In-Situ Post-processing with NSU3D

Starting with version r-400, NSU3D can be used with the customized in-situ post-processing "post_proc" library developed by Scientific Simulations LLC. To use this feature, NSU3D must be compiled with the post_proc feature enabled and linked to the post_proc library.

The post_proc library enables the extraction of surface data, planar slices, volume isosurfces, entire volume data and line/point probe values at run time in a parallel computing environment. Data extraction can be performed at the end of a simulation run, and/or at periodic intervals during the run, thus enabling the construction of dynamic simulation videos for time-dependent runs.

The post_proc library provides support for vtk output file formats as well as the tecplot *.dat *.plt and parallel *.szplt file formats, when linked to the tecio library from tecplot.

Enabling in-situ postprocessing:

- **Compilation and Installation**:

  To enable in-situ postprocessing, NSU3D must be compiled and linked to the post_proc library. First, to compile the post_proc library, enter the post_proc source code directory and type

  ```
  make
  ```

  There are no other library dependencies for the post_proc source code compilation. The result of this compilation will be the creation of the libppdata.a static library. To use the tecio library (enabling szplt files for tecplot) the setting: DTECLIB = TECLIB_STATIC should be used at compile time. If the tecio library is not available, the setting: DTECLIB = TECLIB_SHARED should be used. In this case, the tecio library will only be available as a shared library but will not be required at run time if tecio files are not specified.

  In the NSU3D source code makefile, use the following settings:

  ```
  POST_PROC=POST_PROC_ON
  POST_PROC_DIR=pathname to post_proc library source code
  POST_PROC_LIB=$(POST_PROC_DIR)/libppdata.a
  INCLUDEH += -I$(POST_PROC_DIR)
  INCLUDELIB += $(POST_PROC_LIB) -lstdc++ -lpthread -lm -ldl
  INCLUDELIBB += $(POST_PROC_LIB)
  ```

- **Optional Parameter Setting**:

  The optional parameter **IN_SITU_POST_PROC** is used to enable or disable the in-situ post-processing capability. This parameter must be listed at the end of the standard NSU3D input file, along with other optional parameters.

  IN_SITU_POST_PROC = 1 enables in-situ post-processing, provided NSU3D has been compiled and linked with the post_proc library. If this parameter is set = 1 and NSU3D has NOT been compiled with the post_proc library, NSU3D will terminate execution after reading in all input parameters with an error message. In this case, either setting **IN_SITU_POST_PROC = 0** or removing it from the optional parameter list (default) will enable execution of NSU3D without in-situ post processing.

  IN_SITU_POST_PROC = 0 disables in-situ post-processing. If this optional parameter is not specified, the default value is = 0. Note that in-situ postprocessing requires slightly more memory for steady-state NSU3D runs (approximately 20% depending on cases). Therefore setting IN_SITU_POST_PROC = 0 reduces the memory footprint of the run for cases where in-situ postprocessing is not required.

- **Input file for in-situ post processing**: *input.postproc*

The post_proc library reads an input file that specifies what types of data are to be extracted and the run-time frequency for data extraction. This file name is currently hardwired as:

input.postproc

This file should be created and placed in the run directory, in the same location as the nsu3d std input file.

The *input.postproc* file is read at startup and checked for consistency. After this, the file is read at each occurrence when a new data extraction operation is performed. In this manner, the input.postproc file can be modified on the fly as the simulation progresses. However, user errors are not checked after the initial startup, although errors will result in a data extraction being skipped rather than simulation failure.

If the *input.postproc* file is not found, either at the start of the run, or has been subsequently removed during the simulation, no data extraction will occur and the post_proc library will not be used.

Detailed description of *input.postproc* file:

A sample *input.postproc* file is given below. This is a standard Fortran namelist file. "!"symbols identify comment lines which are ignored by the reader. Key words are prefaced with the "&" symbol. These include the frequency parameter, and keywords identifying the type of plot or data extraction to be performed. Each plot type is terminated with a forward slash.

The default location for writing out all post_proc data files is in the subdirectory post_proc created by the solver at runtime in the ./WRK directory. Note that, if it exists, the ./WRK/post_proc directory is purged and re-created at startup of the flow solver. Output files can still be written to locations outside of this directory by specifying file names with pathnames relative to the ./WRK/post_proc directory (for example ../../my_output/slice.szplt)

The sample **input.post_proc** file below illustrates the various plot types. Following the entries in this example, the supported types are given as:

- **data_def**: This contains definitions for global default values. The data_def type contains the following default parameters:
    - **Frequency**: This parameter specifies the frequency of data extraction for the plot types enumerated below in the file. In this case, data extraction will be performed every 100 cycles for all plot types, by default. In addition, all plot type data extractions are performed at the end of the simulation. If the frequency parameter is not specified under this **data_def** section, data extraction occurs only at the end of the simulation run. The frequency parameter may be overridden for any plot types. For example, the statement : frequency = -1 can be placed within the plot type description as described below to force output only at the end of the run. For steady-state runs, the frequency parameter refers to the number of nonlinear cycles on the last mesh of the MMESH mesh sequencing inputs in the NSU3D input file. For time-dependent runs, the frequency parameter refers to the number of time steps in the simulation.
- **volume_def**: Defines a volume plot or flow field data throughout the entire flowfield. The volume_def type contains the following parameters:
    - **file_name**: Specifies the file name for the output file. Can include a relative or absolute pathname. As mentioned above, all post_proc data files are written to the subdirectory "post_proc" created by the solver at runtime in the ./WRK directory. The suffix .plt indicates this will be a tecplot .plt or szplt binary file. The file format can be specified also using the file_format key word described below. The available file formats are described in the next section. If no suffix is recognized, vtk format is assumed.
    - **file_format**: Specifies the file format. This can be used instead of specifying the file suffix in the file_name above. This will also over-ride any file format specification in the file_name specification. If no file format is recognized, vtk format is assumed.
    - **file_binary**: For VTK formats, setting **file_binary=0** produces an ASCII or readable text file. Default is binary file (**file_binary=1**). Has no effect on tecplot files which are always binary.

