
Flow360 Documentation

Flexcompute Inc

Nov 29, 2021

TABLE OF CONTENTS

1	Quick Start	3
2	Capabilities	17
3	Interfaces	21
4	Case Studies	23
5	API Reference	47
6	Webinar	57
7	Frequently Asked Questions	59
8	White Paper	61

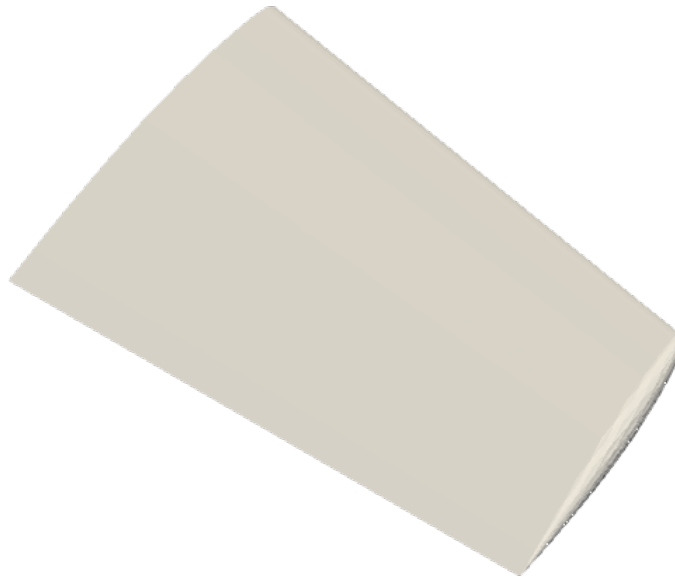
Flow360 is the next-generation Navier-Stokes solver. It is 100 times faster than industry-leading solvers, with the same high-fidelity and reduced cost. It expedites your time to market, shortens design cycles, reduces development cost, and facilitates more comprehensive testing to avoid program delays.

QUICK START

1.1 Run CFD using Web UI: An example of ONERA M6 Wing

The Onera M6 wing is a classic CFD validation case for external flows because of its simple geometry combined with complexities of transonic flow (i.e. local supersonic flow, shocks, and turbulent boundary layer separation). It is a swept, semi-span wing with no twist and uses a symmetric airfoil using the ONERA D section. More information about the geometry can be found at [NASA's website](#). The geometry parameters are:

- Mean Aerodynamic Chord (MAC) = 0.80167
- Semi-span = 1.47602
- Reference area = 1.15315



The mesh used for this case contains 113K nodes and 663K tetrahedrons, and the flow conditions are:

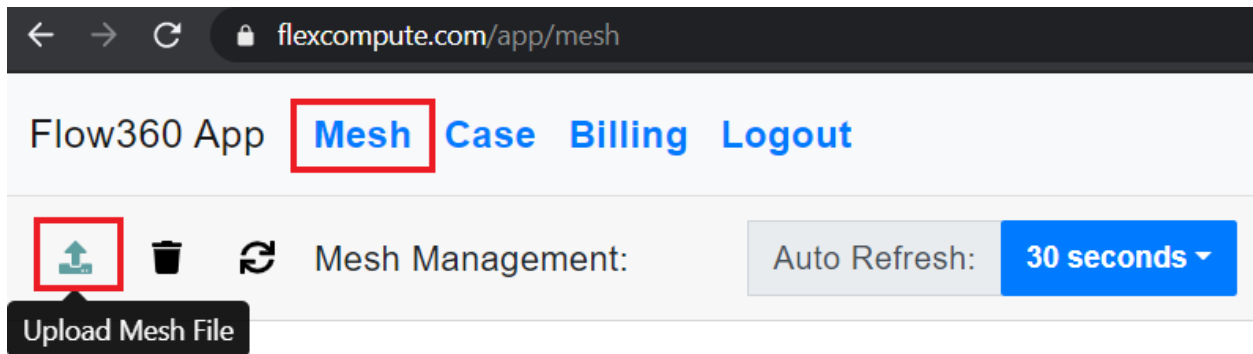
- Mach Number = 0.84
- Reynolds Number (based on MAC) = 11.72 Million
- Alpha = 3.06°.
- Reference Temperature = 297.78 K

1.1.1 Get CFD results in two simple steps

After you sign in, visit <https://client.flexcompute.com/app/mesh>.

Step 1. Upload the Mesh File:

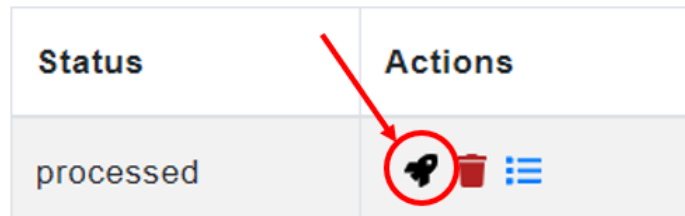
To upload a mesh file, under the Mesh tab click on Upload Mesh File icon as displayed below:



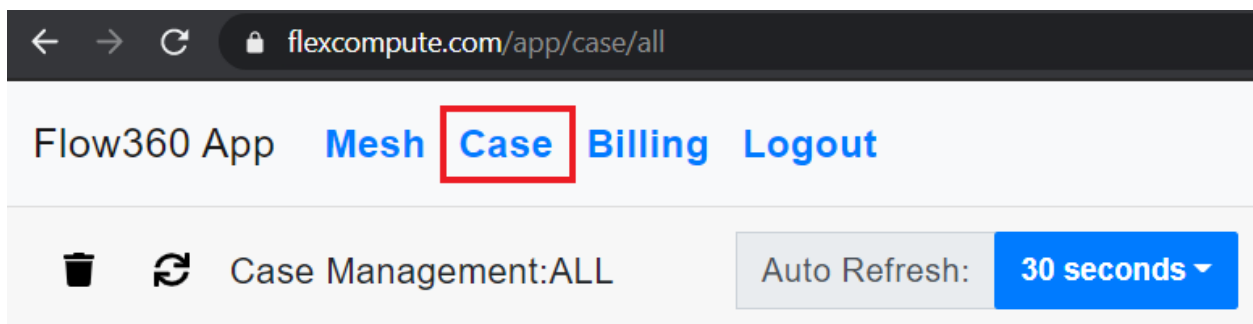
This will open a window. You need to upload two files here. The first is the mesh. Download an example mesh of M6 Wing [here](#). The second is the mesh configure json. Download one [here](#). When uploading the mesh, be sure to set the Endianness to 'little endian' for this tutorial. More information on endianness can be found [here](#). (**Note:** Mesh Name and Tags are optional.) Then click submit to upload and process your mesh. You can see the status of the mesh under the **Status** tab.

Step 2. Launch a CFD Case:

You do not have to wait for your mesh to be processed. Once your mesh is uploaded you can start the case submission process. To start a new case, click on the airplane icon under the **Actions** tab.



Once you click the airplane icon, another window will pop up to give you an option to choose a corresponding [Flow360.json](#) configuration file for your case. A full dictionary of configuration parameters for the JSON input file can be found [here](#). You may also provide a Case Name and Tags. Hit Submit to run your case. Once the case is submitted, you can check the case status by clicking the **Case** tab.



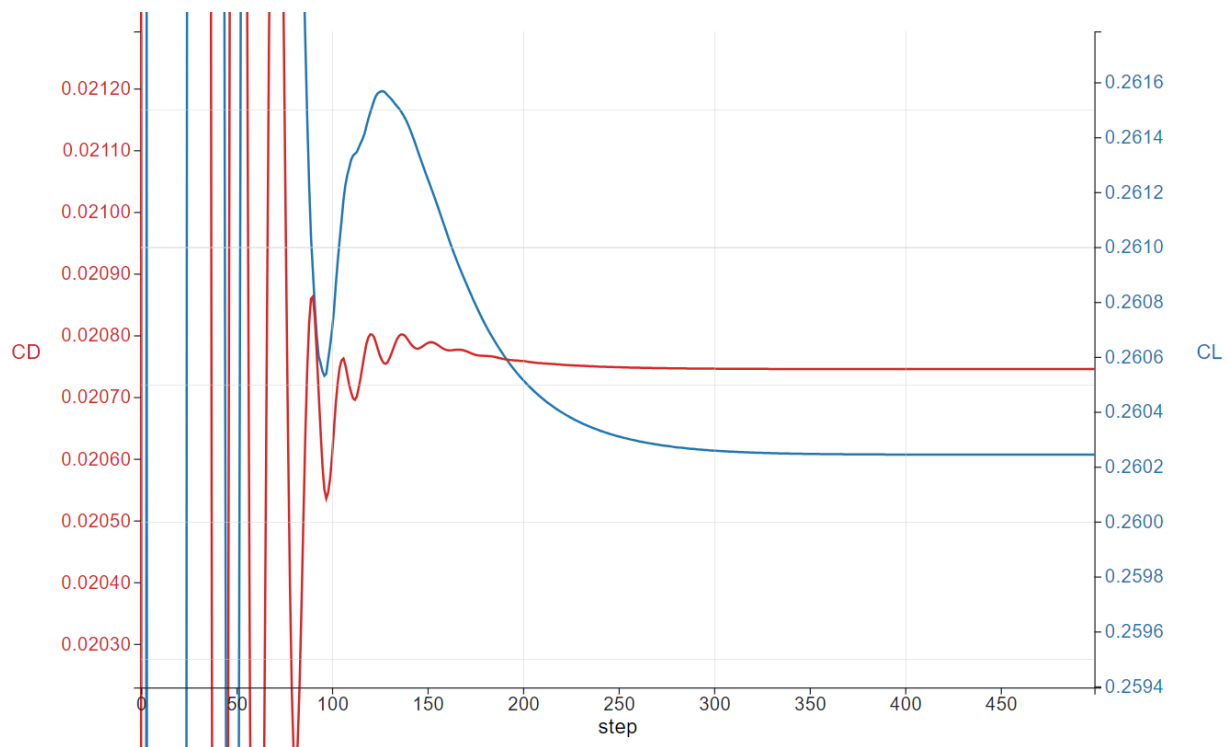
1.1.2 Advanced Functions

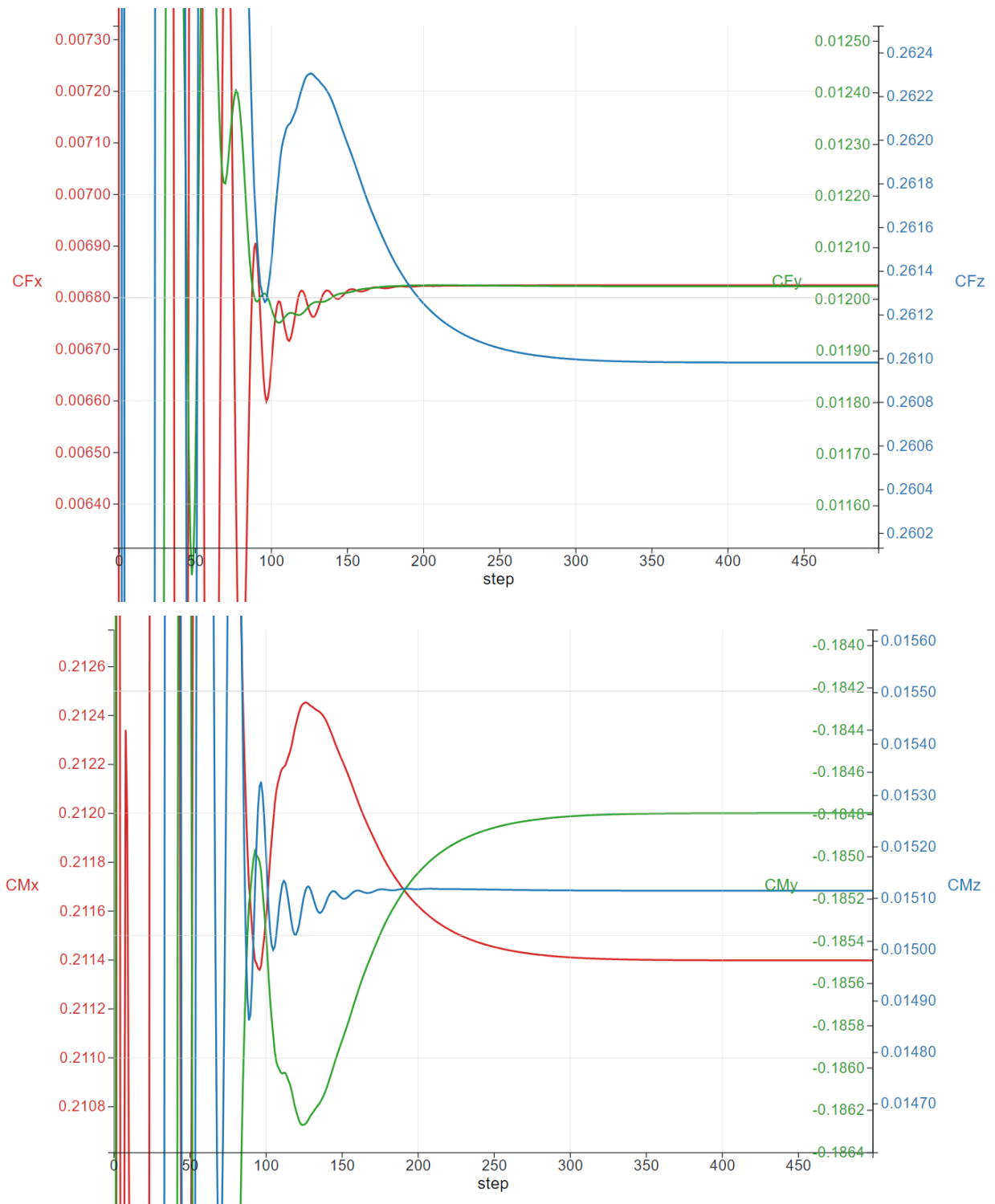
Visualizing the Results:

While your case is running, or after that, you can visualize the Residuals and Forces plot by clicking on your case name and viewing them under the **Convergence** and **Forces** tabs, respectively.

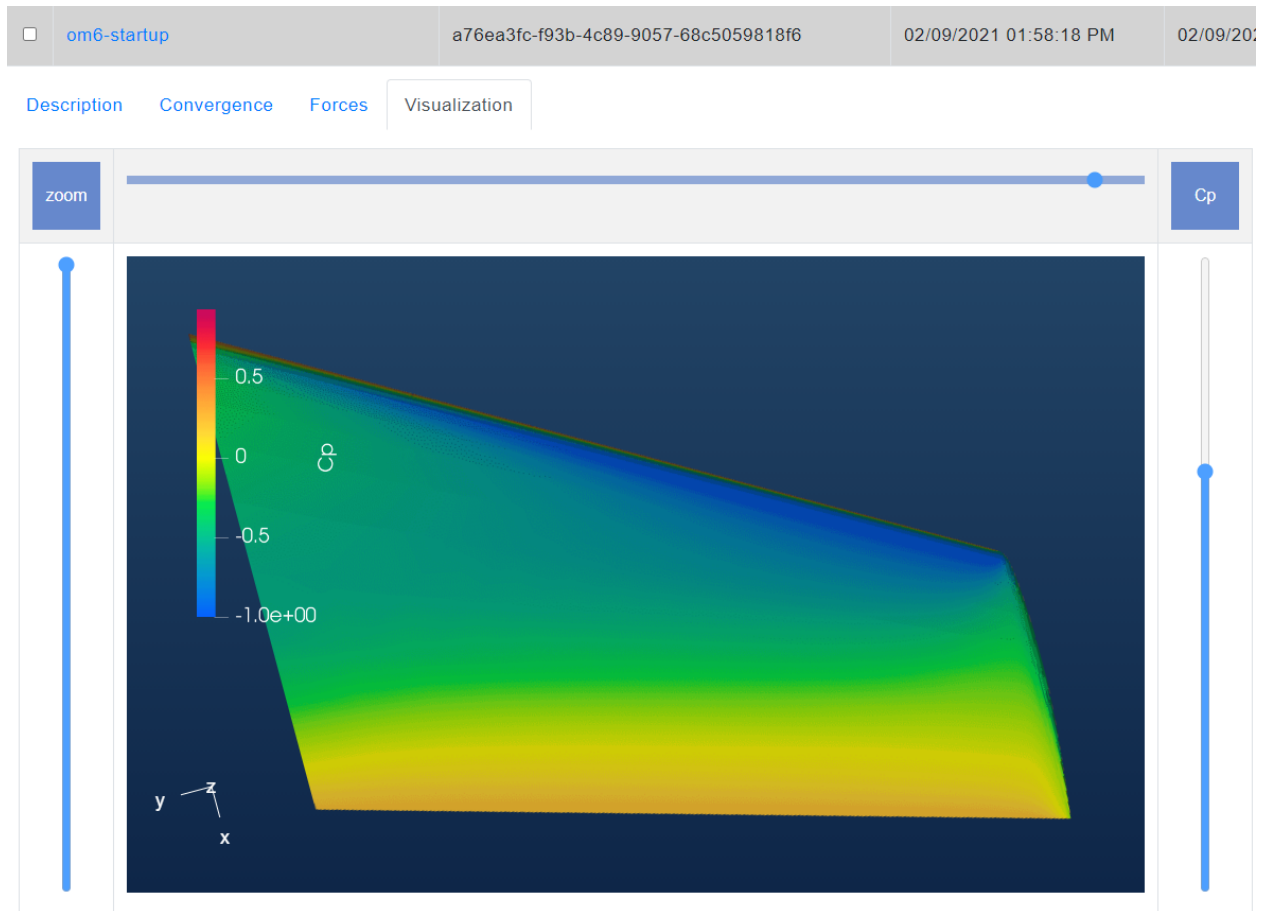


For example, the Forces plots for this case are:







Once your case has completed running, you can also visualize the contour plots of the results under the **Visualization** tab. Currently, contour plots for Coefficient of Pressure (C_p), Coefficient of Skin Friction (C_f), Y^+ , and C_f with streamlines are provided.



