

```
  nreplicate         = 5
  order              = 1 !displace,scale
! order              = 2 !scale,displace
/
!---------------------------------------------------
&mesh_replicate_def
  mesh_file_name     = '../grids/blade.part.16'
  xscale             = 1.0
  xdisplace(1:3)     = 0.0,0.0,0.0
  rotation(1:4)      = 1.0,0.0,0.0,0.0  !Axis and magnitude in degrees
  rot_center(1:3)    = 0.0,0.0,0.0
  final_trans(1:3)   = 0.0,0.0,0.0
/
!---------------------------------------------------
&mesh_replicate_def
  mesh_file_name     = '../grids/blade.part.16'
  xscale             = 1.0
  xdisplace(1:3)     = 0.0,0.0,0.0
  rotation(1:4)      = 0.0,1.0,0.0,10.  !Axis and magnitude in degrees
  rot_center(1:3)    = 0.0,0.0,0.0
  final_trans(1:3)   = 0.715,2.5,0.0
/
!---------------------------------------------------
 &mesh_replicate_def
  mesh_file_name     = '../grids/blade.part.16'
  xscale             = 1.0
  xdisplace(1:3)     = 0.0,0.0,0.0
  rotation(1:4)      = 0.0,1.0,0.0,10.  !Axis and magnitude in degrees
  rot_center(1:3)    = 0.0,0.0,0.0
  final_trans(1:3)   = 0.715,2.5,0.0
/
!---------------------------------------------------
 &mesh_replicate_def
  mesh_file_name     = '../grids/blade.part.16'
  xscale             = 1.0
  xdisplace(1:3)     = 0.0,0.0,0.0
  rotation(1:4)      = 0.0,1.0,0.0,10.  !Axis and magnitude in degrees
  rot_center(1:3)    = 0.0,0.0,0.0
  final_trans(1:3)   = 0.715,2.5,0.0
/
!---------------------------------------------------
 &mesh_replicate_def
  mesh_file_name     = '../grids/wing.part.32'
  xscale             = 1.0
  xdisplace(1:3)     = 0.0,0.0,0.0
  rotation(1:4)      = 0.0,1.0,0.0,10.  !Axis and magnitude in degrees
  rot_center(1:3)    = 0.0,0.0,0.0
  final_trans(1:3)   = 0.715,2.5,0.0
/
!---------------------------------------------------
```

Specifying Dynamic Overset Mesh Motion:

For dynamic overset mesh cases, the mesh motion is specified in the same way as described for a single mesh in the **NSU3D TIME ACCURATE RUNS** documentation. This relies on the specification of a mesh motion input file in the NSU3D input file in the section for **OPTIONAL TIME ACCURATE** runs as shown below:

```
----------------------------------------------------------------------
OPTIONAL TIME ACCURATE PARAMETERS
NTIME_STEP  DTACC    DT_REF_LENGTH   NTIME_STEP_OUT   MESHMOTION
2880.       0.5        -1.0            30000.             1.0
MESH MOTION INPUT FILE
input2.meshmotion.nml
----------------------------------------------------------------------
```

For static mesh cases, the paremter **MESHMOTION** is set=0 and no mesh motion inpu file is required. If **MESHMOTION**=1**, the mesh motion input file name must be specified as shown above.

- **MESH MOTION INPUT FILE**: The name of the mesh motion file is specified here. The mesh motion file is a Fortran namelist file that specifies rotation and translation values for solid body motion of the mesh. An example of a mesh motion file is given below for the 4 bladed propellor WIPP test case.

```
!Overset mesh motion input file
!-------------------------------------------------
&mesh_motion_def
  mesh_id           = 1,2,3,4
  rotcenterxyz(1:3) = 0.0,0.0,0.0
  omegaxyz(1:3)     = 0.0,0.0,0.0456389419788
  gvtransxyz(1:3)   = 0.0,0.0,0.0
/
!-------------------------------------------------
&mesh_motion_def
  mesh_id           = 5
  rotcenterxyz(1:3) = 0.0,0.0,0.0
  omegaxyz(1:3)     = 0.0,0.0,0.0
  gvtransxyz(1:3)   = 0.0,0.0,0.0
/
!-------------------------------------------------
```