    - **varname**: These are the flowfield variables to be output. The variables are listed in any order separated by commas. Misspelt or non-existing variable names are ignored. A list of available types is given in the next section. Only the variables with is3Dvar = 1 can be output for volume plots.
    - **frequency**: Here the frequency parameter is set to -1, which results in the volume data only being output at the end of the run. Note this can be applied to any of the plot types. Also note that volume plots produce the largest files and should be used sparingly.
- **surface_def**: Defines a surface plot or flow field data on a boundary surface such as wall or aircraft surface. The surface_def type contains the following parameters:
    - **file_name**: Specifies the file name for the output file. Can include a relative or absolute pathname. As mentioned above, all post_proc data files are written to the subdirectory "post_proc" created by the solver at runtime in the ./WRK directory. Various examples are given in the sample file listing below. The file format can be specified also using the file_format key word described below. The available file formats are described in the next section. If no suffix is recognized, vtk format is assumed.
    - **file_format**: Specifies the file format. This can be used instead of specifying the file suffix in the file_name above. This will also over-ride any file format specification in the file_name specification. If no file format is recognized, vtk format is assumed.
    - **file_binary**: For VTK formats, setting **file_binary=0** produces an ASCII or readable text file. Default is binary file (**file_binary=1**). Has no effect on tecplot files which are always binary.
    - **outtype**: Geometry surface attribute used to define the surface for data extraction. These can be based on patches (patch), components (comp) or boundary conditions (bcs).
    - **bound_def**: The names of the various geometry boundary surface attributes are listed upon which the data is extracted according to the surface definition types in outtype above. Names that do not exist are ignored. A list of supported types is given below.
    - **varname**: These are the flowfield variables to be output. The variables are listed in any order separated by commas. Misspelt or non-existing variable names are ignored. A list of available types is given in the next section.
    - **frequency**: The frequency parameter can be used to override the default frequency set at the top of the input.postproc file.
- **plane_slice_def**: Defines data extracted by a planar cuts through the volume mesh or surface mesh. Any number of planar cuts can be defined and all cuts will be written to a single file. Plots of integrated forces over all planar cuts (i.e. for plotting wing span loads) can also be produced with the "varname" "force" (see below). The plane_slice_def type contains the following parameters:
    - **file_name**: Specifies the file name for the output file. Can include a relative or absolute pathname. As mentioned above, all post_proc data files are written to the subdirectory "post_proc" created by the solver at runtime in the ./WRK directory. The suffix .plt indicates this will be a tecplot .plt or szplt binary file. The file format can be specified also using the file_format key word described below. The available file formats are described in the next section. If no suffix is recognized, vtk format is assumed. If the variable name (i.e. "varname") type "force" is specified, integerated forces (CL, CD, CM etc.) will be computed for each surface slice and output to a file prefaced with the key work "force_filename".
    - **file_format**: Specifies the file format. This can be used instead of specifying the file suffix in the file_name above. This will also over-ride any file format specification in the file_name specification. If no file format is recognized, vtk format is assumed.
    - **file_binary**: For VTK formats, setting **file_binary=0** produces an ASCII or readable text file. Default is binary file (**file_binary=1**). Has no effect on tecplot files which are always binary.
    - **outtype**: Attribute used to define the region to be cut by the plane for data extraction. This can be a cut through the entire volume data of flow field (**volume** or **field**), or across the surface grid based on patches (patch), components (comp) or boundary conditions (bcs).

- **bound_def**: For cases where the planar cut is over the surface mesh only, the various geometry boundary surface attributes are listed upon which the data is extracted according to the surface definition types in outtype above. Names that do not exist are ignored. A list of supported types is given below. Note that for planar cuts of volume data, this variable is not used.
- **mesh_id**: For overset mesh cases only and only for plane_slice_def cases with **outtype** = volume or field. Selects one of more component meshes. Default value is "all"
- **varname**: These are the flowfield variables to be output. The variables are listed in any order separated by commas. Misspelt or non-existing variable names are ignored. A list of available types is given in the next section.
    - **force varname option**: In addition to the usual variable names, the name "force" can also be used to trigger a plot of integrated forces across all the slices in this "plane_slice_def". Note that the "force" varname automatically triggers Cp and Cf varname values to be output as well, since these are required to compute integrated forces. Two additional output files are called "force_wind.file_name" and "force_xyz.filename". This feature only applies to slices of surface entities and is ignored for slices of "**volume**" or "**field**" as specified in the "**outtype**" key word defined above. Two force files are output:
        - **force_wind.filename**: This is the force file given in wind axes. For each slice station, the file contains the following variables: "XC", "YC", "ZC", "CL", "CD", ""CLP", CLV", CDP", "CDV", "CM_ROLL", "CM_PITCH", "CM_YAW", "CM_2D", "CHORD", "REF_LENGTH". XC, YC and ZC correspond to the computed quarter chord point. All force coefficients are computed in grid units and normalized by REF_LENGTH which is reproduced from the flow solver input file. "CHORD" is the local chord as computed by the slice routine and normalized by "REF_LENGTH". In this manner, the local force coefficients based on the local chord can be obtained by multiplying by the "CHORD" value (for example in a TECPLOT layout or macro). All 3D moment coefficients are computed about the center of moment location prescribed in the flow solver input file. The exception is the 2D pitching moment coefficient: CM_2D. This is computed using the computed local chord length and the quarter chord point as specified in the XC, YC and ZC values for each slice. (Note that moment coefficients about different points can be computed from the information in this file.)
        - **force_xyz.filename**: This is an auxiliary force file that contains similar quantities based on body axes rather than wind axes.
- **origin and normal**: Each plane for a planar cut is defined by an origin and a normal vector in the x-y-z coordinate system of the grid. Any number of planar cuts can be defined for a single plane_slice_def entry. The origin and normal entries can be listed in any order, using the last index to identify the specific plane number.
- **bbox**: Bounding box. An optional bounding box (rectangle or circle) can be defined for each field slice beyond which the mesh and flow field data are cropped. The following options can be used:
    - *bbox(:)* = ' ', no bounding box (default value). Also, if *bbox* keyword is not present no bounding box is used.
    - *bbox(:)* = 'circle': Creates a circular bounding region. Requires *bbox1* as an input for the circle radius:
        - *bbox1(:)* = 100.0 : Radius of circular boundaring box, listed for each slice (:) or (slice number)
    - *bbox(:)* = 'rect' or 'rectangle': Rectangular bounding box. Requires *bbox1* and *bbox2* as inputs for width and height of the rectangle.
        - *bbox1(:)* = 100. : Width of rectangular boundaring box, listed for each slice (:) or (slice number)
        - *bbox2(:)* = 75.0 : Height of rectangular boundaring box, listed for each slice (:) or (slice number)

- **plane_crinkle_def**: Similar to **plane_slice_def** above but uses crinkle cut to produce plane of conforming cell faces. All parameters are the same with the following exceptions: The **outtype** field must be **replaced** by the **cut_type** field:

    - **cut_type**: Determines how crinkle-cut surface is defined. **cut_type=node"** selects faces with nodes on either side of cutting place. **cut_type=cell_center"** selects cells based on position of cell center relative to cutting plane.