Downloading the Results:




Once your case has finished running, you can download the results (Surfaces, Volume and Log) by clicking the download arrow under the **Actions** tab.

Status	Action
completed	<div>Download</div> <div> </div>

The Volume and Surface data can be post-processed in either ParaView or Tecplot, based on your selection of outputFormat in the Flow360.json file. And the Log file (solver.out) can be used to view the solver time and other run details.

Fork a Case:

You can also restart a case, to continue from the last point of the previous run, by clicking the **Fork a Case** icon under the **Actions** tab and change the parameters of your Flow360.json configuration file according to your needs or upload an entirely new configuration file.

Status	Action
completed	<div><div>Fork Case</div><div> </div><div></div></div>

Fork A Case ?



Fork Case Name

Enter Flow360Mesh.JSON or choose a file: *(Optional)*

```
{
  "geometry": {
    "refArea": 1.15315084119231,
    "momentCenter": [
      0,
      0,
      0
    ],
    "momentLength": [
      1.4760179762198,
      0.801672958512342,
      1.4760179762198
    ]
  },
}
```

JSON format text only

Choose File

No file chosen

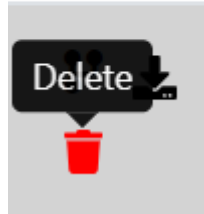
Submit

Close

Deleting a Mesh/Case:

You can delete a mesh/case by clicking on the trash can icon under the **Actions** tab. (*Caution:* You will not be able to recover your deleted case or mesh files including its results after your deletion.)

Action



1.2 Run CFD using Python API: An example of ONERA M6 Wing

The Onera M6 wing is a classic CFD validation case for external flows because of its simple geometry combined with complexities of transonic flow (i.e. local supersonic flow, shocks, and turbulent boundary layer separation). It is a swept, semi-span wing with no twist and uses a symmetric airfoil using the ONERA D section. More information about the geometry can be found at [NASA's website](#). The geometry parameters are:

- Mean Aerodynamic Chord (MAC) = 0.80167
- Semi-span = 1.47602
- Reference area = 1.15315



The mesh used for this case contains 113K nodes and 663K tetrahedrons, and the flow conditions are:

- Mach Number = 0.84
- Reynolds Number (based on MAC) = 11.72 Million

- Alpha = 3.06°.
- Reference Temperature = 297.78 K

1.2.1 Upload the Mesh File

Before uploading a mesh, if you have not done so already, install the Flow360 Python API client. For installation instructions, visit the *API Reference* section of this documentation. To upload a mesh and run a case, open your Python interpreter and import the flow360 client.

```
python3
import flow360client
```

Specify no-slip boundaries - you can do this in three ways:

- a. By directly feeding it into the noSlipWalls argument:

```
noSlipWalls = [1]
```

- b. By using the .mapbc file:

```
noSlipWalls = flow360client.noSlipWallsFromMapbc('/path/to/fname.mapbc')
```

(Note: Make sure the boundary names in your .mapbc file do NOT contain any spaces)

- c. And by using the meshJson object:

```
import json
meshJson = json.load(open('/path/to/Flow360Mesh.json'))
```

The Flow360Mesh.json file for this tutorial has the following contents:

```
{
  "boundaries" :
  {
    "noSlipWalls" : [1]
  }
}
```

Download the mesh file from [here](#). If using options (a) and (b), use the following command to upload your mesh:

```
meshId = flow360client.NewMesh(fname='/path/to/mesh.lb8.ugrid',
                               noSlipWalls=noSlipWalls,
                               meshName='my_mesh',
                               tags=[],
                               endianness='little'
                               )
```

If using option (c), use the following command to upload your mesh:

```
meshId = flow360client.NewMesh(fname='/path/to/mesh.lb8.ugrid',
                               meshJson=meshJson,
                               meshName='my_mesh',
                               tags=[],
                               endianness='little'
                               )
```

(**Note:** Arguments of meshName and tags are optional. If using a mesh filename of the format mesh.lb8.ugrid (little-endian format) or mesh.b8.ugrid (big-endian format), the endianness argument is optional. However, if you choose to use a mesh filename of the format mesh.ugrid, you must specify the appropriate endianness ('little' or 'big') in NewMesh. More information on endianness can be found [here](#).)

Currently supported mesh file formats are .ugrid, .cgns and their .gz and .bz2 compressions. The mesh status can be checked by using:

```
## to list all your mesh files
flow360client.mesh.ListMeshes()
## to view a particular mesh
flow360client.mesh.GetMeshInfo('mesh_Id')
```

Replace the mesh_Id with your mesh's ID.

1.2.2 Run a Case

The Flow360.json file for this case has the following contents. A full dictionary of configuration parameters for the JSON input file can be found [here](#).

```
1 {
2   "geometry" : {
3     "meshName" : "wing_tetra.1.lb8.ugrid",
4     "refArea" : 1.15315084119231,
5     "momentCenter" : [0.0, 0.0, 0.0],
6     "momentLength" : [1.47602, 0.801672958512342, 1.47602]
7   },
8   "runControl" : {
9     "restart" : false
10  },
11  "volumeOutput" : {
12    "outputFormat" : "tecplot",
13    "animationFrequency" : -1,
14    "primitiveVars" : true,
15    "vorticity" : false,
16    "residualNavierStokes" : false,
17    "residualTurbulence" : false,
18    "T" : false,
19    "s" : false,
20    "Cp" : false,
21    "mut" : false,
22    "mutRatio" : false,
23    "Mach" : true
24  },
25  "surfaceOutput" : {
26    "outputFormat" : "tecplot",
27    "animationFrequency" : -1,
28    "primitiveVars" : true,
29    "Cp" : true,
30    "Cf" : true,
31    "CfVec" : false,
32    "yPlus" : false,
33    "wallDistance" : false
```

(continues on next page)

(continued from previous page)

```

34 },
35 "sliceOutput" : {
36   "outputFormat" : "tecplot",
37   "animationFrequency" : -1,
38   "primitiveVars" : true,
39   "vorticity" : true,
40   "T" : true,
41   "s" : true,
42   "Cp" : true,
43   "mut" : true,
44   "mutRatio" : true,
45   "Mach" : true
46 },
47 "navierStokesSolver" : {
48   "absoluteTolerance" : 1e-10,
49   "kappaMUSCL" : -1.0
50 },
51 "turbulenceModelSolver" : {
52   "modelType" : "SpalartAllmaras",
53   "absoluteTolerance" : 1e-8,
54   "kappaMUSCL" : -1.0
55 },
56 "freestream" :
57 {
58   "Reynolds" : 14.6e6,
59   "Mach" : 0.84,
60   "Temperature" : 288.15,
61   "alphaAngle" : 3.06,
62   "betaAngle" : 0.0
63 },
64 "boundaries" : {
65   "1" : {
66     "type" : "NoSlipWall",
67     "name" : "wing"
68   },
69   "2" : {
70     "type" : "SlipWall",
71     "name" : "symmetry"
72   },
73   "3" : {
74     "type" : "Freestream",
75     "name" : "freestream"
76   }
77 },
78 "timeStepping" : {
79   "maxPseudoSteps" : 500,
80   "CFL" : {
81     "initial" : 5,
82     "final" : 200,
83     "rampSteps" : 40
84   }
85 }

```

(continues on next page)

(continued from previous page)

86

}

Use this JSON configuration file and run the case with the following command:

```
caseId = flow360client.NewCase(meshId='mesh_Id',
                               config='/output/path/for/Flow360.json',
                               caseName='my_case',
                               tags=[])
)
```

(**Note:** Arguments of meshName and tags are optional.)

The case status can be checked by using:

```
## to list all your cases
flow360client.case.ListCases()
## to view a particular case
flow360client.case.GetCaseInfo('case_Id')
```

Replace the mesh_Id and case_Id with your mesh and case IDs, respectively.

1.2.3 Deleting a Mesh/Case

An uploaded mesh/case can be deleted using the following commands (*Caution:* You will not be able to recover your deleted case or mesh files including its results after your deletion):

```
## Delete a mesh
flow360client.mesh.DeleteMesh('')
## Delete a case
flow360client.case.DeleteCase('')
```

1.2.4 Download the Results

To download the surface data (surface distributions and slices) and the entire flow field, use the following command lines, respectively:

```
flow360client.case.DownloadSurfaceResults('case_Id', 'surfaces.tar.gz')
flow360client.case.DownloadVolumetricResults('case_Id', 'volume.tar.gz')
```

Once downloaded, you can postprocess these output files in either Tecplot or ParaView. You can specify this in the Flow360.json file under the volumeOutput, surfaceOutput, and sliceOutput sections.

To download the solver.out file, use the following command:

```
flow360client.case.DownloadSolverOut('case_Id', fileName='solver.out')
```

You can also download the nonlinear residuals, surface forces and total forces by using the following command line:

```
flow360client.case.DownloadResultsFile('case_Id', 'fileName.csv')
```

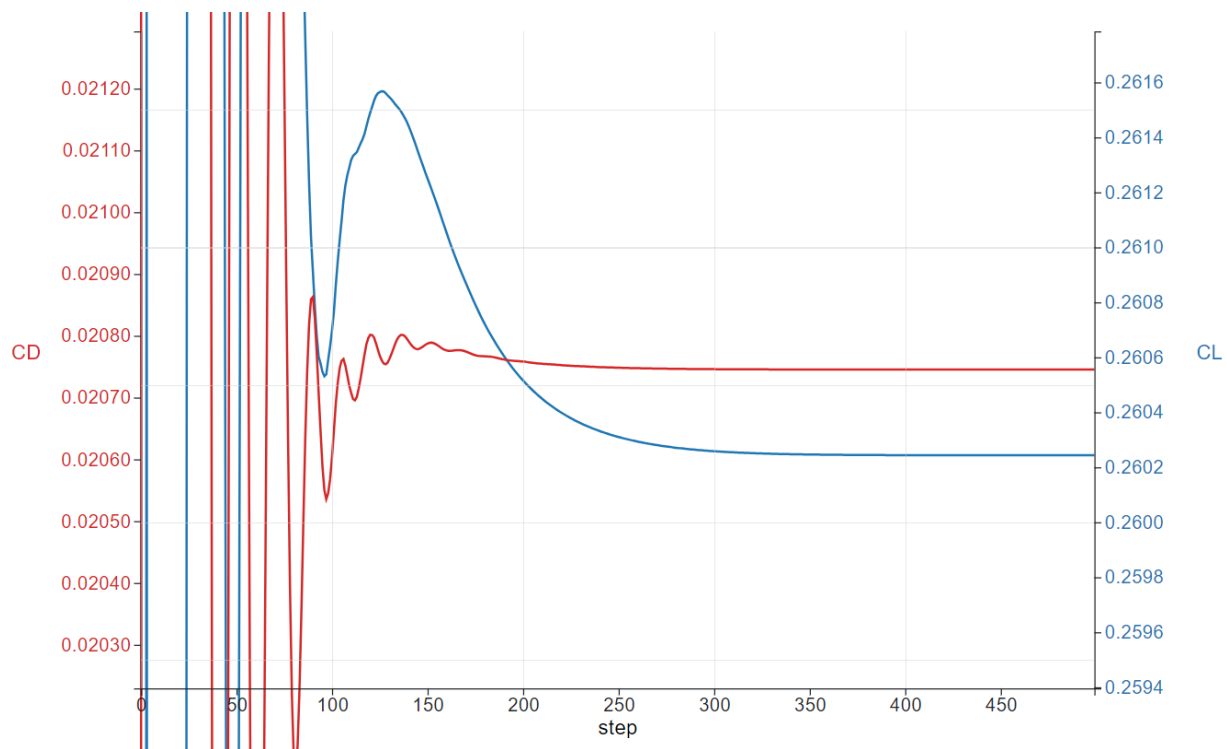
Replace caseId with your caseId and fileName with nonlinear_residual, surface_forces and total_forces for their respective data.

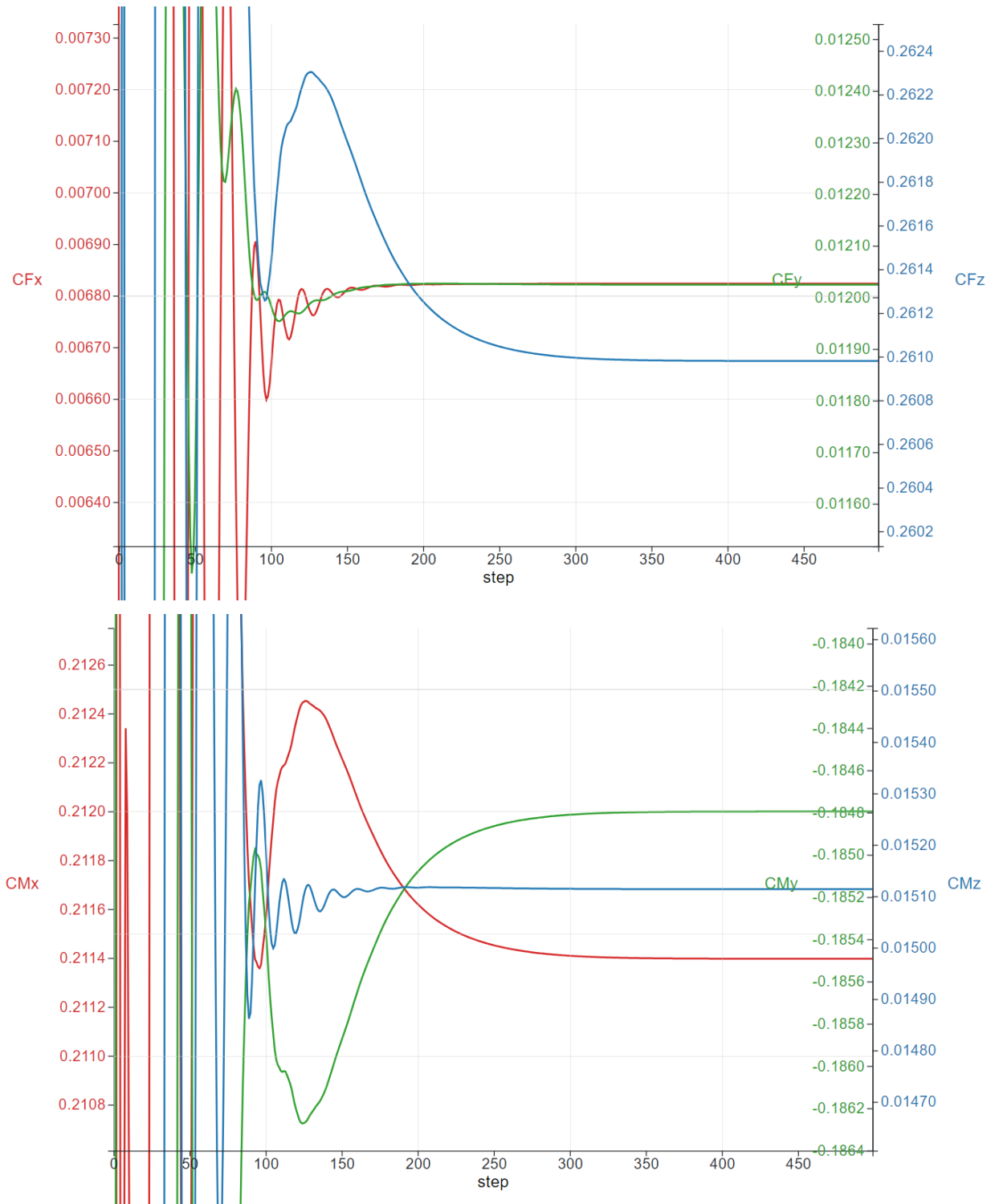
1.2.5 Visualizing the Results

While your case is running, or after that, you can visualize the Residuals and Forces plot on our website (<https://client.flexcompute.com/app/case/all>) by clicking on your case name and viewing them under the **Convergence** and **Forces** tabs, respectively.