The parameters are defined as:

- **mesh_motion_def**: Key word in namelist identifying list of mesh motion parameters for one or more component meshes.
- **mesh_id**: Specifies one or more mesh ids to which this motion applied. The ids are based on the order in which the component meshes are specified in the **mesh_file.replicate** file. In this case, the four blade meshes move with the same motion and the wing mesh is fixed. Note that the last mesh (**mesh_id=5**) which represents the wing and bacground mesh is fixed and need not be specified, since no motion is the default for all component meshes.
- **rotcenterxyz(1:3)**: Center of rotation for mesh motion.
- **omegaxyz(1:3)**: Rotation rate vector components.
- **gvtransxyz(1:3)**: Grid velocity components for moving grid translation.

Determining the non-dimensional mesh grid velocities from dimensional or other non-dimensional quantities is described in the **NSU3D TIME DEPENDENT RUNS** documentation. Note that the mesh motion input file is also copied to the ./WRK directory at run time in order to produce an archival record of all inputs for the current run.

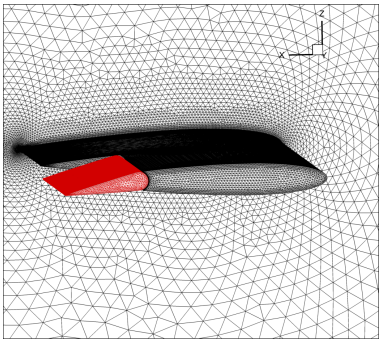
Sample Test Cases

Test Case 1: Wing Aileron Simulation (static grids, steady state)

This case consists of a wing with a control surface simulated using a global mesh for the main wing overset with a component mesh for the control surface. The overset mesh definition file (mesh_replicate) used for this case is given below:

```
!-------------------------------------------------------
!Overset mesh input file (mesh_replicate)  for WINGCS case
!-------------------------------------------------------
&number_of_meshes_def
  nreplicate        = 2
  order             = 1 !displace,scale
! order             = 2 !scale,displace
/
&mesh_replicate_def
  mesh_file_name    = '../grids/NACA0012_AR10_1.10Mn.merged.part.16'
  xscale            = 1.0
  xdisplace(1:3)    = 0.0,0.0,0.0
  rotation(1:4)     = 1.0,0.0,0.0,0.0  !Axis and magnitude in degrees
  rot_center(1:3)   = 0.0,0.0,0.0
  final_trans(1:3)  = 0.0,0.0,0.0
/
!-------------------------------------------------
&mesh_replicate_def
  mesh_file_name    = '../grids/NACA0012_Aileron_0.48Mn.merged.part.8'
  xscale            = 1.0
  xdisplace(1:3)    = 0.0,0.0,0.0
  rotation(1:4)     = 0.0,1.0,0.0,10.  !Axis and magnitude in degrees
  rot_center(1:3)   = 0.0,0.0,0.0
  final_trans(1:3)  = 0.715,2.5,0.0
/
!-------------------------------------------------
```

Note that the first mesh is not displaced from its original position. The second (control surface) mesh is rotated about the y-axis by the aileron deflection value in degrees (10 degrees downwards corresponding to positive rotation in y axis). The resulting mesh configuration is illustrated in Figure 1.1, including details in the gap region showing the tight tolerances that can be handled by the overset mesh approach.



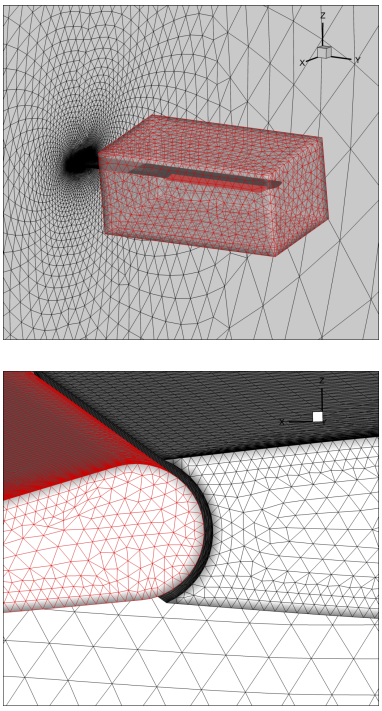


Figure 1.1: Illustration of wing/control-surface overset mesh configuration for 10 degree downward deflection of control surface and associated mesh.