  - **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
- **isosurface_def**: Defines isosurfaces to be output based on variable and value. Isosurfaces can also be colored by other specified variables.

  - **file_name**: Specifies the file name for the output file. Can include a relative or absolute pathname. As mentioned above, all post_proc data files are written to the subdirectory "post_proc" created by the solver at runtime in the ./WRK directory. Various examples are given in the sample file listing below. The file format can be specified also using the file_format key word described below. The available file formats are described in the next section. If no suffix is recognized, vtk format is assumed.
  - **file_format**: Specifies the file format. This can be used instead of specifying the file suffix in the file_name above. This will also over-ride any file format specification in the file_name specification. If no file format is recognized, vtk format is assumed.
  - **file_binary**: For VTK formats, setting **file_binary=0** produces an ASCII or readable text file. Default is binary file (**file_binary=1**). Has no effect on tecplot files which are always binary.
  - **varname**: These are the flowfield variables to be output on the iso_surface. The variables are listed in any order separated by commas. Misspelt or non-existing variable names are ignored. A list of available types is given in the next section. Only the variables with **is3Dvar=1** can be used.
  - **iso_varname**: The flowfield variable used to extract the iso_surface. Misspelt or non-existing variable name is ignored. A list of available types is given in the next section. Only the variables with **is3Dvar=1** can be used.
  - **iso_value**: The value of the flowfield variable used to extract iso_surface (i.e. surface value).
  - **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
  - **frequency**: The frequency parameter can be used to override the default frequency set at the top of the input.postproc file.
- **point_probe_def**: Defines data extracted at the input point locations.

  - **file_name**: Specifies the file name for the output file. Can include a relative or absolute pathname. As mentioned above, all post_proc data files are written to the subdirectory "post_proc" created by the solver at runtime in the ./WRK directory. Various examples are given in the sample file listing below. The file format can be specified also using the file_format key word described below. The available file formats are described in the next section. If no suffix is recognized, vtk format is assumed.
  - **file_format**: Specifies the file format. This can be used instead of specifying the file suffix in the file_name above. This will also over-ride any file format specification in the file_name specification. If no file format is recognized, vtk format is assumed.
  - **file_binary**: For VTK formats, setting **file_binary=0** produces an ASCII or readable text file. Default is binary file (**file_binary=1**). Has no effect on tecplot files which are always binary.
  - **varname**: These are the flowfield variables to be output. The variables are listed in any order separated by commas. Misspelt or non-existing variable names are ignored. A list of available types is given in the next section.
  - **frequency**: The frequency parameter can be used to override the default frequency set at the top of the input.postproc file.

    - **pts**: XYZ coordinates of the input probe points. The maximum number of zones is 50 and each zone can consist of maximum 100 points
- **line_probe_def**: Defines data extracted on a line. There are four types of line definitions. The maximum number of lines is 50.
    - **file_name**: Specifies the file name for the output file. Can include a relative or absolute pathname. As mentioned above, all post_proc data files are written to the subdirectory "post_proc" created by the solver at runtime in the ./WRK directory. Various examples are given in the sample file listing below. The file format can be specified also using the file_format key word described below. The available file formats are described in the next section. If no suffix is recognized, vtk format is assumed.
    - **file_format**: Specifies the file format. This can be used instead of specifying the file suffix in the file_name above. This will also over-ride any file format specification in the file_name specification. If no file format is recognized, vtk format is assumed.
    - **file_binary**: For VTK formats, setting **file_binary=0** produces an ASCII or readable text file. Default is binary file (**file_binary=1**). Has no effect on tecplot files which are always binary.
    - **varname**: These are the flowfield variables to be output. The variables are listed in any order separated by commas. Misspelt or non-existing variable names are ignored. A list of available types is given in the next section.
    - **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
    - **frequency**: The frequency parameter can be used to override the default frequency set at the top of the input.postproc file.
    - **npts**: The number of points on one line. The default number is 51 points
    - **pt_begin**: 1st point
    - **pt_end**: last point
    - **norm**: line direction
    - **delta** : 1st space
    - **ratio** : growth ratio
    - **line_type**:
        - **uniform**: the point distribution is uniform and required two input points (pt_begin and pt_end)
        - **stretch_1pt_norm_delta_ratio**: the point distribution is stretched and required one input point **pt_begin**, the line direction **norm**, the 1st space **delta** and the growth ratio **ratio**
        - **stretch_2pt_delta**: the point distribution is stretched and required two input points **pt_begin pt_end** and the 1st space **delta**. The growth ratio is computed.
        - **stretch_2pt_delta_ratio**: the point distribution is stretched and required two input points **pt_begin pt_end**, the 1st space **delta** and the growth ratio **ratio**. The last point is approximated.
        - **line_type** = **stretch_1pt_wall_delta_ratio**: the point distribution is stretched and required one input point **pt_begin**, the 1st space **delta** and the growth ratio **ratio**. The code will search the viscous wall and find the nearest point to the input point. The wall normal direction of that point is used as the line direction.
- **face_probe_def**: Defines data extracted on the rectangle face. The maximum number of faces is 50. The rectangle is defined by three input points.
    - **file_name**: Specifies the file name the for output file. Can include a relative or absolute pathname. As mentioned above, all post_proc data files are written to the subdirectory "post_proc" created by the solver at runtime in the ./WRK directory. Various examples are given in the sample file listing below. The file format can be specified also using the file_format key

word described below. The available file formats are described in the next section. If no suffix is recognized, vtk format is assumed.