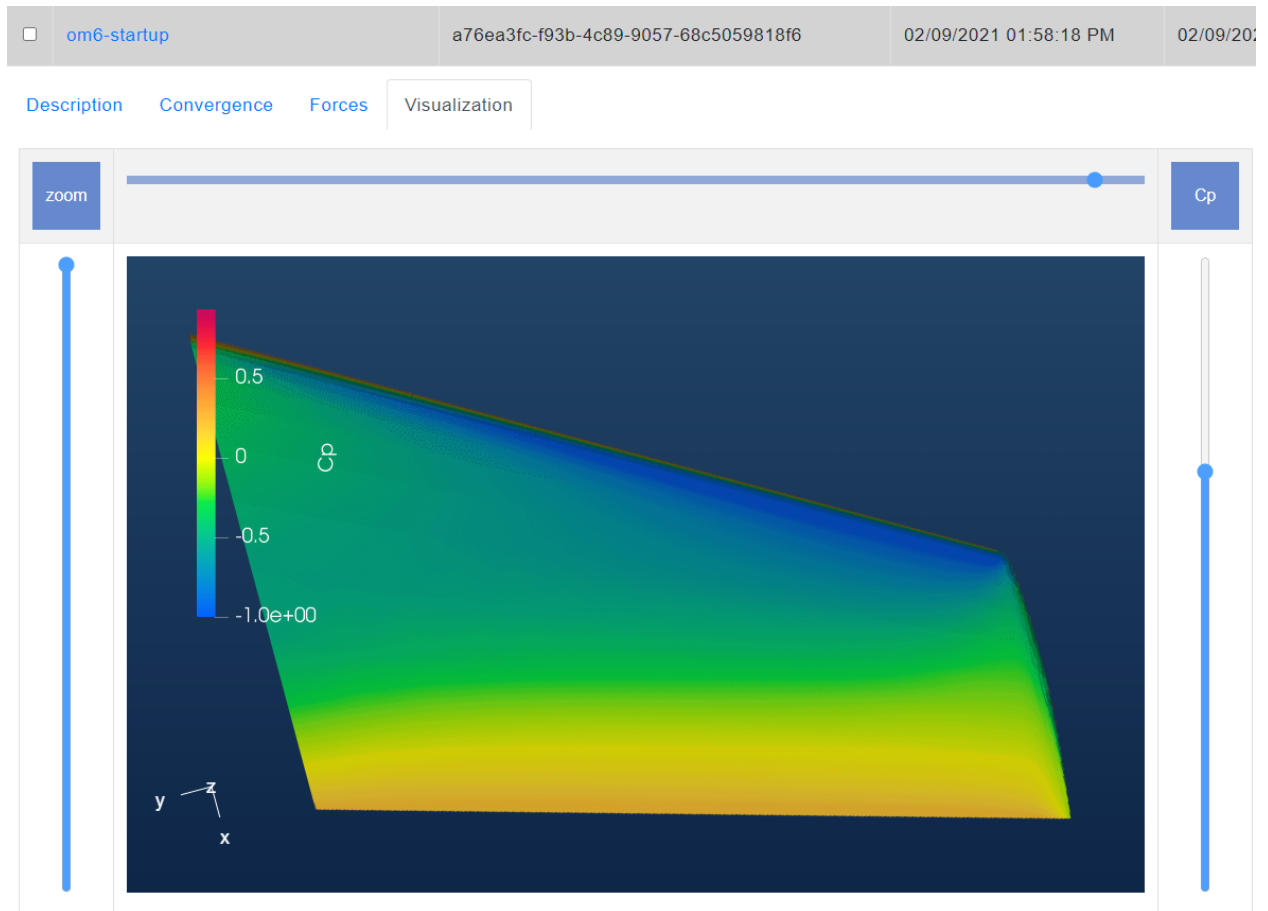


For example, the Forces plots for this case are:





Once your case has completed running, you can also visualize the contour plots of the results under the **Visualization** tab. Currently, contour plots for Coefficient of Pressure (C_p), Coefficient of Skin Friction (C_f), Y^+ , and C_f with streamlines are provided.



CAPABILITIES

2.1 Overview

2.1.1 Equations

- Steady and unsteady viscous flows.
- Coupled stationary and rotating domains.
- Reynolds-Averaged Navier-Stokes.
- Detached Eddy Simulation.
- Laminar-Turbulent Transition model.
- *Blade Element Theory model*.
- Actuator Disk model.

2.1.2 Turbulence Models

- Spalart-Allmaras (sa-neg).
- Spalart-Allmaras with Rotation-Curvature correction.
- k- SST.

2.1.3 Boundary Conditions

- Freestream (with optional location dependent velocity).
- Slip Wall.
- No-slip Wall (with optional tangential wall velocity).
- Isothermal Wall.
- Subsonic Inflow (total pressure, total temperature).
- Subsonic Outflow (back pressure).
- Subsonic Outflow (Mach).
- Mass Flow In.
- Mass Flow Out.

2.2 Blade Element Theory Model

2.2.1 Overview

Based on Blade Element Theory, Flow360 provides 2 related solvers, which can be configured in *BETDisks* section of Flow360.json:

- Steady blade disk solver

To use the steady blade disk solver, the `bladeLineChord` needs to be set as 0, which is its default value if omitted.

- Unsteady blade line solver

To use the unsteady blade line solver, `bladeLineChord` has to be a positive value and `initialBladeDirection` also needs to be set.

In the *BETDisks* section of the Flow360.json, except the `bladeLineChord` and `initialBladeDirection`, other parameters are necessary for both solvers.

2.2.2 BET Loading Output

After the simulation is completed, a “bet_forces_v2.csv” file is created for the case, which contains the time history of the following quantities:

1. Integrated x-, y-, z-component of non-dimensional forces and non-dimensional moments acted on each disk, represented by “Disk[diskID]_Force_x,y,z” and “Disk[diskID]_Moment_x,y,z” in the “bet_forces_v2.csv file” respectively. The non-dimensional force is defined as

$$\text{Force}_{\text{non-dimensional}} = \frac{\text{Force}_{\text{physical}}(\text{SI}=\text{N})}{\rho_{\infty} C_{\infty}^2 L_{\text{gridUnit}}^2} \quad (2.2.1)$$

The non-dimensional moment is defined as

$$\text{Moment}_{\text{non-dimensional}} = \frac{\text{Moment}_{\text{physical}}(\text{SI}=\text{N}\cdot\text{m})}{\rho_{\infty} C_{\infty}^2 L_{\text{gridUnit}}^3}, \quad (2.2.2)$$

where the moment center is the `centerOfRotation` of each disk, defined in *BETDisks* of Flow360.json.

2. Sectional thrust coefficient C_t and sectional torque coefficient C_q on each blade at several radial locations, represented by “Disk[diskID]_Blade[bladeID]_R[radialID]” with suffix “_Radius”, “_ThrustCoeff” and “_TorqueCoeff”. The number of radial locations is specified in `nLoadingNodes`.

The definition of C_t is

$$C_t(r) = \frac{\text{Thrust per unit blade length (SI=N/m)}}{\frac{1}{2}\rho_{\infty} ((\Omega r)^2) \text{chord}_{\text{ref}}} \cdot \frac{r}{R} \quad (2.2.3)$$

The definition of C_q is

$$C_q(r) = \frac{\text{Torque per unit blade length (SI=N)}}{\frac{1}{2}\rho_{\infty} ((\Omega r)^2) \text{chord}_{\text{ref}} R} \cdot \frac{r}{R} \quad (2.2.4)$$

where r is the distance between the node to the axis of rotation. $\text{chord}_{\text{ref}}$ is the dimensional reference chord length. R is the radius of the rotor disk.

Note: All the quantities in the right hand side of Eq.(2.2.1), Eq.(2.2.2), Eq.(2.2.3) and Eq.(2.2.4) are **dimensional**, which are different from the **non-dimensional** values in *BETDisks (list)* of Flow360.json.

Warning: For simulations of the steady blade disk solver, the resulting C_t and C_q are only saved on the first blade, named by “Blade0”. They are written as all zeros for other blades, because all the blades have the same sectional loadings in steady blade disk simulations. For the unsteady blade line solver, each blade has its own C_t and C_q values.

Here is an example of the header of a “bet_forces_v2.csv” file from a simulation containing two BET disks (assume `nLoadingNodes = 20`, `numberOfBlades = 3` for each disk):

```
physical_step, pseudo_step,
Disk0_Force_x, Disk0_Force_y, Disk0_Force_z, Disk0_Moment_x, Disk0_Moment_y, Disk0_
↪Moment_z,
Disk0_Blade0_R0_Radius, Disk0_Blade0_R0_ThrustCoeff, Disk0_Blade0_R0_TorqueCoeff,
Disk0_Blade0_R1_Radius, Disk0_Blade0_R1_ThrustCoeff, Disk0_Blade0_R1_TorqueCoeff,
...
Disk0_Blade0_R19_Radius, Disk0_Blade0_R19_ThrustCoeff, Disk0_Blade0_R19_TorqueCoeff,
Disk0_Blade1_R0_Radius, Disk0_Blade1_R0_ThrustCoeff, Disk0_Blade1_R0_TorqueCoeff,
Disk0_Blade1_R1_Radius, Disk0_Blade1_R1_ThrustCoeff, Disk0_Blade1_R1_TorqueCoeff,
...
Disk0_Blade1_R19_Radius, Disk0_Blade1_R19_ThrustCoeff, Disk0_Blade1_R19_TorqueCoeff,
Disk0_Blade2_R0_Radius, Disk0_Blade2_R0_ThrustCoeff, Disk0_Blade2_R0_TorqueCoeff,
Disk0_Blade2_R1_Radius, Disk0_Blade2_R1_ThrustCoeff, Disk0_Blade2_R1_TorqueCoeff,
...
Disk0_Blade2_R19_Radius, Disk0_Blade2_R19_ThrustCoeff, Disk0_Blade2_R19_TorqueCoeff,
Disk1_Force_x, Disk1_Force_y, Disk1_Force_z, Disk1_Moment_x, Disk1_Moment_y, Disk1_
↪Moment_z,
...
...
...
Disk1_Blade2_R19_Radius, Disk1_Blade2_R19_ThrustCoeff, Disk1_Blade2_R19_TorqueCoeff
```

2.2.3 BET Visualization

An additional option `betMetrics` in *volumeOutput* is available to visualize the BET related quantities.

INTERFACES

3.1 Input

- AFLR3 (.ugrid).
- CGNS (single and multi-block).

3.2 Output

- ParaView (.pvtu).
- Tecplot (.dat and .szplt).

3.3 User Interface

- Web Interface.
- *Python API*.

CASE STUDIES

4.1 High Lift Common Research Model (HL-CRM)

4.1.1 Introduction

The purpose of this case study is to compute the turbulent flow past a NASA Common Research Model (CRM) with a full flap gap from the AIAA CFD High Lift Prediction Workshop. The goal of the workshop is to research the state-of-the-art in difficult-to-predict aircraft configurations, such as take-off and landing scenarios. These cases may be harder to converge, and exhibit complex flow physics. The workshop allows comparisons to be made between different solvers and equation sets, and for performance and accuracy to be assessed.

In this case study, we will use mesh provided by the workshop committee and run Flow360 CFD solver developed by Flexcompute Inc. We will demonstrate how to upload a mesh, run a case and perform post processing. Performance of results from Flow360 will be compared to flow solutions from other leading open-source and commercial solvers, which are published in the [3rd AIAA CFD High Lift Prediction Workshop \(HiLiftPW-3\)](#).

4.1.2 Problem Description

The problem considers the flow around the HL-CRM at angles of attack $\alpha = 8^\circ$ and 16° and a freestream Mach number of 0.2 ($M_\infty = 0.2$). The geometry of the HL-CRM with gapped configuration can be downloaded from [here](#), and is shown in [Fig. 4.1.1](#). The airplane has mirror symmetry. As a common practice, only half of the plane is simulated. Summary of geometry parameters is given below.

- Mean aerodynamic chord (MAC) = 275.8 in, located at $y = 468.75$ in
- Wing semi-span ($b/2$) = 1156.75 in
- Reference area of the semi-span model = $S_{ref}/2 = 297,360.0 \text{ in}^2$
- Moment reference center (MRC): $x = 1325.90$ in, $y = 0.0$ in, $z = 177.95$ in
- Aspect Ratio (AR) = $b^2/S_{ref} = 9.0$

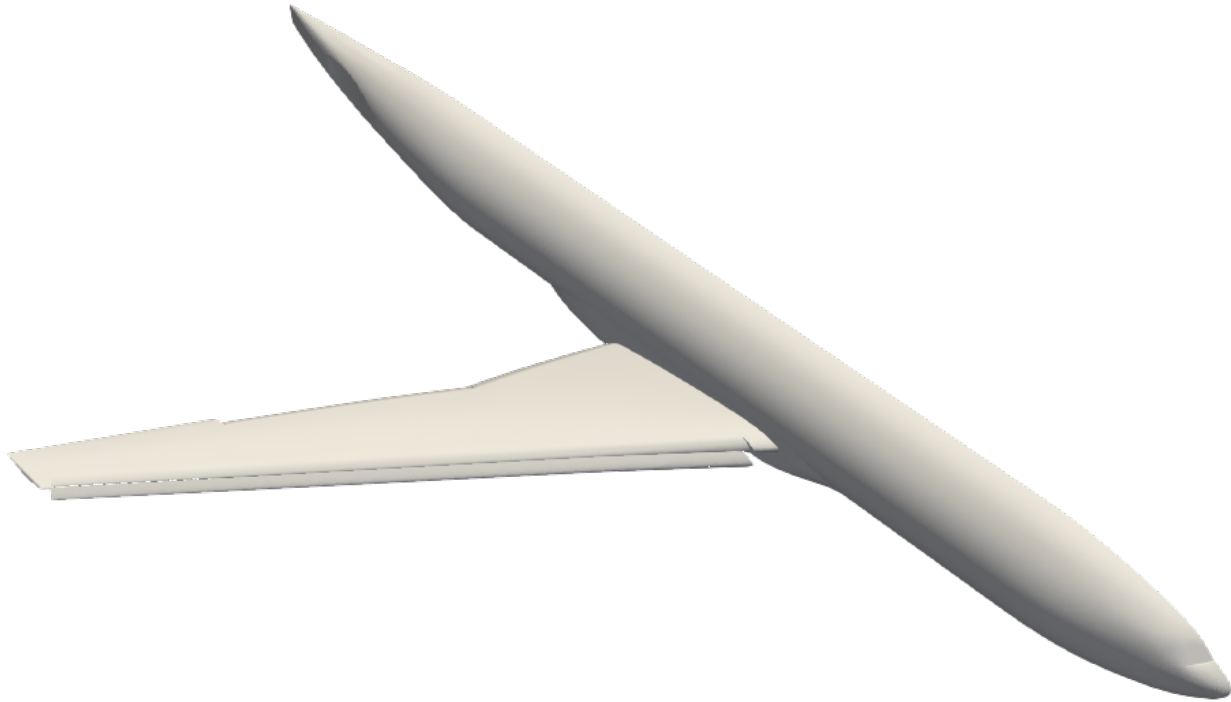


Fig. 4.1.1: Test geometry of the High Lift Prediction Workshop.

4.1.3 Mesh

Grids have been developed and provided by the workshop to enable a fair comparison across a variety of solvers. Flow360 supports mesh file formats in AFLR3 and CGNS, or their .gz or .bz2 compressions. For the purpose of this case study we will use ugrid mesh file, which is of the AFLR3 format. A medium-resolution mesh, with mixed hexahedral, tetrahedral, prism, and pyramid elements and with about 27M nodes, can be downloaded from [here](#). The file is already compressed in a .gz format. Also, download the .mapbc file, to specify the solid wall (no-slip) boundaries, from [here](#). Meshes of other resolutions can also be downloaded from [here](#).

4.1.4 Setup and Solution

Upload the Mesh File

Now that the [mesh](#) has been downloaded, it can be uploaded to the Flow360 cloud. We will do this using the Python API. Open the Python API and import the Flow360 client.

```
python3
import flow360client
```

Before we upload the mesh, we need to specify our no-slip boundary conditions. We will do this using the .mapbc file (downloaded from [here](#)). Make sure the boundary names in your .mapbc file do not have any spaces, otherwise the python code will not parse it. To specify the no-slip boundaries, use the following command line:

```
noSlipWalls = flow360client.noSlipWallsFromMapbc('output/path/for/hlcrm.mapbc')
```

Replace the file path with your own .mapbc file's path.

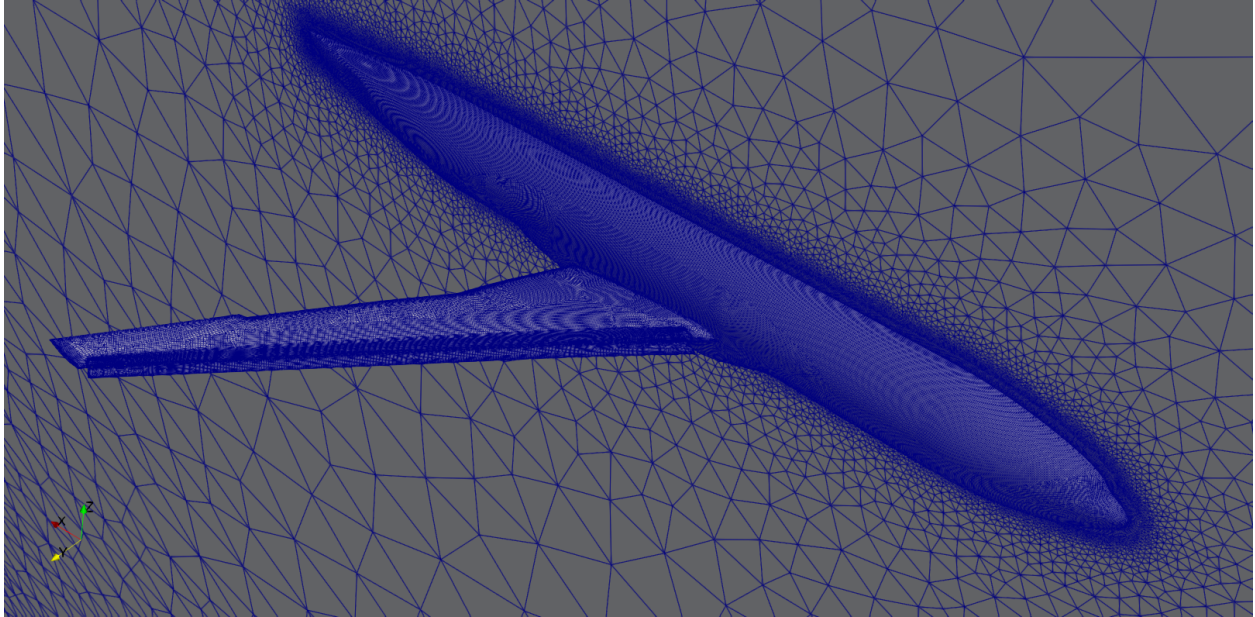


Fig. 4.1.2: Medium-resolution mesh of the HL-CRM; Outboard Flap (bottom-left) and Slat (bottom-right).