In the second case, simply changing the control surface mesh rotation value from +10 to -10 degrees produces the following configuration with an upwards aileron deflection. In this manner, various different configurations can be analyzed quickly without the need for remeshing.

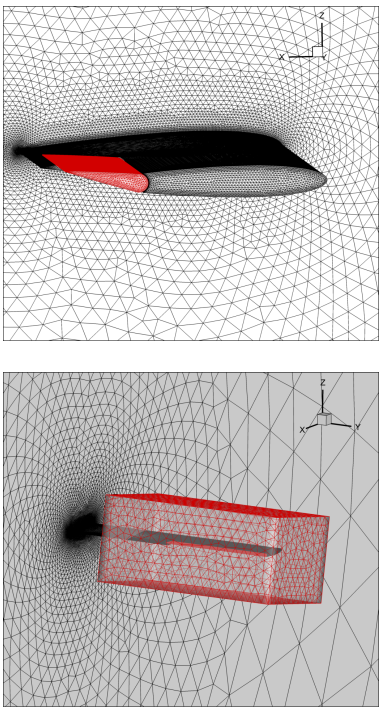


Figure 1.2: Illustration of wing/control-surface overset mesh configuration for 10 degree upward deflection of control surface and associated mesh.

Because this test case is a steady-state static overlapping mesh case, no mesh motion file is required. Using the in-situ postprocessing library, slices through the combined overset mesh system can be generated as shown below.

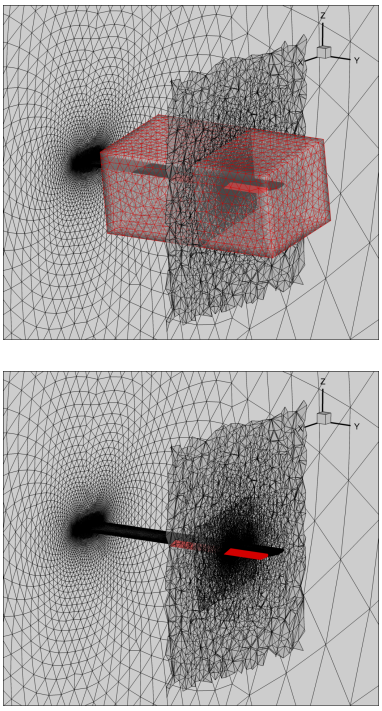


Figure 1.3: Illustration of plane slice through mesh overlap region.

By default, this slice cuts through all meshes in the overlapping regions. Using the **mesh_id** parameter, the user can specify cuts through individual meshes in the overlapping mesh system. Figure 1.4 illustrates the same slice applied only to the global wing mesh, and applied only to the control-surface component mesh, respectively.

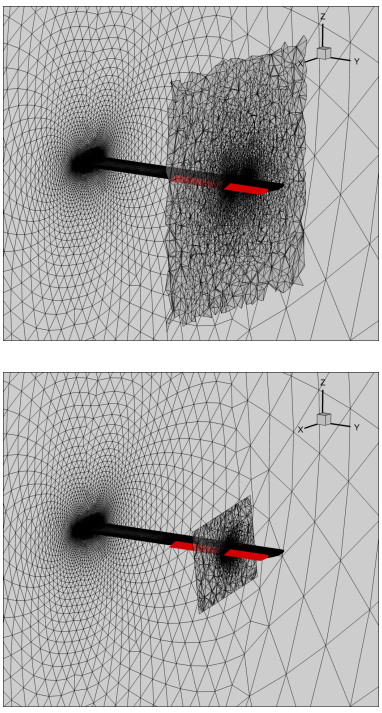


*Figure 1.4: Illustration of plane slice through mesh overlap region spearated out for each mesh individually using the **mesh_id** option.*

Isolating the slices through individual meshes is useful for visualizing the computed solution fields more clearly, including for debugging purposes. The in-situ postprocessor can visualize the overset mesh IBLANK values, which determine which mesh is dominant in overlap regions. By convention, IBLANK = 1 denotes dominant mesh points or control volumes (i.e. solution computed on this mesh), IBLANK= -1 denotes fringe

points or control volumes (solution not computed but interpolated from dominant mesh), and IBLANK = 0 denotes a hole, or mesh region obscured by a body (i.e. not active in simulation). Figure 1.5 illustrates the resulting IBLANK patterns on the slice plane through both meshes, showing how the overlapping meshes have different values as expected, making this difficult to visualize clearly. This can be improved by visualizing the IBLANK patterns on each mesh individually. Here, the second and third figures show the IBLANK patterns on the slice plane obtained using the mesh_id parameter for each indvidual component mesh, producing more informative plots.