- **file_format**: Specifies the file format. This can be used instead of specifying the file suffix in the file_name above. This will also over-ride any file format specification in the file_name specification. If no file format is recognized, vtk format is assumed.
- **file_binary**: For VTK formats, setting **file_binary=0** produces an ASCII or readable text file. Default is binary file (**file_binary=1**). Has no effect on tecplot files which are always binary.
- **varname**: These are the flowfield variables to be output. The variables are listed in any order separated by commas. Misspelt or non-existing variable names are ignored. A list of available types is given in the next section.
- **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
- **frequency**: The frequency parameter can be used to override the default frequency set at the top of the input.postproc file.
- **face_type**: only suppored type is **uniform** : the points on the face are evenly distributed in width and height directions
- **pt1**: xyz coordinates of lower left point of the rectangle
- **pt2**: xyz coordinates of lower right point of the rectangle
- **pt3**: xyz coordinates of upper left point of the rectangle
- **npts1**: number of points in the width direction
- **npts2**: number of points in the height direction

Listing 1: Example of user input file for F6 regression test case: *input.postproc*

```
!-------------------------------------------------
 !input file for nsu3d post-process
!-------------------------------------------------
&data_def
! frequency = 0    !<--Never (commented out here)
! frequency = -1   !<--At end only. Note this is the default value if not specified
  frequency = 100  !<--Every 100 cycs and at end
/
!-------------------------------------------------
 &volume_def
   file_name     = 'vol'
   varname       = 'rho, rhou, rhov, rhow, rhoe, turb1'
  frequency      = -1
!-------------------------------------------------
&surface_def
 file_name     = 'out_comp_.plt'
 outtype       = 'comp'
 bound_def     = 'WING, FUSELAGE'
 varname       = 'cp, mach'
/
!-------------------------------------------------
&surface_def
  file_name     = 'out_bc_'
  file_format   = 'plt'
  outtype       = 'bcs'
  bound_def     = 'all'
  varname       = 'cp, mach'
/
!-------------------------------------------------
&surface_def
  file_name     = 'out_patch_.szplt'
  outtype       = 'patch'
  bound_def     = '6:10,17:25,27:43'
  varname       = 'yplus, cf , cfx , cfy , cfz'
/
!-------------------------------------------------
&plane_slice_def
  file_name      = 'slice_circle_.szplt'
  varname        = 'cp, mach'
  outtype        = 'field'
  origin(:,1)    = 140.,-80.0,0.0
  origin(:,2)    = 155.,-130.0,4.7
  origin(:,3)    = 167.,-180.0,12.0
  origin(:,4)    = 180.,-230.0,18.0
  origin(:,5)    = 195.,-280.0,22.5
  origin(:,6)    = 226.,-330.0,26.5
  origin(:,7)    = 247.,-380.0,30.5
  origin(:,8)    = 268.,-430.0,35.0
  origin(:,9)    = 288.,-480.0,41.0
  origin(:,10)   = 307,-530.0,45.0
  origin(:,11)   = 327.,-580.0,50.
  normal(1,:)    = 11*0.0,
  normal(2,:)    = 11*1.0,
  normal(3,:)    = 11*0.0,
  frequency      = -1
```

```
!
   bbox(:)       = 11*'circle',
   bbox1(1)      = 100.,
   bbox1(2)      = 95. ,
   bbox1(3)      = 90. ,
   bbox1(4)      = 80. ,
   bbox1(5)      = 75. ,
   bbox1(6)      = 70. ,
   bbox1(7)      = 65. ,
   bbox1(8)      = 60. ,
   bbox1(9)      = 55. ,
   bbox1(10)     = 50. ,
   bbox1(11)     = 45.,
/
!-------------------------------------------------
&plane_slice_def
  file_name      = 'slice_rectangle_.szplt'
  varname        = 'cp, mach'
  outtype        = 'field'
  origin(:,1)    = 140.,-80.0,0.0
  origin(:,2)    = 155.,-130.0,4.7
  origin(:,3)    = 167.,-180.0,12.0
  origin(:,4)    = 180.,-230.0,18.0
  origin(:,5)    = 195.,-280.0,22.5
  origin(:,6)    = 226.,-330.0,26.5
  origin(:,7)    = 247.,-380.0,30.5
  origin(:,8)    = 268.,-430.0,35.0
  origin(:,9)    = 288.,-480.0,41.0
  origin(:,10)   = 307,-530.0,45.0
  origin(:,11)   = 327.,-580.0,50.
  normal(1,:)    = 11*0.0,
  normal(2,:)    = 11*1.0,
  normal(3,:)    = 11*0.0,
  frequency      = -1
!
   bbox(:)       = 11*'rectangle',
!
   bbox1(1)      = 90. ,
   bbox1(2)      = 85. ,
   bbox1(3)      = 80. ,
   bbox1(4)      = 75. ,
   bbox1(5)      = 70. ,
   bbox1(6)      = 65. ,
   bbox1(7)      = 60. ,
   bbox1(8)      = 55. ,
   bbox1(9)      = 50. ,
   bbox1(10)     = 45. ,
   bbox1(11)     = 40.,
!
   bbox2(1)      = 120.,
   bbox2(2)      = 115. ,
   bbox2(3)      = 100.,
   bbox2(4)      = 90. ,
   bbox2(5)      = 85. ,
   bbox2(6)      = 80. ,
   bbox2(7)      = 75. ,
   bbox2(8)      = 60. ,
```

```
   bbox2(9)      = 65. ,
   bbox2(10)     = 60. ,
   bbox2(11)     = 55.,
/
!-----------------------------------------------------
&plane_slice_def
  file_name      = 'slice_bound_.szplt'
  varname        = 'cp cf force'
  outtype        = 'comp'
  bound_def      = 'FUSELAGE,  WING'
  origin(:,1)    = 140.,-80.0,0.0
  origin(:,2)    = 155.,-130.0,4.7
  origin(:,3)    = 167.,-180.0,12.0
  origin(:,4)    = 180.,-230.0,18.0
  origin(:,5)    = 195.,-280.0,22.5
  origin(:,6)    = 226.,-330.0,26.5
  origin(:,7)    = 247.,-380.0,30.5
  origin(:,8)    = 268.,-430.0,35.0
  origin(:,9)    = 288.,-480.0,41.0
  origin(:,10)   = 307,-530.0,45.0
  origin(:,11)   = 327.,-580.0,50.
  normal(1,:)    = 11*0.0,
  normal(2,:)    = 11*1.0,
  normal(3,:)    = 11*0.0,
  frequency      = -1
/
!-----------------------------------------------------
 &point_probe_def

  file_name      = 'point_'
  file_format    = 'plt'
  varname        = 'cp, mach, rhou,rhov,rhow'

  pts(:,:,1) = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
               2.285727809E+02, -8.000000000E+01, -9.996889337E+00,
               2.285724849E+02, -8.000000000E+01, -1.005945839E+01,
               2.285721888E+02, -8.000000000E+01, -1.012202744E+01,
               2.285718927E+02, -8.000000000E+01, -1.018459649E+01,

  pts(:,:,2) =  2.400446104E+02, -8.000000000E+01, -1.041767556E+01,
                2.400446104E+02, -8.000000000E+01, -1.021767556E+01,
                2.400446104E+02, -8.000000000E+01, -1.001767556E+01,
                2.400600239E+02, -8.000000000E+01, -9.972865147E+00,
                2.400754373E+02, -8.000000000E+01, -9.928054734E+00,
                2.400908508E+02, -8.000000000E+01, -9.883244322E+00,
                2.401062643E+02, -8.000000000E+01, -9.838433909E+00,
                2.401216777E+02, -8.000000000E+01, -9.793623496E+00,
 /
!-----------------------------------------------------
 &line_probe_def

  file_name      = 'line_'
  file_format    = 'plt'
  varname        = 'cp, mach, rhou,rhov,rhow'
!
  line_type(1)   = 'uniform'
  pt_begin(:,1)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
```

```
  pt_end(  :,1)  = 2.285585696E+02, -8.000000000E+01, -1.300020376E+01,

!
  line_type(2)   = 'stretch_2pt_delta'
  pt_begin(:,2)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
  pt_end(  :,2)  = 2.285585696E+02, -8.000000000E+01, -1.300020376E+01,
  delta(2)       = 0.02
!
  line_type(3)   = 'stretch_2pt_delta_ratio'
  pt_begin(:,3)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
  pt_end(  :,3)  = 2.285585696E+02, -8.000000000E+01, -1.300020376E+01,
  delta(3)       = 0.002
  ratio(3)       = 1.15
!
  line_type(4)   = 'stretch_1pt_wall_delta_ratio'
  pt_begin(:,4)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
  delta(4)       = 0.002
  ratio(4)       = 1.15
 /
!-----------------------------------------------------
 &line_probe_def

  file_name      = 'line_'
  file_format    = 'vtk'
  varname        = 'cp, mach, uvel,vvel,wvel'
 !
  line_type(1)   = 'uniform'
  pt_begin(:,1)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
  pt_end(  :,1)  = 2.285585696E+02, -8.000000000E+01, -1.300020376E+01,

 !
  line_type(2)   = 'stretch_2pt_delta'
  pt_begin(:,2)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
  pt_end(  :,2)  = 2.285585696E+02, -8.000000000E+01, -1.300020376E+01,
  delta(2)       = 0.02
 !
  line_type(3)   = 'stretch_2pt_delta_ratio'
  pt_begin(:,3)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
  pt_end(  :,3)  = 2.285585696E+02, -8.000000000E+01, -1.300020376E+01,
  delta(3)       = 0.002
  ratio(3)       = 1.15
 !
  line_type(4)   = 'stretch_1pt_wall_delta_ratio'
  pt_begin(:,4)  = 2.285730770E+02, -8.000000000E+01, -9.934320286E+00,
  delta(4)       = 0.002
  ratio(4)       = 1.15
 /
!-----------------------------------------------------
&line_probe_def
!
  file_name      = 'line5_'
  file_format    = 'szplt'
  varname        = 'cp, mach, uvel,vvel,wvel'

  line_type(1:5)  = 5*'stretch_1pt_wall_delta_ratio'
  delta(1:5)      = 5*0.002
  ratio(1:5)      = 5*1.15
```

```
  pt_begin(:,1)  = 68.6931, -130, 15.7146,
  pt_begin(:,2)  = 89.2447, -130, 20.5009,
  pt_begin(:,3)  = 119.25,  -130, 20.1047,
  pt_begin(:,4)  = 178.091, -130, 13.5398,
  pt_begin(:,5)  = 236.476, -130, -0.111608
/
!-------------------------------------------------
 &face_probe_def
 !
  file_name      = 'face_'
  file_format    = 'szplt'
  varname        = 'cp, mach, uvel,vvel,wvel'

  face_type(1)  = 'uniform'

  pt1(:,1)  = 250., -250, -17.,
  pt2(:,1)  = 250., -210, -17.,
  pt3(:,1)  = 250., -250,  20.,

  npts1(1)  = 21
  npts2(1)  = 21
 /
!-------------------------------------------------
 &isosurface_def
  file_name      = 'iso_mach_'
  file_format    = 'szplt'
  iso_varname    = 'mach'
  iso_value(1:2) = 0.9, 1.0
  varname        = 'mach, cp, pressure, rhou, rhov, rhow '
  frequency      = 100
 /
!-------------------------------------------------
 &isosurface_def
  file_name      = 'iso_vort_'
  file_format    = 'szplt'
  iso_varname    = 'vort_x'
  iso_value(1:2) = -0.002, -0.004,
  varname        = 'pressure, mach, rhou, rhov, rhow '
  frequency      = -1
 /
!-------------------------------------------------
```