You can also specify the no-slip boundary conditions using the JSON mesh configuration files (download Flow360Mesh.json file from [here](#)). Use the following commands to do so:

```
import json
meshJson = json.load(open('/path/to/Flow360Mesh.json'))
```

Again, replace the file path with your own file's path. The Flow360Mesh.json file for this tutorial has the following contents:

```
{
  "boundaries" :
  {
    "noSlipWalls" : [2,3,5,7,9]
  }
}
```

If using the .mapbc file, use the following command to upload your mesh:

```
meshId = flow360client.NewMesh(fname='/path/to/hlcrm.b8.ugrid',
                              noSlipWalls=noSlipWalls,
                              meshName='HLCRM_medium',
                              tags=[])
)
```

Or, if you choose to use the mesh JSON configuration file, use the following command to upload your mesh:

```
meshId = flow360client.NewMesh(fname='/path/to/hlcrm.b8.ugrid',
                              meshJson=meshJson,
                              meshName='HLCRM_medium',
                              tags=[])
)
```

Replace the path in fname with your own file's path. Parameter inputs of meshName and tags are optional. Upon executing this command, it will return the meshId. Use that for the next step.

Run the Case

To run a case, first prepare a JSON input file, either manually or by using the fun3d_to_flow360.py script:

```
python3 /path/to/flow360/flow360client/fun3d_to_flow360.py
/path/to/fun3d.nml/path/to/hlcrm.mapbc /output/path/for/Flow360.json
```

The Flow360.json configuration file for this case ($= 16^\circ$) can be downloaded from [here](#). Be sure to change the flow and solver parameters in JSON configuration file for other cases (e.g. $= 8^\circ$). A full description of the flow configuration parameters can be found [here](#). The case can be submitted using the following command line:

```
caseId = flow360client.NewCase(meshId=meshId,
                                config='/output/path/for/Flow360.json',
                                caseName='HLCRM_case',
                                tags=['alpha 16']
                                )
```

Replace the meshId generated from the above step, and give your own config path. Parameter inputs of caseName and tags are optional. Upon finishing this command, it will return the caseId. Use that for the next step.

Downloading the Results

To download the solver log for your case, use the following command:

```
flow360client.case.DownloadSolverOut(caseId, fileName='path/to/log')
```

The second argument fileName is optional, its default value is 'solver.out'.

The surface data (surface distributions and slices) and the entire flow field can be downloaded, respectively, using:

```
flow360client.case.DownloadSurfaceResults(caseId, fileName='path/to/surfaceSol')
flow360client.case.DownloadVolumetricResults(caseId, fileName='path/to/volumeSol')
```

The second argument fileName is optional, its default value is "volumes.tar.gz" for volume solution and "surfaces.tar.gz" for surface/slice solution.

The residuals history and total forces can be downloaded, respectively, using the following commands:

```
flow360client.case.GetCaseResidual(caseId)
flow360client.case.GetCaseTotalForces(caseId)
```

Finally, you can download the surface forces by their component names using the following commands:

```
flow360client.case.GetCaseSurfaceForcesByNames(caseId, <list of boundaryNames>)
```

Replace the caseId with your own case's ID and <list of boundaryNames> with names of the boundaries. There is a [script](#) to download all of the above stuff by: `python3 downloadAll.py caseId`.

4.1.5 Grid Convergence

The grid convergence of the chosen mesh in this case study has been done by comparing the coefficients of lift and drag for angle of attacks 8° and 16° obtained through Flow360 with solutions presented by other solvers at the High Lift workshop. This grid convergence is presented in Fig. 4.1.3, and the mesh statistics for a coarse, medium, fine and extra fine grids are presented in Table 4.1.1.

Table 4.1.1: Mesh Statistics for different grid sizes

	# of Nodes	# of Cells
Coarse	8.3 M	18.9 M
Medium	27 M	46.8 M
Fine	70.7 M	116.3 M
Extra Fine	208 M	385.6 M

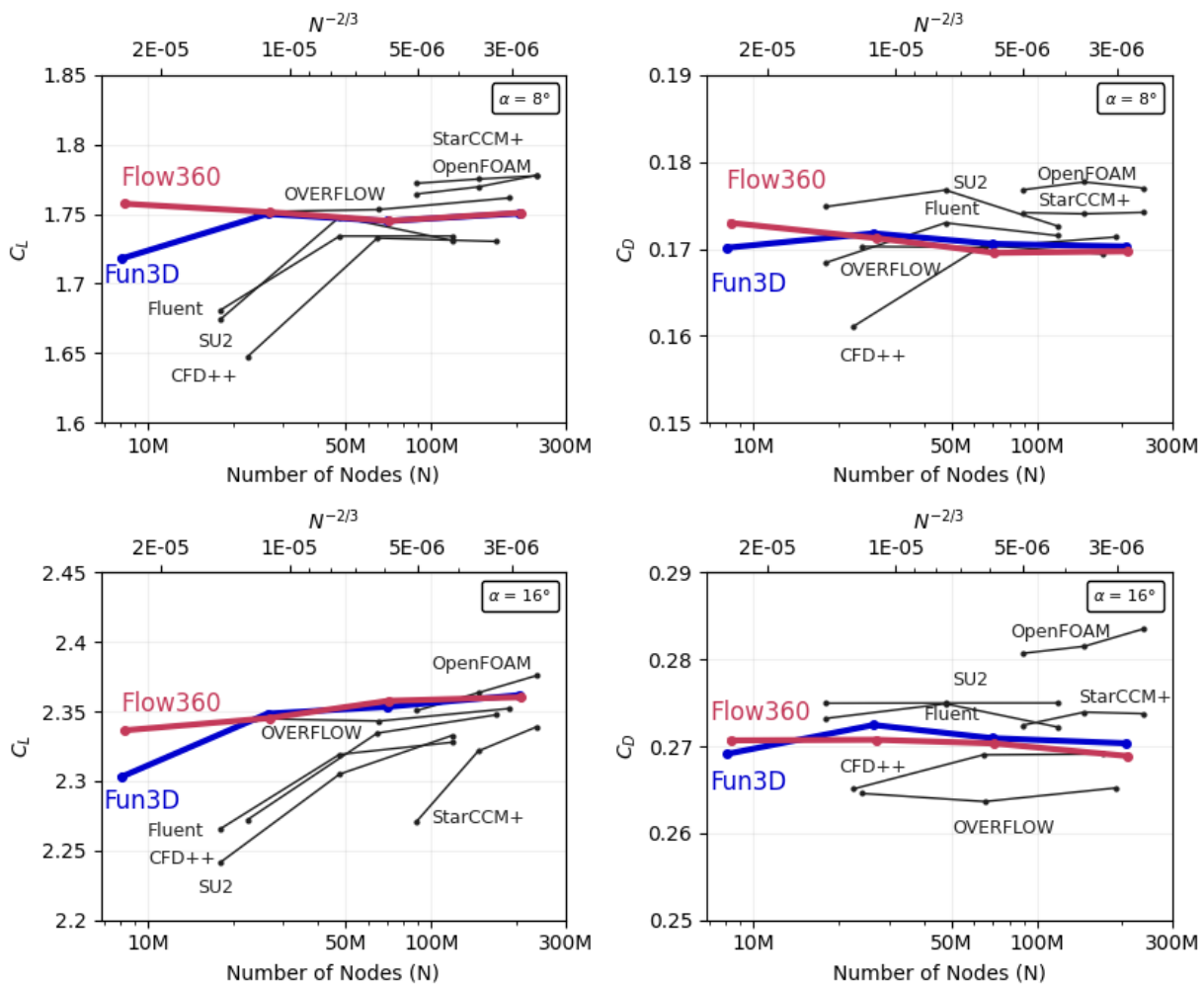


Fig. 4.1.3: Grid convergence of Flow360 compared to the participants of the 3rd AIAA CFD High Lift Prediction Workshop in 2017. The x-axis is the number of mesh grid points N . The scale of the expected numerical error is also labeled for $N^{-2/3}$; the y-axis shows the lift (left plots) and drag (right plots) coefficients computed on various grids, at 8° (upper plots) and 16° (lower plots) degrees angles of attack.

Flow360 performs very well compared to other CFD solvers. Even using coarse mesh, Flow360 obtains results that are

very close to results calculated from fine meshes.

4.1.6 Results

The results data obtained from the previous step can be post-processed in either ParaView or Tecplot. We will use ParaView to post-process these results and plot coefficients of skin friction and pressure.

Fig. 4.1.4 shows the contour plots of coefficient of skin friction for angle of attacks 8° and 16° .

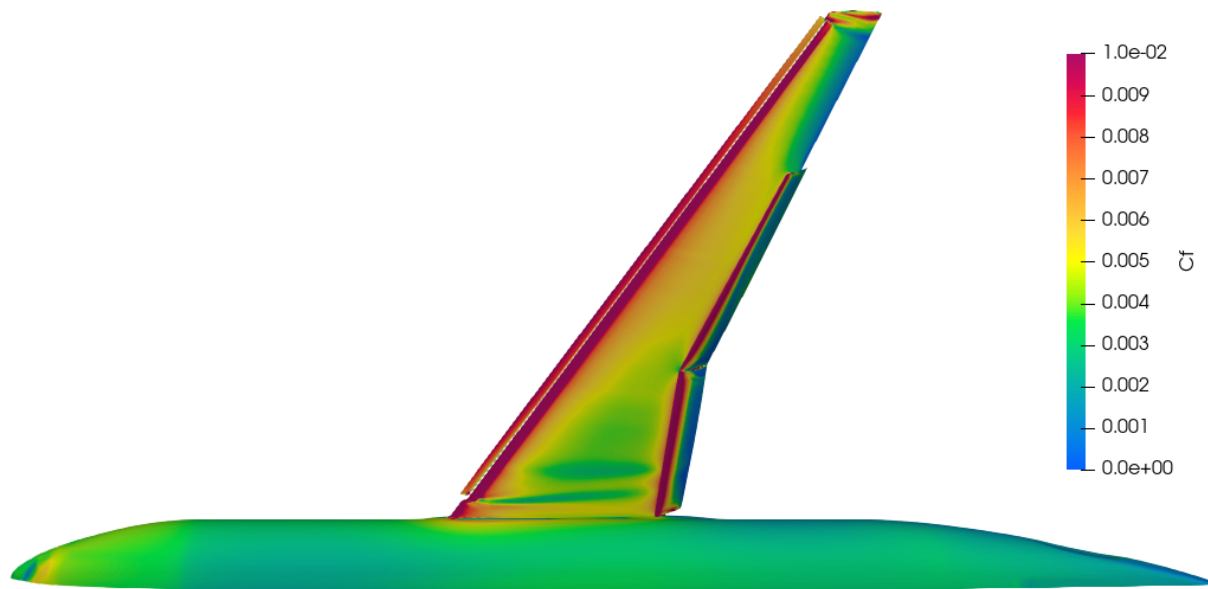


Fig. 4.1.5 shows contour plots of coefficient of pressure angle of attacks 8° and 16° for both the upper and lower surfaces of the wing.

Fig. 4.1.6 shows the Q-criterion plot for Mach number for the coarse grid (8.3M nodes) for the angle of attack 16° .

4.1.7 Summary

This case study demonstrated how to set up, solve and present results of an external aerodynamics flow over NASA's Common Research Model (CRM) from the High Lift Prediction Workshop using Flow360 Python API. The C_L and C_D data obtained through Flow360 has been compared with the solutions from various other solvers from the High Lift Prediction Workshop for accuracy.

4.2 Drag Prediction of Common Research Model

4.2.1 Introduction

The purpose of this case study is to compute the turbulent flow past NASA's Common Research Model (CRM) (a transonic wing-body-tail model) from the 4th AIAA CFD Drag Prediction Workshop. The goal of the workshop is to assess the state-of-the-art computational methods as practical aerodynamic tools for aircraft force and moment prediction of industry relevant geometries. The workshop allows comparisons to be made between different solvers and equation sets, and for performance and accuracy to be assessed.

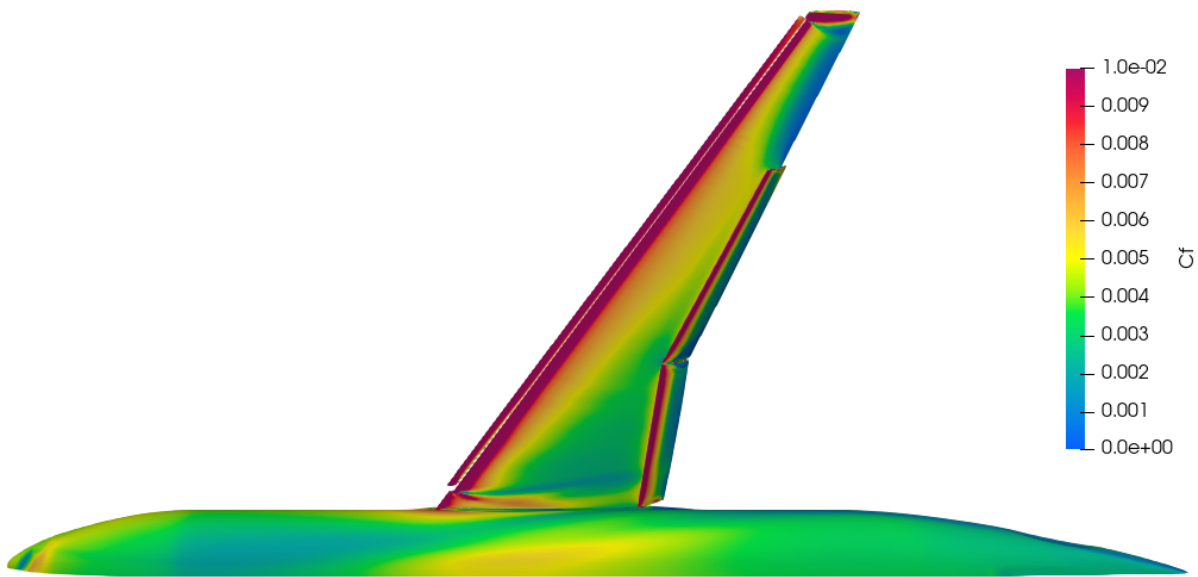
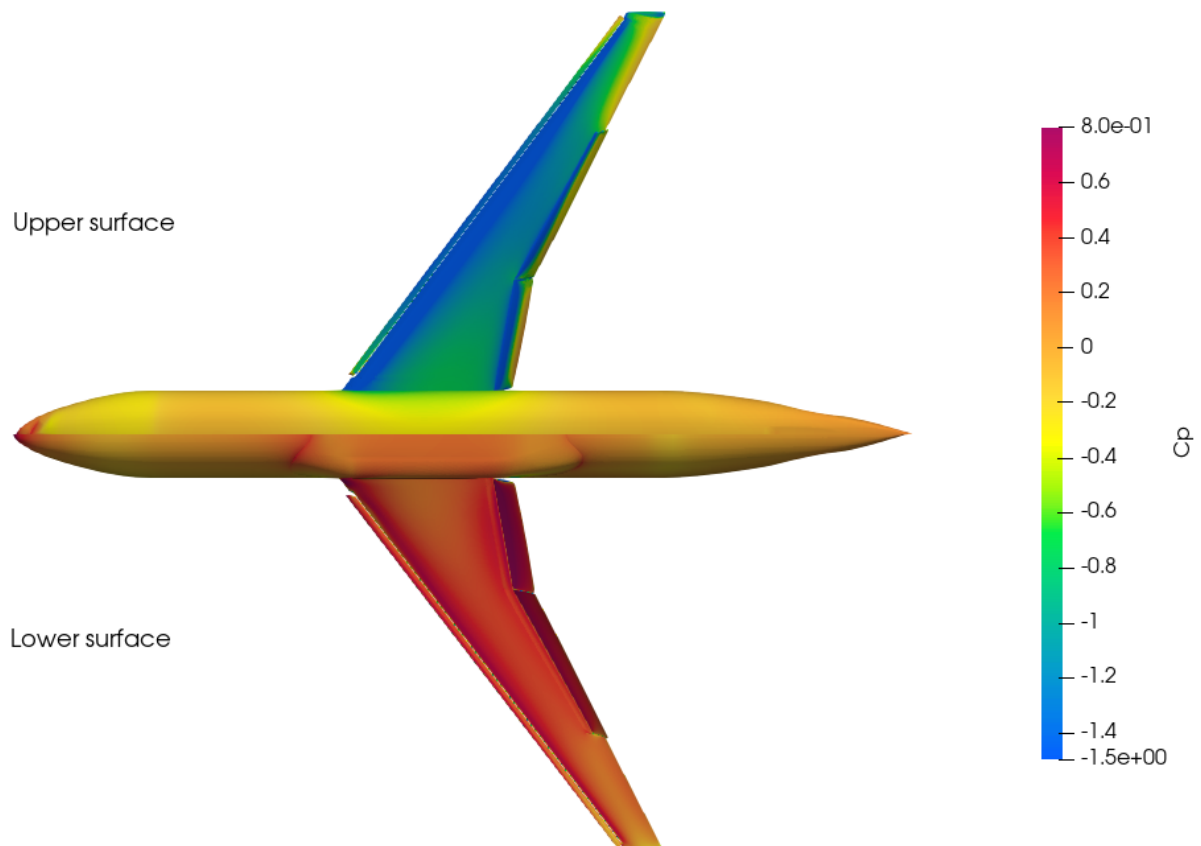


Fig. 4.1.4: Coefficient of skin friction for angle of attacks 8 (top) and 16 (bottom) degrees.