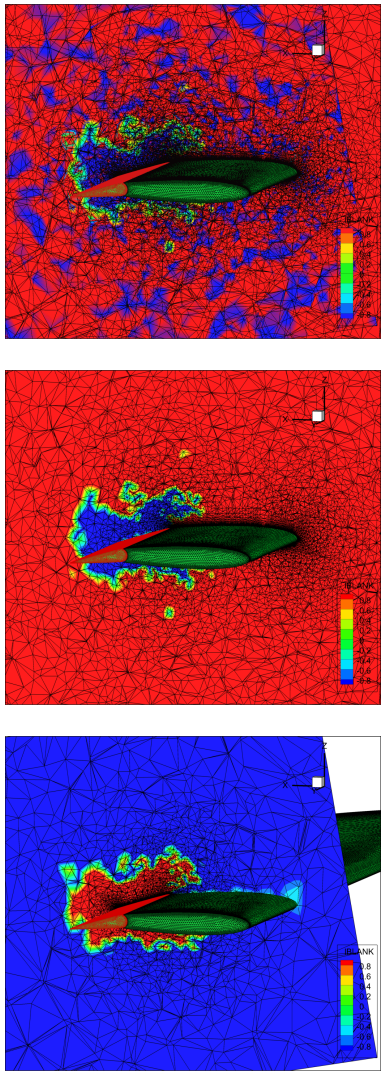


Figure 1.5: Illustration of rendering of the IBLANK pattern on the single slice plane that includes both overlapping meshes compared with IBLANK patterns rendered on each mesh slice individually.

Because the slice plane cuts through mesh cells and necessarily interpolates values between the cell vertices, a more informative IBLANK plot can be obtained using the crinkle cut feature in the postprocessing library. Figure 1.6 illustrates the IBLANK pattern on the crinkle cut planes which cut along cell faces, for each mesh independently, producing a more clear picture of the individual IBLANK regions on each mesh.

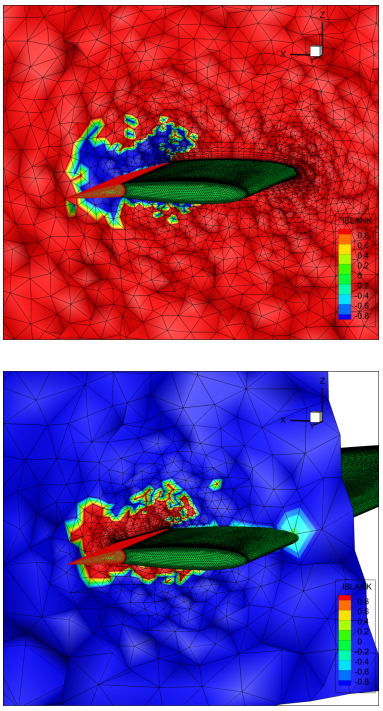


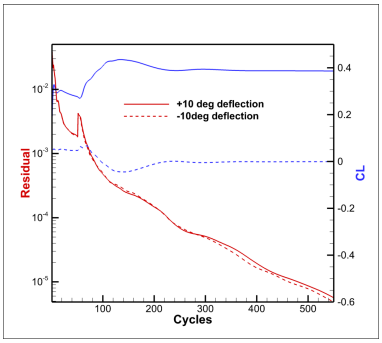
Figure 1.6: Illustration of IBLANK pattern rendered on individual component mesh crinkle cuts, which is the preferred approach for examining IBLANK patterns.