Reference Section

List of Supported File Types

The output file names listed by the file_name parameter use the file suffix to control the format of the resulting file. Alternatively, this can be specified in the **file_format** key word entry, which also overrides the suffix specified in the output file name. Four general file formats are supported: tecplot *.dat (formatted or ASCII) and *szplt (binary) and VTK (formatted and binary). VTK type files are the default, and produced when the file format or suffix is not specified or not recognized, whereas tecplot file output requires that the tecio parallel library is linked to NSU3D. Specifying the parameter **file_binary=0** produces an ASCII or readable text file for VTK files. The default is binary file (**file_binary=1**). This parameter has no effect on tecplot files which are always binary unless the tecplot *.dat format is specified.

The default location for writing out all post_proc data files is in the subdirectory post_proc created by the solver at runtime in the ./WRK directory. Note that, if it exists, the ./WRK/post_proc directory is purged and re-created at startup of the flow solver. Output files can still be written to locations outside of this directory by specifying file names with pathnames relative to the ./WRK/post_proc directory (for example ../../my_output/slice.szplt)

- **Filetypes**:
    - .dat: Tecplot formatted (i.e. ASCII readable) file. (Not recommended for large files).
    - .plt: Tecplot binary file (requires tecio library). For parallel execution, these revert to .szplt files.
    - .szplt: Tecplot szplt parallel binary file (requires tecio library)
    - .vtk: vtk file with xml format, with extension name pvtu(volume) or vtm(surface or slice)
        - Default is binary. Can be output as ASCII text files using **file_binary=0**
    - No extension, or unrecognized: append vtk type file extensions and write out vtk based files.
    - the time step number or iteration number will be appended on the file name.