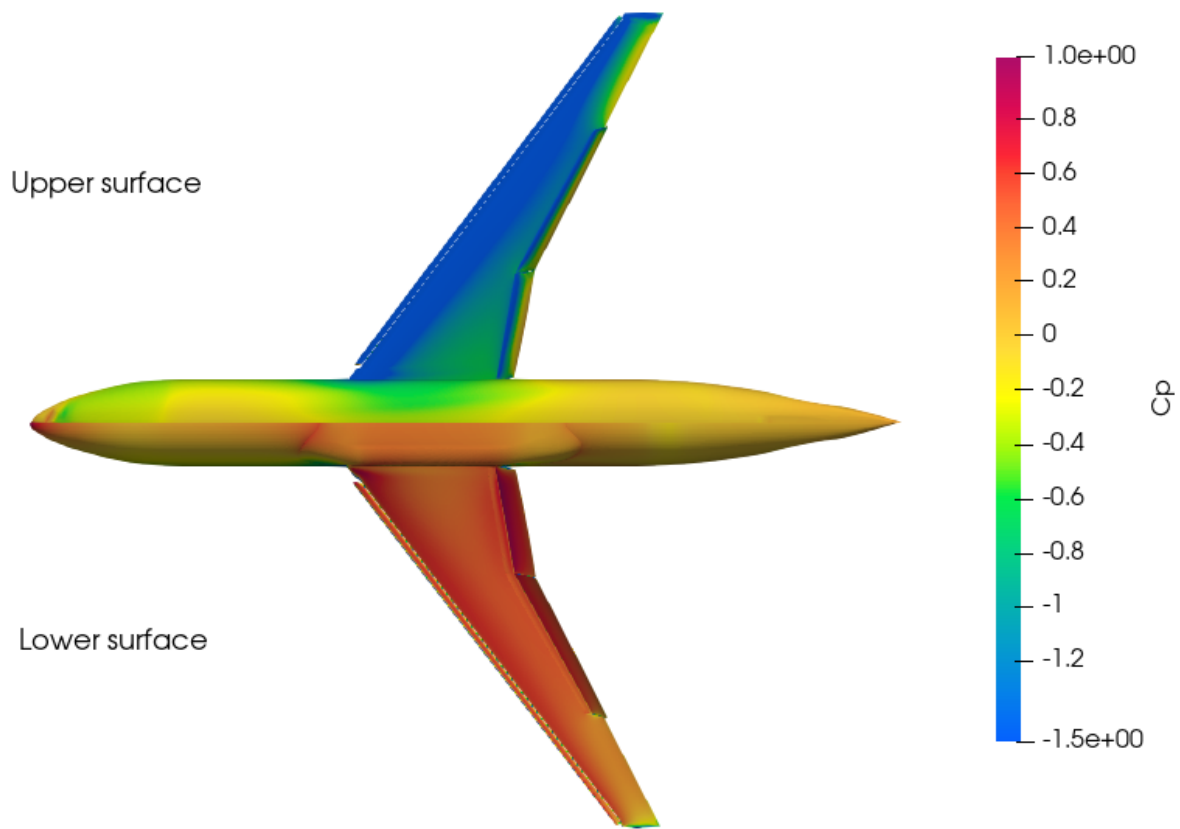
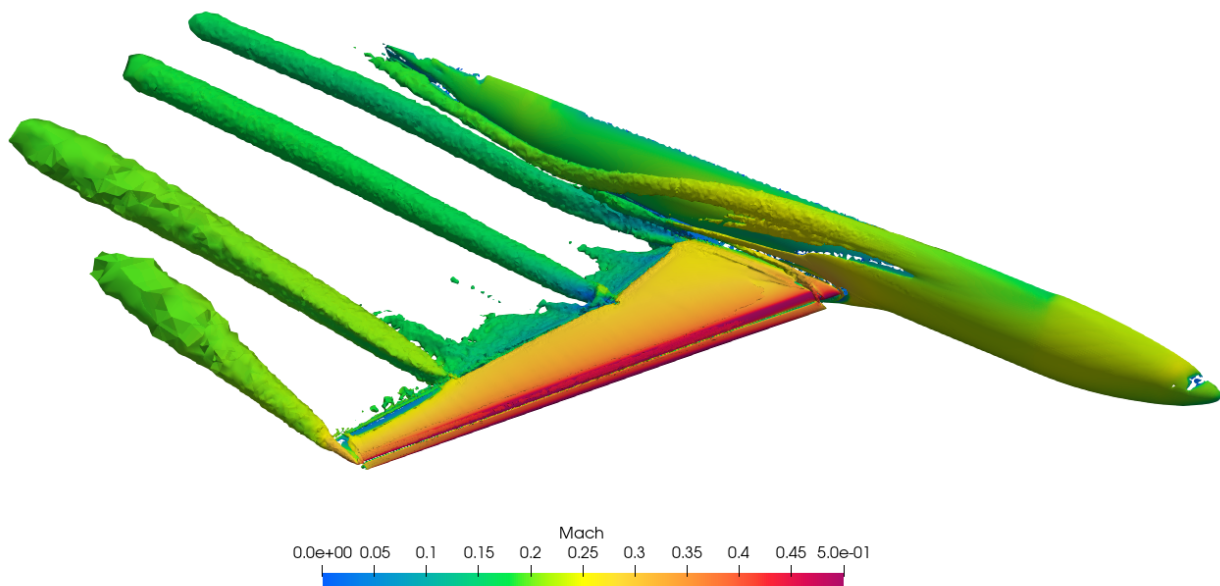


Fig. 4.1.5: Coefficient of pressure for angle of attacks 8 (top) and 16 (bottom) degrees.



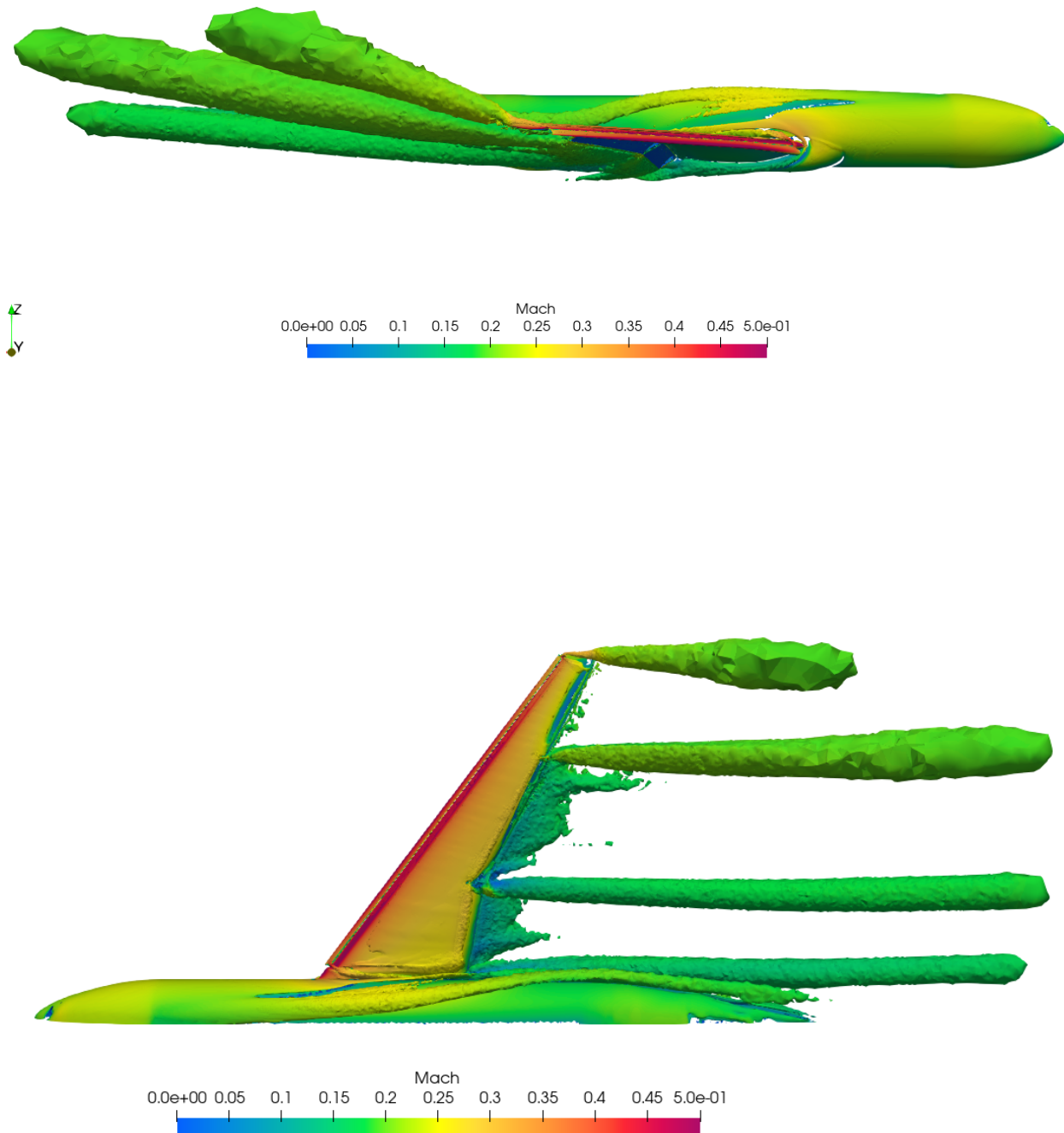


Fig. 4.1.6: Q-criterion plots of the Mach Number for the coarse grid for angle of attack 16 degrees.

In this case study, we will use mesh created in Pointwise and run it through the Flow360 CFD solver developed by Flexcompute Inc. We will demonstrate how to upload a mesh, run a case and view results. Performance of results from Flow360 will be compared with flow solutions from FUN3D, which were presented at the [4th AIAA CFD Drag Prediction Workshop \(DPW-4\)](#).

4.2.2 Problem Description

The problem considers the flow around the CRM at a freestream Mach number of 0.85 ($M_\infty = 0.85$) to achieve a target lift coefficient of 0.5 ($C_L = 0.5$). The geometry of the CRM with a wing-body-tail configuration can be downloaded from [here](#), and is shown in [Fig. 4.2.1](#). The aircraft has a mirror symmetry. Summary of the geometric parameters is given below.

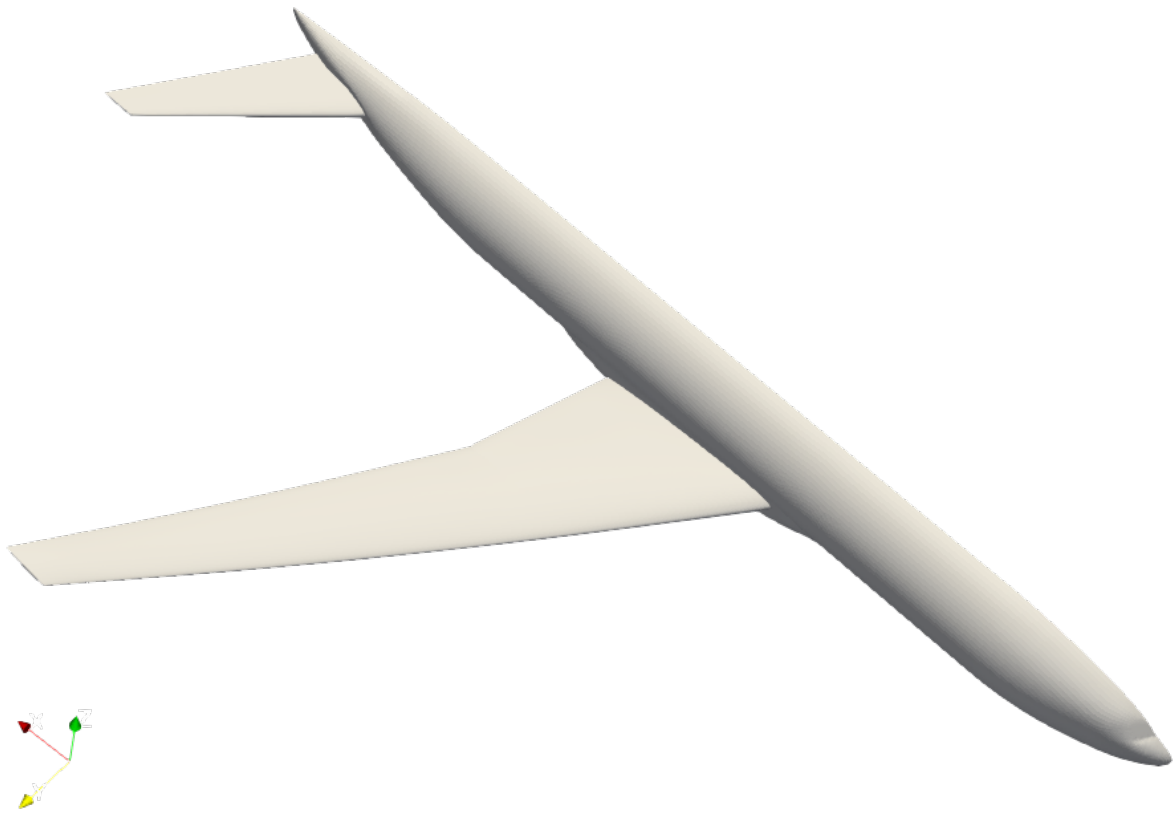


Fig. 4.2.1: Test geometry of the Drag Prediction Workshop

- Mean aerodynamic chord (MAC) = 275.8 in, located at $y=468.75$ in.
- Wing semi-span ($b/2$) = 1156.75 in.
- Reference area of the semi-span model = $S_{ref}/2 = 297,360.0 \text{ in}^2$.
- Moment reference center (MRC): $x = 1325.90$ in, $y = 468.75$ in, $z = 177.95$ in.
- Aspect Ratio (AR) = $b^2/S_{ref} = 9.0$.

And the corresponding flow conditions are:

- Freestream Mach Number, $M_\infty = 0.85$.
- Target Lift Coefficient, $C_L = 0.5$.

- Tail Incidence angle = 0° .
- Reynolds Number (based on MAC) = 5 Million.
- Reference Temperature = 310.93 K.

4.2.3 Mesh

Grids were developed using Pointwise. Flow360 supports mesh file formats in ugrid, cgns, and their .gz or .bz2 compressions. For the purpose of this case study a cgns mesh file format and a medium-resolution mesh, with mixed hexahedral, tetrahedral, prism, and pyramid elements and with about 11.3M nodes was used.

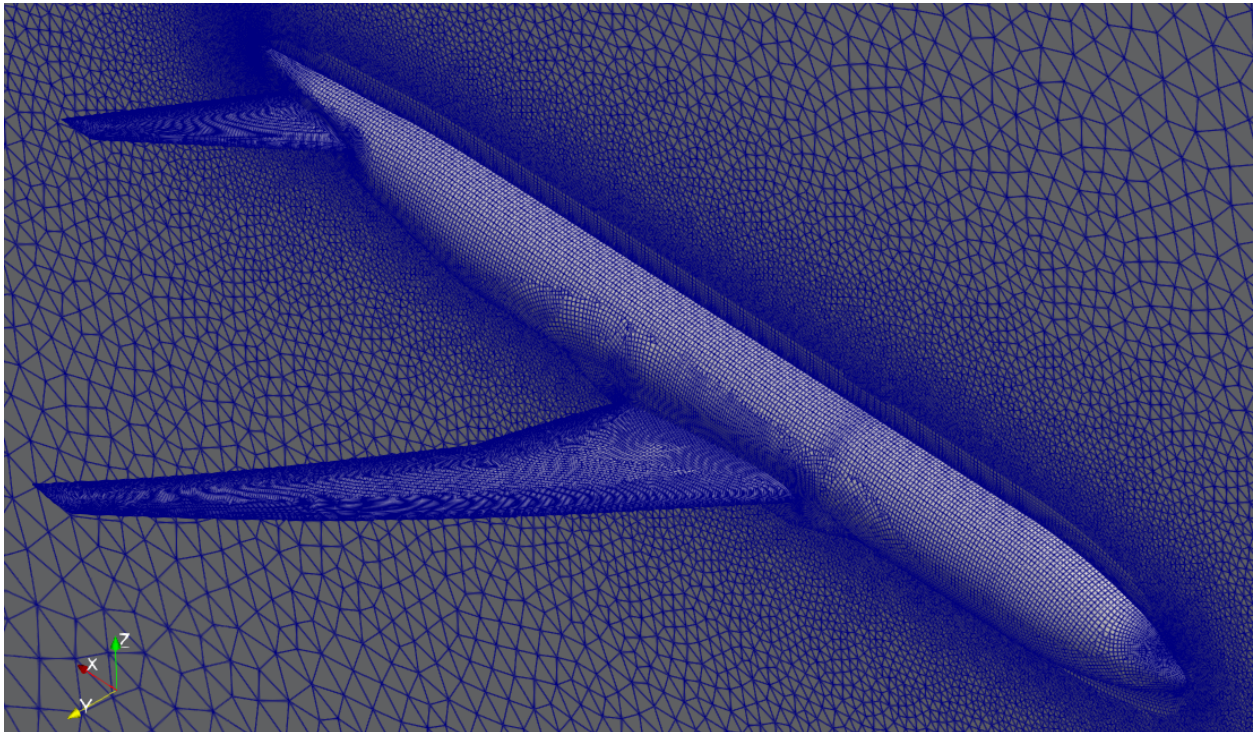


Fig. 4.2.2: Medium-resolution mesh of the CRM

4.2.4 Setup and Solution

Upload the Mesh File

The first thing you need to do is to upload the mesh file and the mesh configuration file to Flow360 cloud. These files can be obtained from here:

- Mesh file: [mesh](#)
- Mesh configuration file: [Flow360Mesh.json](#)

We will do this step-by-step using the Python API. Open your Python interpreter and import the Flow360 client, json module and read the configuration of mesh.

```
python3
import flow360client
import json

meshJson = json.load(open('/path/to/Flow360Mesh.json'))
```

Now upload the mesh using the following command:

```
meshId = flow360client.NewMesh(fname='/path/to/dpw4.cgns',
                               meshJson=meshJson,
                               meshName='DPW4_medium',
                               tags=[])
)
```

Replace the file paths in meshJson and NewMesh with your own file paths. Mesh name and tags are optional. Executing this command will return a meshId, which can be used in the next step to run the case.