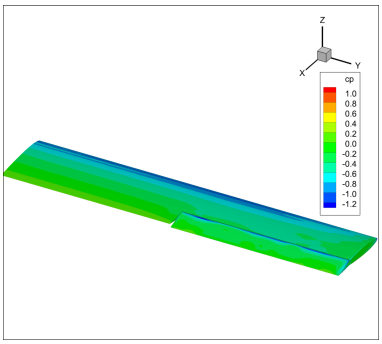
The aerodynamic flow over the wing/control-surface overset mesh configuration is computed for two deflection values: +10 deg and -10deg. The flow conditions are Mach=0.6, Angle of attach = 2.0deg. RE=3M. The NSU3D input file for these cases is reproduced below:

```
NSU3D INPUT FILE FOR WINGCS CASE
RESTARTF  RESTARTT  RNTCYC
0.0        0.0       00.
RESTART FILE
WRKab/restart.out
MMESH     NTHREAD
1.0       1.0              (1st order = -1)
NCYC      NPRNT     N MESH    MESHLEVEL   CFLMIN    RAMPCYC   TURBFREEZE
50.         0.0         1.        1.0        1.0       0.0       0.
500.        0.0         4.        1.0        1.0       0.0       0.
CFL       CFLV      ITACC     INVBC     ITWALL    TWALL
1.0        1000.      0.0       0.0       0.0      0.0
VIS 1     VIS 2     H FACTOR  SMOOP     NCYCSM
0.0        20.        0.00     0.00      0.0
C1        C2        C3        C4        C5        C
0.5321    1.3711    2.7744
FIL1      FIL2      FIL3      FIL4      FIL5      FIL6
1.0       1.0       1.0
-------------------------------------------------------------------------
COARSE LEVEL AND MULTIGRID PARAMETERS
CFLC      CFLVC     SMOOPC    NSMOOC
1.0       1000.     0.0       0.0
VIS0      MGCYC     SMOOMG    NSMOOMG
4.0       2.        0.8       2.0
-------------------------------------------------------------------------
TURBULENCE EQUATION(S)
ITURB     IWALLF    WALLDIST
4.        0.0        1.0
```

```
CT1       CT2        CT3      CT4       CT5       CT6
1.0       1.0        1.0
CTC1      CTC2       CTC3     CTC4      CTC5      CTC6
1.0        1.0        1.0
VIST0  TSMOOMG   NTSMOOMG
2.0        0.8        2.0
-------------------------------------------------------------------------
MACH      Z-ANGLE   Y-ANGLE   RE        RE_LENGTH
0.6        0.00       2.0     3000000.     1.0
-------------------------------------------------------------------------
FORCE/MOMENT COEFFICIENT PARAMETERS
REF_AREA  REF_LENGTH XMOMENT  YMOMENT   ZMOMENT  ISPAN    REF_MACH
5.           1.0     0.25     0.0         0.0     2.0      -1.0
-------------------------------------------------------------------------
MESH DATA FILE
mesh_file.replicate
-------------------------------------------------------------------------
OPTIONAL PARAMETERS (MODIFY FROM DEFAULT VALUES SET IN set_lim_values.f)
PARAMETER NAME                          VALUE(real number with decimal)
IN_SITU_POST_PROC                       1.0
```

Figure 1.7 illustrates the convergence for both aileron deflection cases, as well as the computed surface pressure distributions. Note that the CL value for the downwards deflection case converges to CL=0.386, whereas for the upwards deflection case the final CL is slightly negative, close to 0.0. This test case also illustrates the ability of the overset mesh approach to handly close tolerances between geometry and component meshes, which must be of the order of the local cell size.





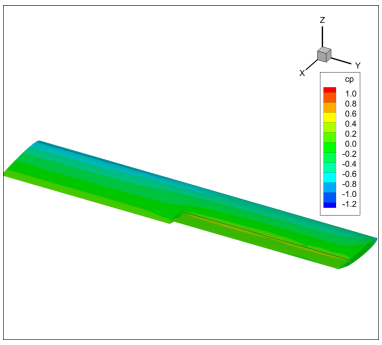


Figure 1.7: Convergence history for positive and negative 10 deg. control surface deflection cases and corresponding computed surface pressure distributions.

Test Case 2: WIPP Propeller Case (5 meshes, moving overset blade meshes)