List of Supported definition and plot types

- **data_def**: General inputs and default values for all plot types
    - **frequency** : Defines default frequency for output of all plot types. Note that additional **frequency** parameters may be set in each individual plot type to override the default value set here.
        - **frequency = 100** : Output every 100 cycles AND at end of run
        - **frequency = 0** : Never output
        - **frequency = -1** : Output only at end of run
- **volume_def**: Outputs entire volume flow field data
    - **file_name types**: .szplt, .vtk
    - **varname**: See varname definition in next section with is3Dvar=1
- **surface_def**: Outputs values on a surface entity
    - **file_name types**: .dat, .szplt, .plt, .vtk,
    - **outtype**: patch, comp, bcs
    - **bound_def**: Must correspond to patch, comp or bc type in mesh, see the definition in next sections
    - **varname**: See varname definition in next section
- **plane_slice_def**: Outputs values on one or more planar slices through flow field or through a surface entity
    - **file_name types**: .dat, .szplt, .plt, .vtk
    - **outtype**: volume (or field), or a surface type: patch, comp, bcs
    - **bound_def**: Must correspond to patch, comp or bc type in mesh, see the definition in next sections
    - **mesh_id**: For overset mesh cases only and only for plane_slice_def cases with **outtype** = volume or field. Selects one of more component meshes. Default value is "all"
    - **varname**: See varname definition in next section
        - The Varname **force** can also be used to trigger a plot of integrated forces across all the slices in a "plane_slice_def". Two additional output files are called "force_wind.file_name" and "forces_xyz.file_name". See description below. Note this feature only applies to slices of surface entities and is ignored for slices of "volume" or "field" as specified in the "outtype" key word defined above.

- **origin**: XYZ coordinates of point in plane. 2D array defined as origin(1:3,1:NSLICE). (also serves as center for bounding box below)
- **normal**: Normal vector for plane definition. 2D array defined as normal(1:3,1:NSLICE).
- **bbox**: Bounding box. An optional bounding box (rectangle or circle) can be defined for each field slice beyond which the mesh and flow field data are cropped. The following options can be used:
    - *bbox(:)* = ' ', no bounding box (default value). Also, if *bbox* keyword is not present no bounding box is used.
    - *bbox(:)* = 'circle': Creates a circular bounding region. Requires *bbox1* as an input for the circle radius:
        - *bbox1(:)* = 100.0 : Radius of circular boundaring box, listed for each slice (:) or (slice number)
    - *bbox(:)* = 'rect' or 'rectangle': Rectangular bounding box. Requires *bbox1* and *bbox2* as inputs for width and height of the rectangle.
        - *bbox1(:)* = 100. : Width of rectangular boundaring box, listed for each slice (:) or (slice number)
        - *bbox2(:)* = 75.0 : Height of rectangular boundaring box, listed for each slice (:) or (slice number)
- **plane_crinkle_def**: Similar to **plane_slice_def** above but uses crinkle cut to produce plane of conforming cell faces. All parameters are the same with the following exceptions: the **outtype** field must be **replaced** by the **cut_type** field:
    - **cut_type**: Determines how crinkle-cut surface is defined. **cut_type=node"** selects faces with nodes on either side of cutting place. **cut_type=cell_center"** selects cells based on position of cell center relative to cutting plane.
    - **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
- **isosurface_def**: Outputs values on an iso_surface entity
    - **file_name types**: .dat, .szplt, .plt, .vtk,
    - **varname**: Variables to be output on the iso_surface. Only variables of type **is3Dvar=1** can be used.
    - **iso_varname**: The flowfield variable used to extract the iso_surface. Misspelt or non-existing variable name is ignored. A list of available types is given in the next section. Only variables with **is3Dvar=1** can be used.
    - **iso_value**: The value of the flowfield variable (iso_varname) used to extract iso_surface (i.e. surface value).
    - **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
- **point_probe_def**: Defines data extracted at the defined probe points.
    - **file_name types**: .dat, .szplt, .plt, .vtk,
    - **varname**: See varname definition in previous section
    - **pts**: XYZ coordinates of input points. The maximum number of zones is 50 and each zone can consist of maximum 100 points
    - **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
- **line_probe_def**: Defines data extracted along a line. There are four types of line definitions.
    - **file_name types**: .dat, .szplt, .plt, .vtk,

- **varname**: See varname definition in previous section
- **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"
- **line_type**:
  - **uniform**: the point distribution is uniform and requires two input points: **pt_begin** and **pt_end**
  - **stretch_1pt_norm_delta_ratio**: the point distribution is stretched and requires one input point **pt_begin**, the line direction **norm**, the 1st space **delta** and the growth ratio **ratio**
  - **stretch_2pt_delta**: the point distribution is stretched and requires two input points **pt_begin pt_end**, and the 1st space **delta**. The growth ratio is computed.
  - **stretch_2pt_delta_ratio**: the point distribution is stretched and requires two input points **pt_begin pt_end**, the 1st space **delta** and the growth ratio **ratio**. The last point is approximated.
  - **stretch_1pt_wall_delta_ratio**: the point distribution is stretched and requires one input point **pt_begin**, the 1st space **delta** and the growth ratio **ratio**. The code will search the viscous surface and find the nearest point to the input point. The wall normal direction of that point is used as the line direction.
- **face_probe_def**: Defines data extracted on a rectangular face.
  - **file_name types**: .dat, .szplt, .plt, .vtk,
  - **varname**: See varname definition in previous section
  - **face_type** = **uniform** is supported now
  - **pt1**: xyz coordinates of lower left point of the rectangle
  - **pt2**: xyz coordinates of lower right point of the rectangle
  - **pt3**: xyz coordinates of upper left point of the rectangle
  - **npts1**: number of points in the width direction
  - **npts2**: number of points in the height direction
  - **mesh_id**: For overset mesh cases only. Selects one of more component meshes. Default value is "all"

Outtype definition

- **volume or field**:

  No further parameters. This option only valid for **plane_slice_def** (not valid for **surface_def**)

- **patch**:

  bound_def must include a list of patch numbers. Example patch definition:

  - 'ALL' or 'all' or ' : ' or ': all patches'
  - ' 3:10 ' : patch list from No.3 to No.10
  - ' 1, 4, 10' : patch No.1, No.4 and No. 10
  - ' 1, 3:10, 20, 22' : patch list from No.3 to No.10 and No.1, No.20 and No.22

- **comp**:

  bound_def must include a list of component names. The component names should be consistent with the comp file.