Run the Case

To run a case, first download the the configuration file, named Flow360.json, of the CFD solver from [here](#). This file has been configured to run for this medium-resolution mesh. If you decide to run a case with another mesh or flow condition, the configuration parameters need to be adjusted accordingly. A detailed description of the configuration parameters can be found in our [API Reference](#) section. The case can be submitted using the following command:

```
caseId = flow360client.NewCase(meshId=meshId,
                               config='/output/path/for/Flow360.json',
                               caseName='DPW4_case',
                               tags=[])
)
```

Replace the file path in config with your own. You could provide a caseName, which is “Flow360” by default. The tags are optional.

Downloading the Results

To download the solver log for your case, use the following command:

```
flow360client.case.DownloadSolverOut(caseId, fileName='path/to/log')
```

The second argument fileName is optional, its default value is ‘solver.out’.

The surface data (surface distributions and slices) and the entire flow field can be downloaded, respectively, using:

```
flow360client.case.DownloadSurfaceResults(caseId, fileName='path/to/surfaceSol')
flow360client.case.DownloadVolumetricResults(caseId, fileName='path/to/volumeSol')
```

The second argument fileName is optional, its default value is “volumes.tar.gz” for volume solution and “surfaces.tar.gz” for surface/slice solution.

The residuals history and total forces can be downloaded, respectively, using the following commands:

```
flow360client.case.GetCaseResidual(caseId)
flow360client.case.GetCaseTotalForces(caseId)
```


Finally, you can download the surface forces by their component names using the following commands:

```
flow360client.case.GetCaseSurfaceForcesByNames(caseId, <list of boundaryNames>)
```

Replace the caseId with your own case's ID and <list of boundaryNames> with names of the boundaries. There is a [script](#) to download all of the above stuff by: `python3 downloadAll.py caseId`.

4.2.5 Grid Convergence

The grid convergence of the chosen mesh in this case study has been done by comparing the angles of attack, coefficients of total drag, pressure drag, skin friction drag and pitching moment obtained from Flow360 with solutions presented by FUN3D at the Drag Prediction Workshop. This grid convergence is presented in [Fig. 4.2.3](#), and the number of nodes and cells for a coarse, medium and fine grids are presented in [Table 4.2.1](#).

Table 4.2.1: Mesh Statistics for different grid sizes

	# of Nodes	# of Cells
Coarse	5.5 M	10.2 M
Medium	11.3 M	18.7 M
Fine	43 M	84.4 M

4.2.6 Results

The results data obtained from the previous step can be post-processed in either ParaView or Tecplot. You may specify your preference in the Flow360.json configuration file under the outputFormat argument. The coefficients of skin friction (C_f) and pressure (C_p) can be viewed on our [Web UI](#) under the visualization tab.

[Fig. 4.2.4](#) shows the contour plots of coefficient of skin friction

[Fig. 4.2.5](#) shows contour plots of coefficient of pressure

[Fig. 4.2.6](#) shows the Q-criterion plot, colored by Mach number. The threshold for Q-criterion for this case is $1.35e-7$, which can be calculated using:

$$Q_{threshold} = \left(\frac{\text{Freestream Mach Number}}{\text{Wing Span}} \right)^2 = \left(\frac{M_\infty}{b} \right)^2 \quad (4.2.1)$$

4.2.7 Summary

This tutorial demonstrated how to set up, solve and present results of an external aerodynamics flow over NASA's Common Research Model (CRM) from the Drag Prediction Workshop using Flow360's Python API. The Angle of Attack, C_D , $C_{D,p}$, $C_{D,vis}$, and C_{My} data for a target C_L of 0.5 obtained through Flow360 has been compared with the FUN3D solutions presented at the 4th Drag Prediction Workshop for accuracy.

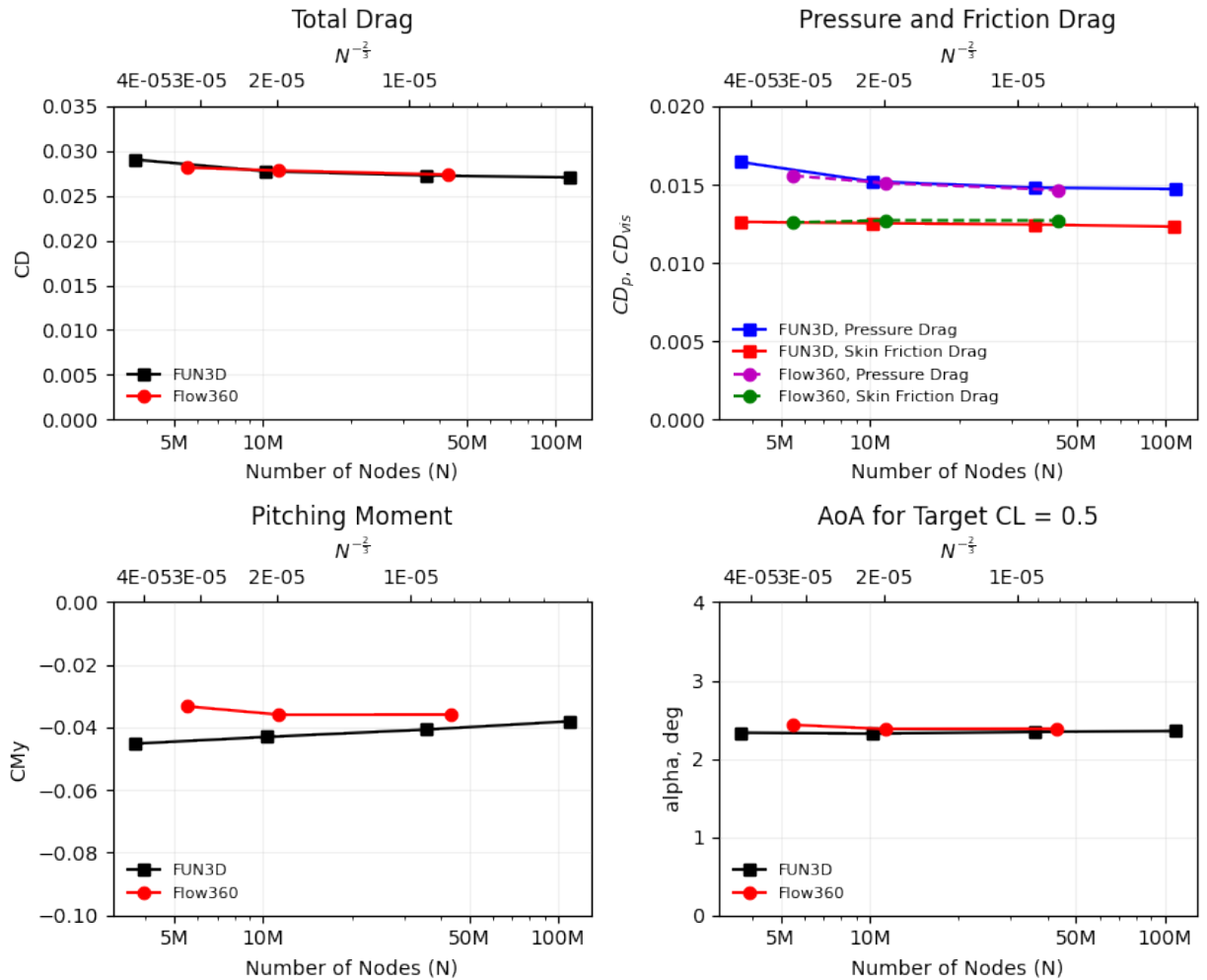


Fig. 4.2.3: Grid convergence of Flow360 compared with FUN3D results from the 4th AIAA CFD Drag Prediction Workshop in 2009. The x-axis is the number of mesh grid points N . The scale of the expected numerical error is also labeled for $N^{-2/3}$; the y-axis shows the total drag (top-left plot), pressure and skin friction drag (top-right plot), pitching moment (bottom-left plot) coefficients and angles of attack (bottom-right plot).

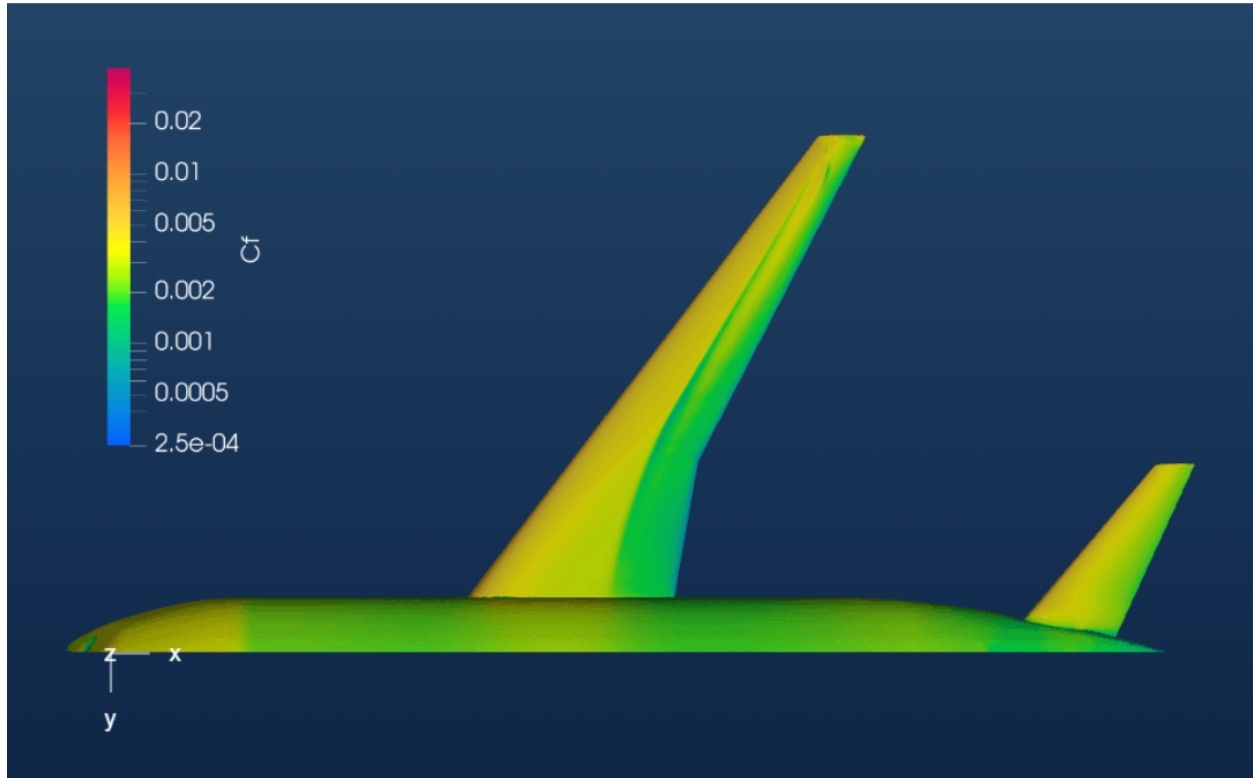


Fig. 4.2.4: Coefficient of Skin Friction

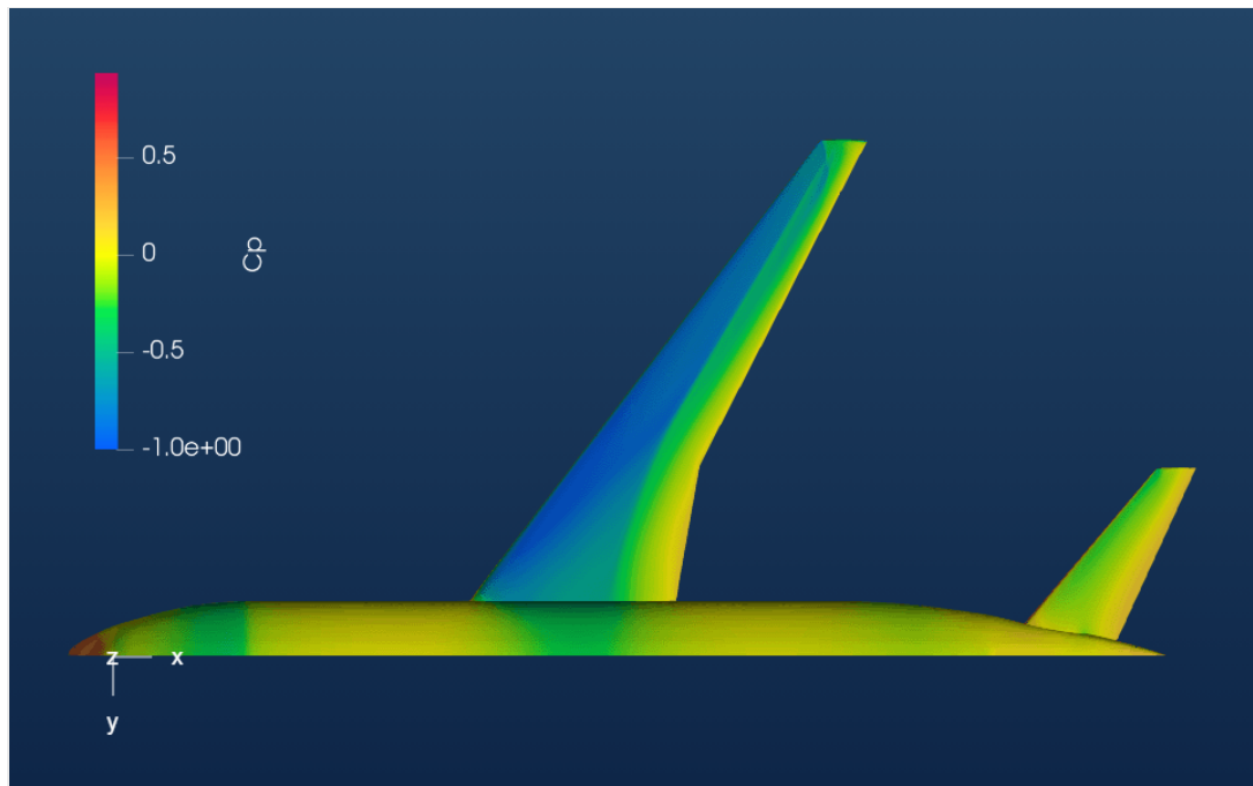


Fig. 4.2.5: Coefficient of Pressure

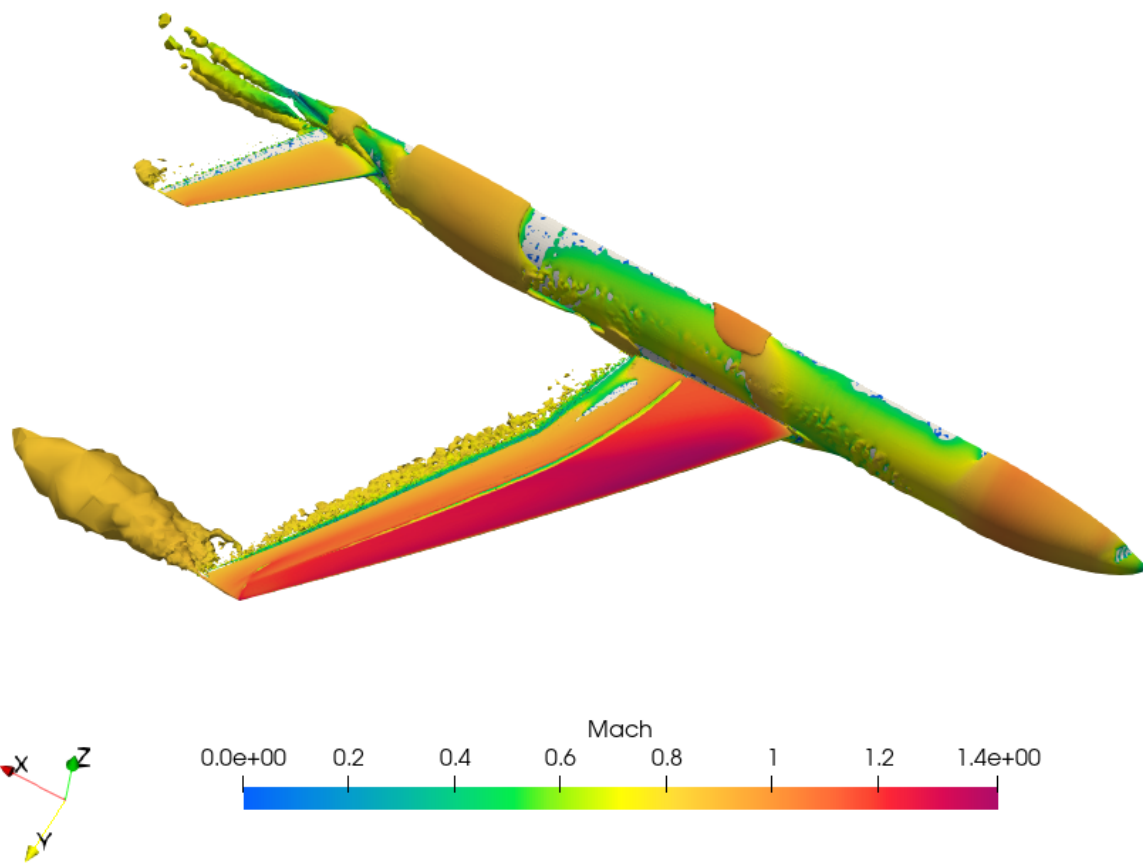


Fig. 4.2.6: Q-criterion plot, colored by Mach Number

4.3 ONERA M6 Wing

4.3.1 Introduction

The Onera M6 wing is a classic CFD validation case for external flows because of its simple geometry combined with complexities of transonic flow (i.e. local supersonic flow, shocks, and turbulent boundary layer separation). It is a swept, semi-span wing with no twist and uses a symmetric airfoil using the ONERA D section. More information about the geometry can be found at [NASA's website](#). The parameters of the geometry are:

- Mean Aerodynamic Chord (MAC) = 0.80167.
- Semi-span = 1.47602.
- Reference area = 1.15315.