  - for example: 'FUSELAGE, WING', use 'ALL' or 'all' to include all components

- **bcs**:

bound_def must include a list of boundary condition names from the following: (use 'ALL' or 'all' to include all bcs):

- "iobc1"
- "iobc2"
- "iobc3"
- "itbc"
- "insbc"
- "ifin"
- "ifout"
- "icore"
- "iprbc"
- "isymbc"

List of supported variable names

varname	keyword	is3Dvar
Density	"density","DENSITY","rho","RHO"	1
x-momentum	"rhou","RHOU"	1
y-momentum	"rhov","RHOV"	1
z-momentum	"rhow","RHOW"	1
Energy	"rhoe","RHOE","energy","ENERGY"	1
u velocity (in x-direction)	"u", "U", "uvel"	1
v velocity (in y-direction)	"v", "V", "vvel"	1
w velocity (in z-direction)	"w", "W", "wvel"	1
Turbulence variable 1	"turb1","TURB1"	1
Turbulence variable 2	"turb2","TURB2"	1
Turbulence variable 3	"turb3","TURB3"	1
Turbulence variable 4	"turb4","TURB4"	1
IBLANK: For overset mesh cases	"iblank ","IBLANK"	1
Mach number	"ma","Ma","MA","mach","MACH"	1
Pressure	"pressure","pre","p","PRESSURE","PRE","P"	1
Coefficient of Pressure	"cp","Cp","CP"	1
Coefficient of Total Pressure	"cptotal","cpt","Cpt","CPT","CpT"	1
Temperature	"temperature","temp","Temp","TEMP","t","T"	1
Entropy	"entropy","ENTROPY"	1
Skin Friction Coefficient	"cf","Cf","CF"	0
x-y-z-direction Cf	"cfx","CFx","CFX"	0
Yplus	"yplus","Yplus","y+","Y+","YPLUS"	0

Eddy	"eddy","ed","EDDY","ED"	1
Itmask: Masking variable for specified transition	"itmask","ITMASK"	1
Wall distance function for Turbulence modeling	"dist","DIST","Dist"	1
Heatflux (at wall)	"heatflux","HeatFlux","HEATFLUX","hflux","Hflux","HFLUX"	0
Vorticity	"vorticity","VORTICITY","vort","Vort","VORT","omega","Omega"	1
x-component of vorticity	"vorticity_x","VORTICITY_X","vort_x","VORT_X","omegax","vortx","VORTX"	1
y-component of vorticity	"vorticity_y","VORTICITY_Y","vort_y","VORT_Y","omegay","vorty","VORTY"	1
z-component of vorticity	"vorticity_z","VORTICITY_Z","vort_z","VORT_Z","omegaz","vortz","VORTZ"	1
Q-criterion	"q_criterion","Q_criterion"	1
x-component of grid velocity	"vel_grid_x","v_grid_x","velx_grid"	1
y-component of grid velocity	"vel_grid_y","v_grid_y","vely_grid"	1
z-component of grid velocity	"vel_grid_z","v_grid_z","velz_grid"	1
x-component of surface grid velocity	"bvel_x"	0
y-component of surface grid velocity	"bvel_y"	0
z-component of surface grid velocity	"bvel_z"	0
Force: Triggers integrated force plot across all slices in a plane_slice_def group (i.e. for plotting wing span loads) (automatically triggers output of Cp and Cf)	"force","FORCE"	0

Addendum for Overset Mesh Cases

For overset mesh cases, the default post_process behavior is to output quantities on all meshes of the overset mesh set. The **mesh_id** parameter can be used to output data for certain plot types for individual meshes of the set if needed.

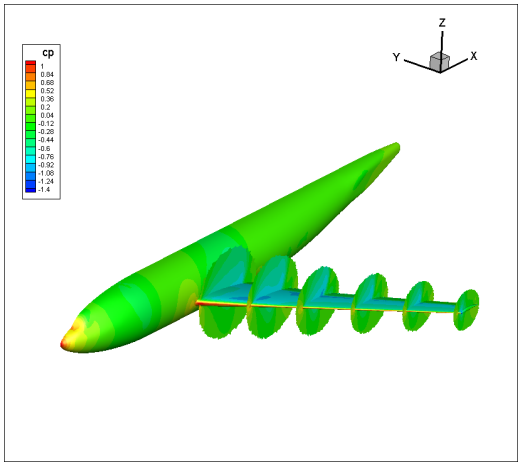
- **volume_def:** For overset meshes, all mesh cells and data are output to a single file. Outputting volume data for individual meshes is currently not supported. This can be done with the sequential post-processing routines, using post_nsu3d and tecform individually for each mesh and restart file.
- **surface_def:** Surface plots cannot specify individual meshes of an overset mesh set. The surface quantities specified such as bcs type will be output for all meshes. This applies also for patch numbers. Specifying a specific patch number may result in multiple patches with the same number belonging to

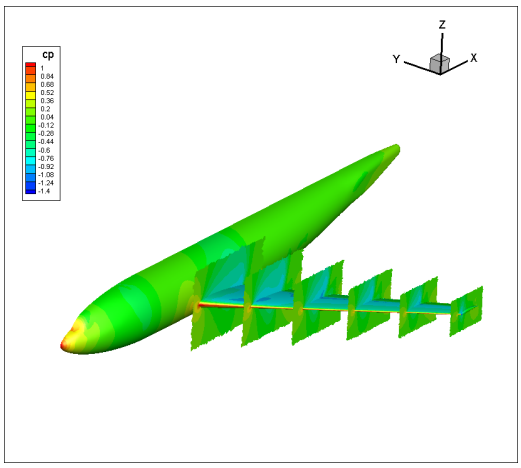
different individual overset meshes being written to the same file. However, for components, NSU3D renames each component from each mesh, appended with the mesh_id number. Therefore, when specifying components to be output to file, the component name that is specific to a given overset mesh can be specified.