Fig. 4.3.1: ONERA M6 Wing Geometry

The mesh used for this case contains 8M nodes and 47M tetrahedrons, and the flow conditions are:

- Mach Number = 0.84.
- Reynolds Number (based on MAC) = 11.72 Million.
- Alpha = 3.06°.
- Reference Temperature = 297.78 K.

4.3.2 Setup and Solution

The 8M-node mesh file, and its corresponding mesh and case configuration files can be downloaded via the following links:

- Mesh file: [wing_tetra.8M.lb8.ugrid](#)
- Mesh configuration file: [Flow360Mesh.json](#)
- Case configuration file: [Flow360.json](#)

For detailed instructions to upload a mesh, run a case and download the results for an ONERA M6 Wing, please refer to the [Quick Start](#) section of this documentation - these details will not be covered in this case study.

4.3.3 Grid Convergence

The grid convergence of the 8M-node mesh in this case study has been done by comparing the coefficients of total lift, total drag, pressure drag, skin friction drag and pitching moment obtained from Flow360 with [solutions provided by various other solvers](#). This grid convergence is presented in [Fig. 4.3.2](#), and the mesh statistics for the coarse, medium, fine and extra fine grids are presented in [Table 4.3.1](#).

Table 4.3.1: Mesh Statistics for different grid sizes

	# of Nodes	# of Cells (Tetrahedrons)
Coarse	113K	663K
Medium	1M	6M
Fine	8M	47M
Extra Fine	90M	537M

The discrepancy between the left and right plots in [Fig. 4.3.2](#) is due to the presence of shock on the upper surface of this transonic wing. To capture these shocks the pressure-density limiter option, `limiterPressureDensity`, needs to be set true (right plots) in the `navierStokerSolver` of the `Flow360.json` configuration file.

4.3.4 Results

The presence of these shock waves are particularly noticeable in [Fig. 4.3.3](#) and [Fig. 4.3.4](#), where the Viscous stress coefficient along x-direction and Coefficient of pressure are plotted over various y-normal slices of the wing and compared with [FUN3D](#) results.

[Fig. 4.3.5](#) shows the contour plot of coefficient of skin friction (C_f).

And [Fig. 4.3.6](#) shows the contour plot of coefficient of pressure (C_p).

4.3.5 Summary

This case study presented the results of a transonic flow over an ONERA M6 Wing, and demonstrated the importance of using a pressure-density limiter to capture the effects of the shock waves on the upper surface of this wing. The CL , CD , CD_p , CD_v , and CM_y data obtained through Flow360 has been compared with various other solvers. Also, the C_p and C_{Fx} y-normal slices data from Flow360 is compared with [FUN3D](#) to show the capturing of these shock waves by turning on the `limitPressureDensity` option in the `navierStokerSolver` of the `Flow360.json` configuration file.

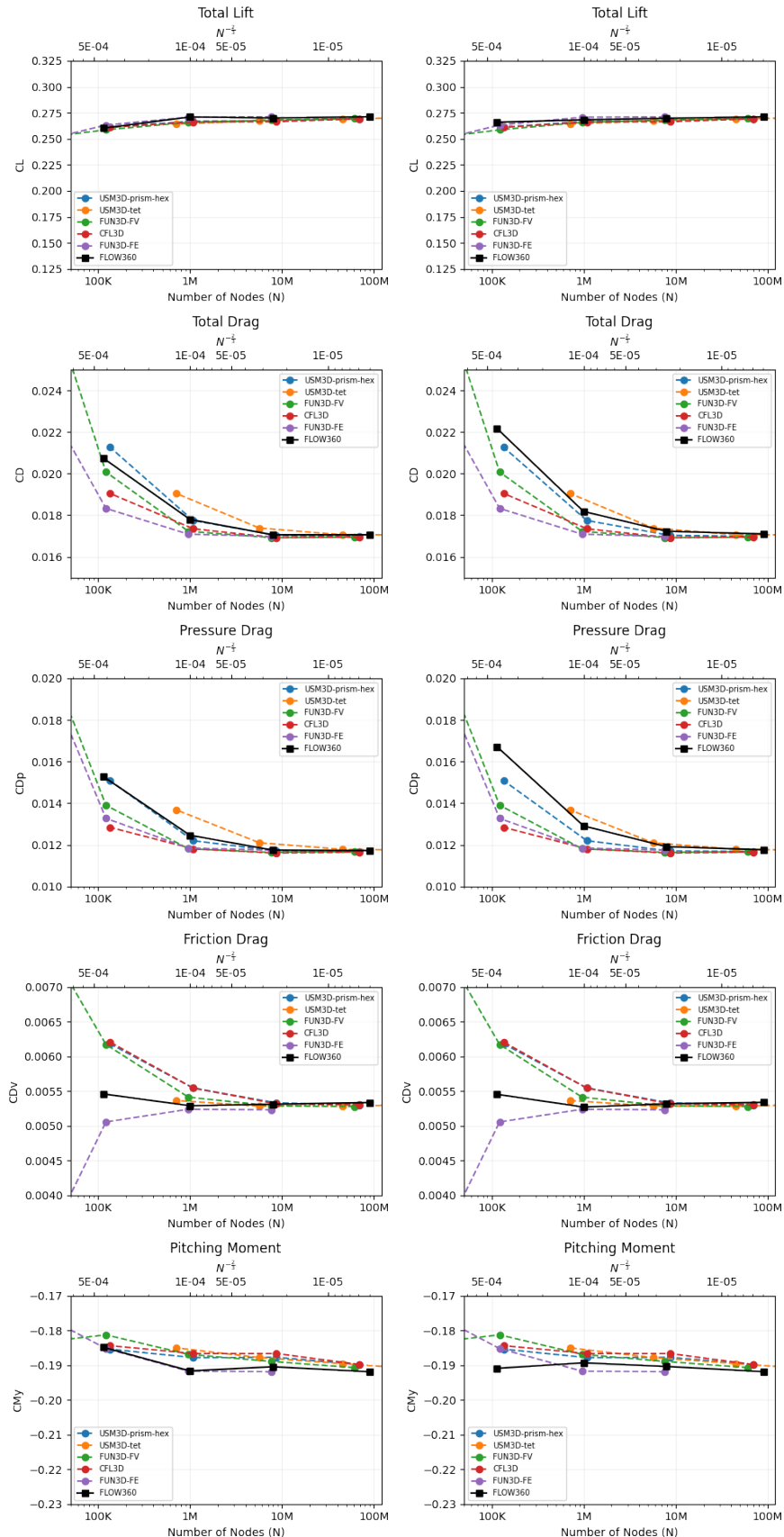


Fig. 4.3.2: Grid convergence of Flow360 results compared with results from various other solvers. The x-axis is the number of mesh grid points N . The scale of the expected numerical error is also labeled for $N^{-2/3}$; the y-axis shows the total lift, total drag, pressure and skin friction drag, and pitching moment coefficients with pressure-density limiter turned off (left) and on (right).

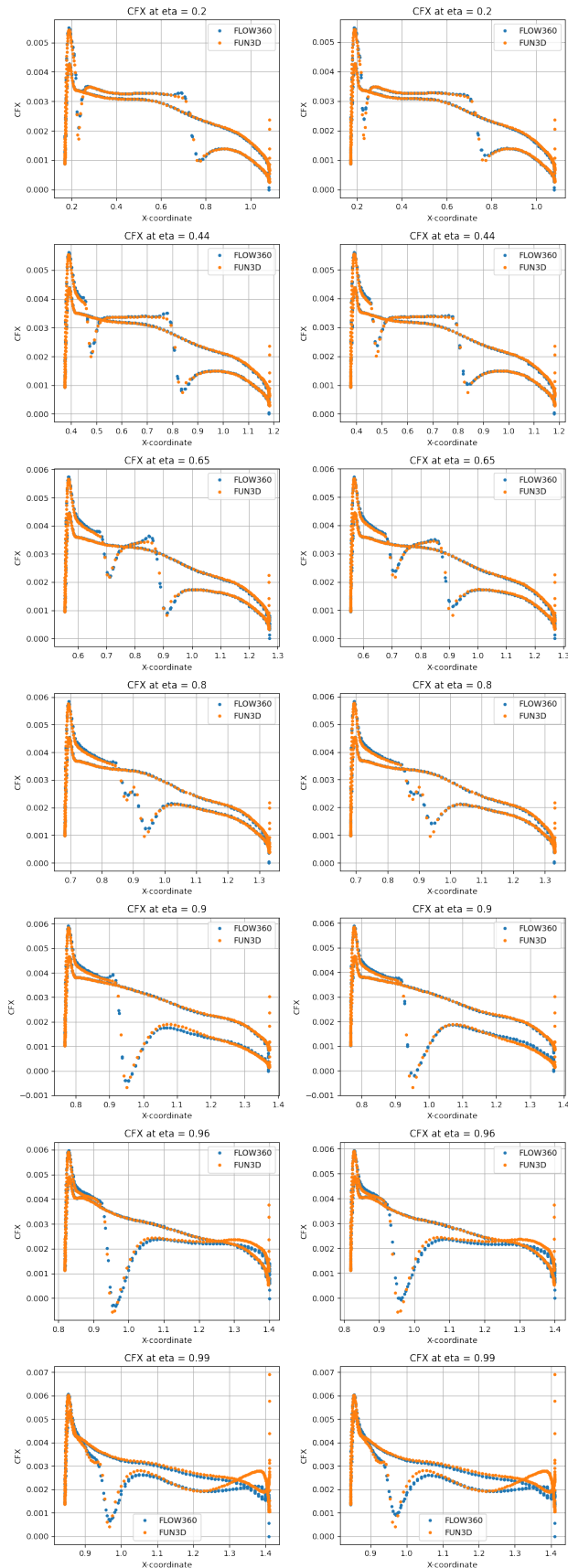


Fig. 4.3.3: Viscous stress coefficient along x-direction plotted over various y-normal slices with pressure-density limiter turned off (left) and on (right).

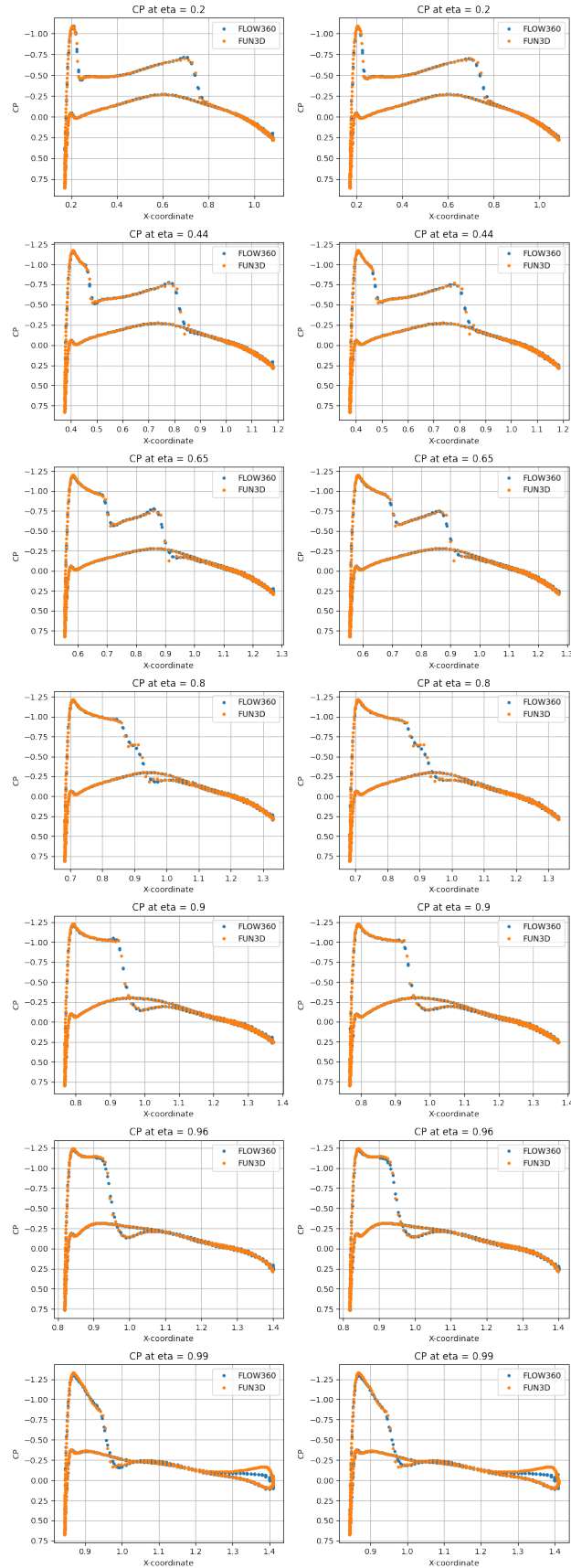


Fig. 4.3.4: Coefficient of pressure along x-direction plotted over various y-normal slices with pressure-density limiter turned on (left) and off (right).