- **plane_slice_def:**
    - For plane_slice_def with **outtype=volume** (or field), the mesh_id parameter can be used to isolate output for a particular mesh in the overset mesh sequence. mesh_id can specify one or any number of meshes in the set. Default value is "all".
    - For plane_slice_def with **outtype=** patch, comp or bcs, mesh_id is not functional. The output behaves similarly to the surface_def files in this case where **bound_def** is used to specify particular bcs, patch numbers of component names that apply to all meshes. However, component numbers are specific to each overset mesh as described above.
- **plane_crinkle_def:** The mesh_id parameter can be used to isolate output for a particular mesh in the overset mesh sequence. mesh_id can specify one or any number of meshes in the set. Default value is "all"
- **iso_surface_def:** The mesh_id parameter can be used to isolate output for a particular mesh in the overset mesh sequence. mesh_id can specify one or any number of meshes in the set. Default value is "all"
- **point_probe_def:** The mesh_id parameter can be used to isolate output for a particular mesh in the overset mesh sequence. mesh_id can specify one or any number of meshes in the set. Default value is "all"
- **line_probe_def:** The mesh_id parameter can be used to isolate output for a particular mesh in the overset mesh sequence. mesh_id can specify one or any number of meshes in the set. Default value is "all"
- **plane_probe_def:** The mesh_id parameter can be used to isolate output for a particular mesh in the overset mesh sequence. mesh_id can specify one or any number of meshes in the set. Default value is "all"

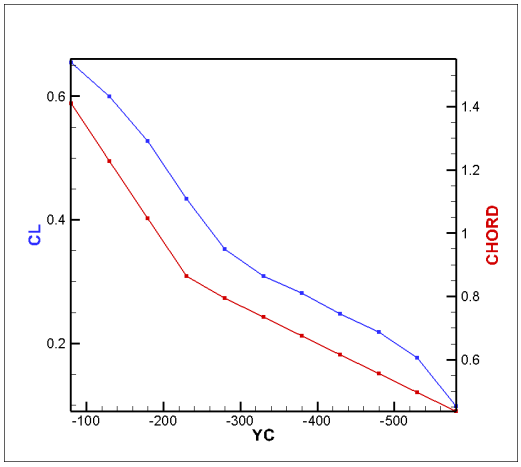
Example Test Case:

Illustration of post_proc slices to examine wing span loads

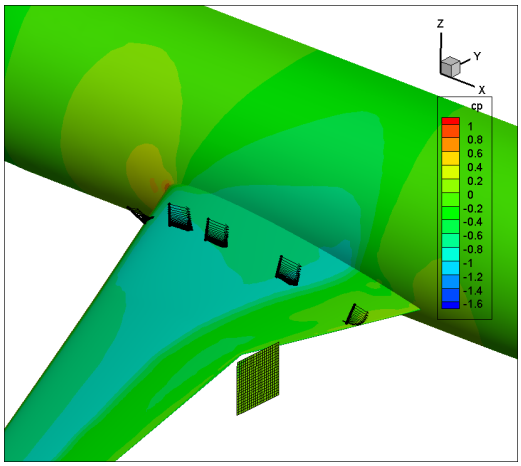


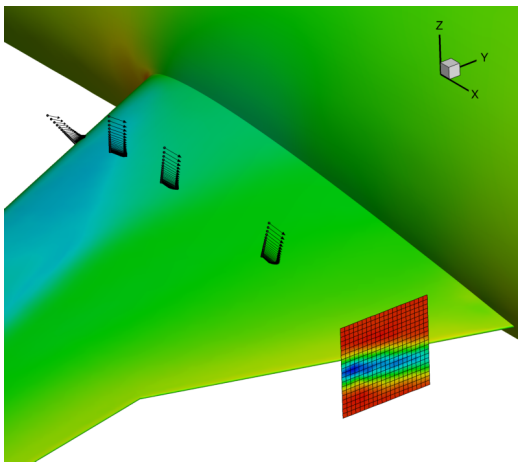


Resulting plot from surface_def and slices defined in plane_slice_def using circle bounding boxes (left) and rectangular bounding boxes (right). 11 slices in total, with only every second slice shown.

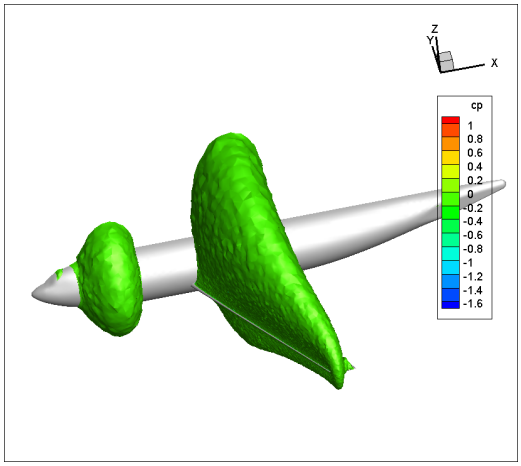


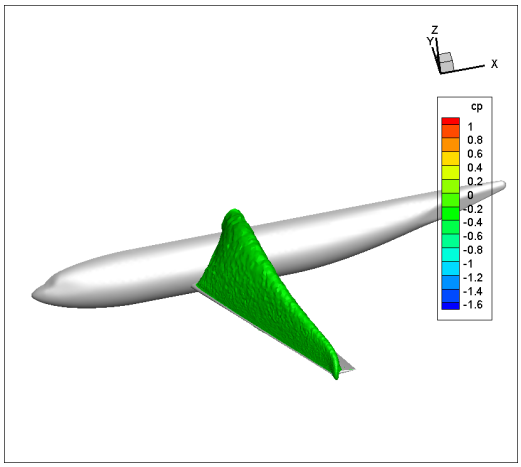
Spanwise lift distribution and local chord obtained from all 11 post_proc slice outputs



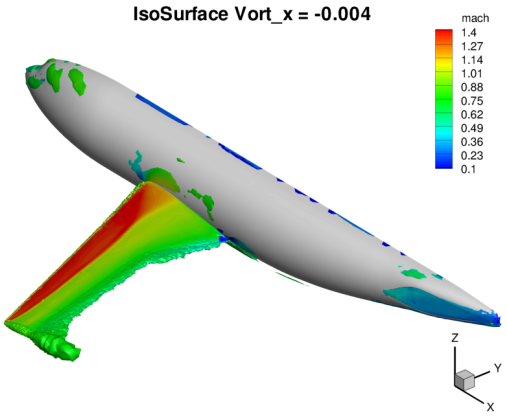


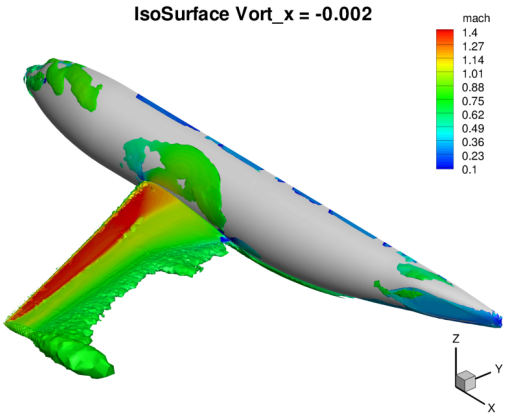
Line probe and face probe





Mach=0.9 isosurface and Sonic (Mach=1) isosurface colored by Cp





IsoSurface with Vort_x, colored by Mach number (using Roe scheme solution)