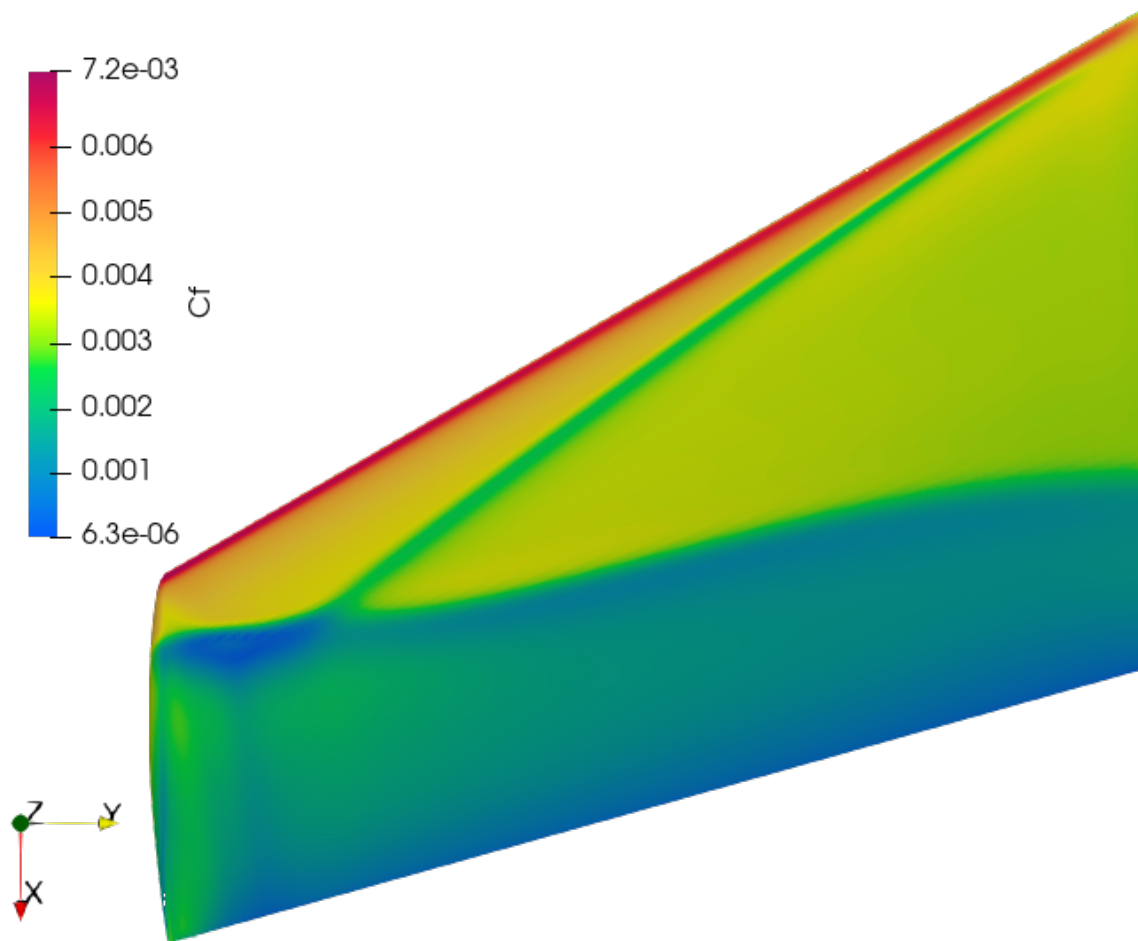


Fig. 4.3.5: Coefficient of Skin Friction

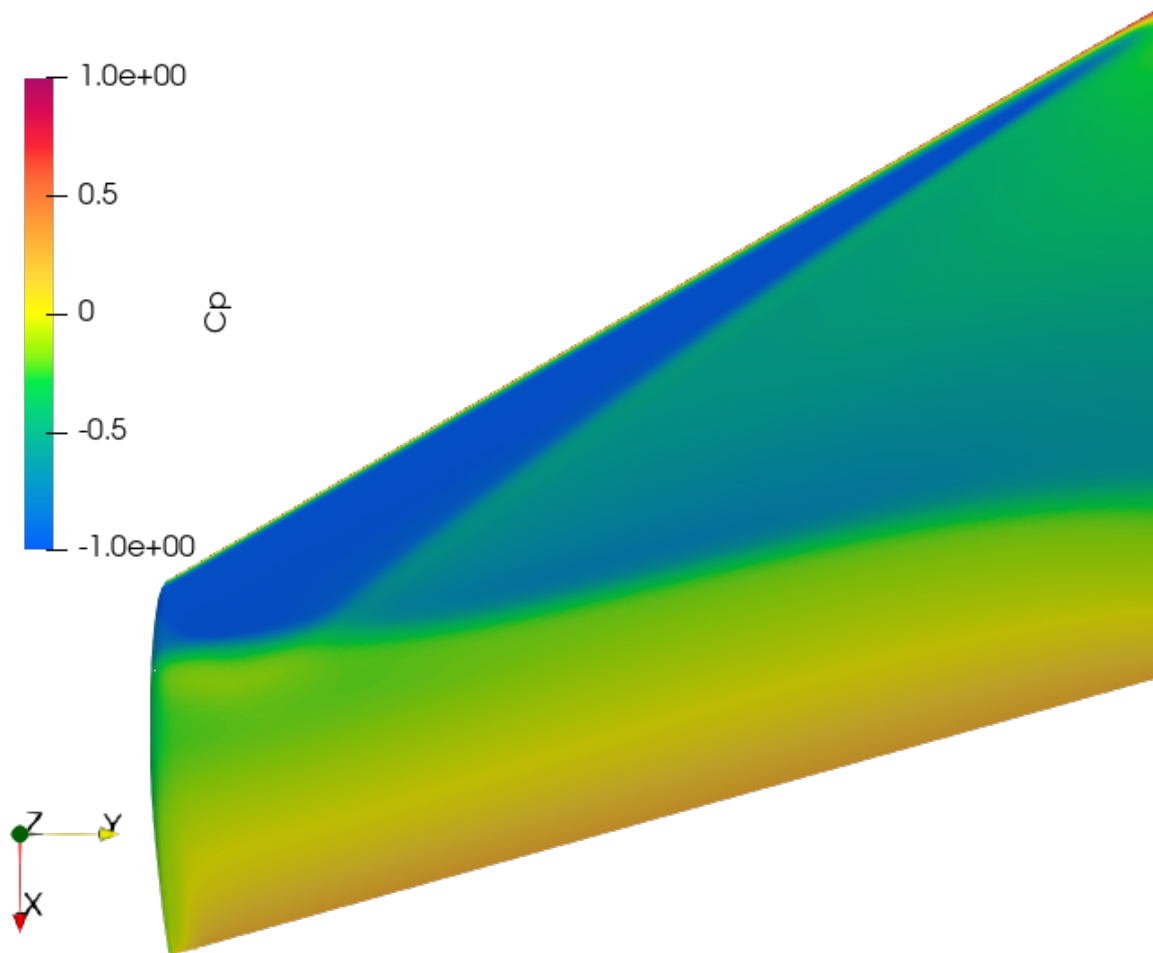


Fig. 4.3.6: Coefficient of Pressure

API REFERENCE

5.1 Installing Flow360 Client

The Flow360 client can be installed (and updated) from PyPI. Make sure you have the Python setuptools. If not, sudo apt-get install python3-setuptools.

```
pip3 install flow360client
pip3 install --upgrade flow360client
```

5.2 Sign in with you Account and Password

An account can be created at <https://client.flexcompute.com/app/signup>.

```
python3
>>> import flow360client
enter your email registered at flexcompute:*****@gmail.com
Password: *****
Do you want to keep logged in on this machine ([Y]es / [N]o)Y
```

Once you have installed the Flow360 client and signed into it, you can run your first case using the ONERA M6 Wing tutorial in the [Quick Start](#) section of this document.

5.3 Configuration Parameters

The current Mesh processor and Solver input configuration parameters for Flow360 are:

5.3.1 Flow360Mesh.json

Type	Options	Default	Description
boundaries	noSlip-Walls	[]	list of names of boundary patches, e.g. [2,3,7] (for .ugrid), ["blk-1/wall1","blk-2/wall2"] (for .cgns)
slidingInterfaces		[]	list of pairs of sliding interfaces
	stationary-Patches	[]	list of names of stationary boundary patches, e.g. ["stationaryField/interface"]
	rotating-Patches	[]	list of names of dynamic boundary patches, e.g. ["rotatingField/interface"]
	axisOfRotation	[]	axis of rotation, e.g. [0,0,-1]
	centerOfRotation	[]	center of rotation, e.g. [0,0,0]

5.3.2 Flow360.json

Some commonly used symbols in Flow360.json:

$L_{gridUnit}$ (**SI unit = m**) physical length represented by unit length in the given mesh file, e.g. if your grid is in feet, $L_{gridUnit} = 1 \text{ foot} = 0.3048 \text{ meter}$; if your grid is in millimeter, $L_{gridUnit} = 1 \text{ millimeter} = 0.001 \text{ meter}$.

C_{∞} (**SI unit = m/s**) speed of sound of freestream

ρ_{∞} (**SI unit = kg/m³**) density of freestream

μ_{∞} (**SI unit = N · s/m²**) dynamic viscosity of freestream

p_{∞} (**SI unit = N/m²**) static pressure of freestream

U_{ref} (**SI unit = m/s**) reference velocity

geometry

Options	Default	Description
refArea	1	The reference area of the geometry
momentCenter	[0.0, 0.0, 0.0]	The x, y, z moment center of the geometry in grid units
momentLength	[1.0, 1.0, 1.0]	The x, y, z moment reference lengths

runControl

Options	Default	Description
restart	FALSE	the solutions are initialized from restarting files or not (no need to set by users)
startAlphaControlPseudoStep	-1	pseudo step at which to start targetCL control. -1 is no trim control. (steady only)
targetCL	-1	The desired trim CL to achieve (associated with startAlphaControlPseudoStep)

freestream

Options	Default	Description
Reynolds	Not required if muRef exists	Non-dimensional Reynolds number based on grid unit, $= \frac{\rho_{\infty} U_{\infty} L_{gridUnit}}{\mu_{\infty}}$. For example, for a mesh with physical length 1.5m represented by 1500 grid units (i.e. mesh is in mm), the $L_{gridUnit}$ in the numerator is 0.001m.
muRef	Not required if Reynolds exists	The reference dynamic viscosity (non-dimensional) in our solver, $= \frac{\mu_{\infty}}{\rho_{\infty} C_{\infty} L_{gridUnit}}$
Mach	REQUIRED	The Mach number, the ratio of freestream speed to the speed of sound.
MachRef	Required if Mach == 0	The reference Mach number to compute the mu, CL/CD, coefficients, etc..., $= U_{ref}/C_{\infty}$. Its default value is "freestream/Mach"
Temperature	REQUIRED	The reference temperature in Kelvin. -1 means globally constant viscosity
alphaAngle	REQUIRED	The angle of attack in degrees
betaAngle	REQUIRED	The side slip angle in degrees
turbulentViscosity	1.0	The multiplicative factor for the freestream turbulent viscosity for the turbulence model
turbulenceIntensity	0.0	The turbulence intensity in the freestream in percent. If greater than zero, activates the transition model. A value of 0.5 in this field means turbulenceIntensity=0.5 %.

boundaries

Type	Format	Description
SlipWall	<pre> "boundary_name" : { "type" : "SlipWall" } </pre>	Slip wall condition. Also used for symmetry.
NoSlipWall	<pre> "boundary_name" : { "type" : "NoSlipWall", "Velocity": [float or "expression" ↵ ↵(default: 0), float or "expression" ↵ ↵(default: 0), float or "expression" ↵ ↵(default: 0)] } </pre>	Sets no-slip wall condition. Optionally, a tangential velocity can be prescribed on the wall using the keyword “Velocity”.
IsothermalWall	<pre> "boundary_name" : { "type" : "IsothermalWall", "Temperature": float or "expression" ↵ ↵(REQUIRED), "Velocity": [float or "expression" ↵ ↵(default: 0), float or "expression" ↵ ↵(default: 0), float or "expression" ↵ ↵(default: 0)] } </pre>	Isothermal wall boundary condition. “Temperature” is specified in Kelvin. Optionally a tangential velocity can be prescribed on the wall using the keyword “Velocity”.
Freestream	<pre> "boundary_name" : { "type" : "Freestream", "Velocity": [float or "expression" ↵ ↵(default: freestream), float or "expression" ↵ ↵(default: freestream), float or "expression" ↵ ↵(default: freestream)] } </pre>	External freestream condition. Optionally, an expression for each of the velocity components can be specified using the keyword “Velocity”.
SubsonicOutflowPressure	<pre> "boundary_name" : { "type" : </pre>	Subsonic outflow, enforced through static pressure ratio.
50	<pre> ↵ "SubsonicOutflowPressure ↵ ", "staticPressureRatio" : ↵ ↵ float } </pre>	Chapter 5. API Reference

Note: “expression” is an expression with “x”, “y”, “z” as independent variables.

volumeOutput

Options	Default	Description
outputFormat	par- aview	“paraview” or “tecplot”
animationFrequency	-1	Frequency at which volume output is saved. -1 is at end of simulation
startAverageIntegrationStep	0	Sub-iteration or time-step to start averaging forces/moments
computeTimeAverages	FALSE	Whether or not to compute time-averaged quantities
primitiveVars	TRUE	Outputs rho, u, v, w, p
vorticity	FALSE	Vorticity
residualNavierStokes	FALSE	5 components of the N-S residual
residualTurbulence	FALSE	Residual for the turbulence model
residualTransition	FALSE	Residual for the transition model
solutionTurbulence	FALSE	Solution for the turbulence model
solutionTransition	FALSE	Solution for the transition model
T	FALSE	Temperature
s	FALSE	Entropy
Cp	TRUE	Coefficient of pressure. $C_p = (\frac{p-p_\infty}{\frac{1}{2}\rho_\infty U_{ref}^2})$.
mut	TRUE	Turbulent viscosity
nuHat	TRUE	nuHat
kOmega	FALSE	k and omega when using kOmegaSST model
mutRatio	FALSE	μ_t/μ_∞
Mach	TRUE	Mach number
VelocityRelative	FALSE	velocity in rotating frame
qriterion	FALSE	Q criterion
gradW	FALSE	Gradient of W
wallDistance	FALSE	wall distance
wallDistanceDir	FALSE	wall distance direction
betMetrics	FALSE	8 quantities related to BET solvers: velocityX, velocityY and velocityZ in rotating reference frame, alpha angle, Cf in axial direction, Cf in circumferential direction, tip loss factor, local solidity multiplied by integration weight

surfaceOutput

Options	Default	Description
outputFormat	par-aview	“paraview” or “tecplot”
animationFrequency	-1	Frequency at which surface output is saved. -1 is at end of simulation
primitiveVars	FALSE	rho, u, v, w, p
Cp	FALSE	Coefficient of pressure
Cf	FALSE	Skin friction coefficient
heatFlux	FALSE	Heat Flux
CfVec	FALSE	Viscous stress coefficient vector. For example, $C_{fVec}[3] = \frac{\tau_{wall}[3]}{\frac{1}{2}\rho_{\infty}U_{ref}^2}$. The τ_{wall} is the vector of viscous stress on the wall.
yPlus	FALSE	y+
wallDistance	FALSE	Wall Distance
Mach	FALSE	Mach number
nodeForcesPerUnitArea	FALSE	$nodeForcesPerUnitArea = \frac{\tau_{wall}[3] - (p - p_{\infty}) * normal[3]}{\rho_{\infty} C_{\infty}^2}$, where the $normal[3]$ is the unit normal vector pointing from solid to fluid.
residualSA	FALSE	Spalart-Allmaras residual magnitude

sliceOutput

Options	Default	Description
outputFormat	par-aview	“paraview” or “tecplot”
animationFrequency	-1	Frequency at which slice output is saved. -1 is at end of simulation
primitiveVars	TRUE	Outputs rho, u, v, w, p
vorticity	FALSE	Vorticity
T	FALSE	Temperature
s	FALSE	Entropy
Cp	FALSE	Coefficient of pressure
mut	FALSE	Turbulent viscosity
mutRatio	FALSE	mut/mu_{∞}
Mach	TRUE	Mach number
gradW	FALSE	gradient of W
slices	[]	List of slices to save after the solver has finished
sliceName		string
sliceNormal		[x, y, z]
sliceOrigin		[x, y, z]

navierStokesSolver

Options	Default	Description
absoluteTolerance	1.00E-10	Tolerance for the NS residual, below which the solver goes to the next physical step
relativeTolerance	1.00E-02	tolerance to the ratio of residual of current pseudoStep to the initial residual, below which the solver goes to the next physical step
CFLMultiplier	1	factor to the CFL definitions defined in “timeStepping” section
linearIterations	30	Number of linear solver iterations
kappaMUSCL	-1	Kappa for the MUSCL scheme, range from [-1, 1], with 1 being unstable.
updateJacobianFrequency	4	Frequency at which the jacobian is updated.
equationEvalFrequency	1	Frequency at which to update the NS equation in loosely-coupled simulations
maxForceJacUpdatePhysicalSteps	0	when which physical steps, the jacobian matrix is updated every pseudo step
orderOfAccuracy	2	order of accuracy in space
extraDissipation	0	add more dissipation to the NS solver
limitVelocity	FALSE	limiter for velocity
limitPressureDensity	FALSE	limiter for pressure and density
viscousWaveSpeedScale	0	Scales the wave speed according to a viscous flux. 0.0 is no speed correction, with larger values providing a larger viscous wave speed correction.

turbulenceModelSolver

Options	Default	Description
modelType	SpalartAllmaras	Turbulence model type can be: “SpalartAllmaras” or “kOmegaSST”
absoluteTolerance	1.00E-08	Tolerance for the turbulence model residual, below which the solver goes to the next physical step
relativeTolerance	1.00E-02	Tolerance to the ratio of residual of current pseudoStep to the initial residual, below which the solver goes to the next physical step
linearIterations	20	Number of linear iterations for the turbulence model linear system
updateJacobianFrequency	4	Frequency at which to update the Jacobian
equationEvalFrequency	4	Frequency at which to evaluate the turbulence equation in loosely-coupled simulations
kappaMUSCL	-1	Kappa for the muscle scheme, range from [-1, 1] with 1 being unstable.
rotationCorrection	FALSE	Rotation correction for the turbulence model. Only support for SpalartAllmaras
orderOfAccuracy	2	Order of accuracy in space
maxForceJacUpdatePhysicalSteps	0	When which physical steps, the jacobian matrix is updated every pseudo step
DDES	FALSE	_true_ enables Delayed Detached Eddy Simulation. Only supported for SpalartAllmaras model

transitionModelSolver

Options	Default	Description
modelType	AmplificationFactorTransport	Transition model type can be: “AmplificationFactorTransport”
absoluteTolerance	1.00E-07	Tolerance for the transition model residual, below which the solver goes to the next physical step
relativeTolerance	1.00E-02	Tolerance to the ratio of residual of current pseudoStep to the initial residual
linearIterations	20	Number of linear iterations for the transition model linear system
updateJacobianFrequency	4	Frequency at which to update the Jacobian
equationEvalFrequency	4	Frequency at which to evaluate the turbulence equation in loosely-coupled simulations
orderOfAccuracy	2	Order of accuracy in space
maxForceJacUpdatePhysicalSteps	0	When which physical steps, the jacobian matrix is updated every pseudo step

initialCondition

Options	Default	Description
type	“freestream”	Use the flow conditions defined in freestream section to set initial condition. Could be “freestream” or an “expression”

timeStepping

Options	Default	Description
maxPhysicalSteps	1	Maximum physical steps
timeStepSize	“inf”	Nondimensional time step size in physical step marching, it is calculated as $\frac{\Delta t_{physical} C_{\infty}}{L_{gridUnit}}$. $\Delta t_{physical}$ is the physical time step size. “inf” means steady solver.
maxPseudoSteps	2000	Maximum pseudo steps within one physical step
CFL->initial	5	Initial CFL for solving pseudo time step
CFL->final	200	Final CFL for solving pseudo time step
CFL->rampSteps	40	Number of steps before reaching the final CFL within 1 physical step

slidingInterfaces (list)

Options	Default	Description
stationaryPatches	Empty	a list of static patch names of an interface
rotatingPatches	Empty	a list of dynamic patch names of an interface
thetaRadians	Empty	expression for rotation angle (in radians) as a function of time
thetaDegrees	Empty	expression for rotation angle (in degrees) as a function of time
omegaRadians	Empty	non-dimensional rotating speed, radians/nondim-unit-time, $= \Omega * L_{gridUnit} / C_{\infty}$, where the SI unit of Ω is rad/s.
omegaDegrees	Empty	non-dimensional rotating speed, degrees/nondim-unit-time, $= \text{omegaRadians} * 180 / \pi$
centerOfRotation	Empty	a 3D array, representing the origin of rotation, e.g. [0,0,0]
axisOfRotation	Empty	a 3D array, representing the rotation axis, e.g. [0,0,1]
volumeName	Empty	a list of dynamic volume names related to the above {omega, centerOfRotation, axisOfRotation}

actuatorDisks (list)

Options	De- fault	Description
center	Empty	center of the actuator disk
axisThrust	Empty	direction of the thrust, it is an unit vector
thickness	Empty	thickness of the actuator disk
forcePerArea->radius (list)	Empty	radius of the sampled locations in grid unit
forcePerArea->thrust (list)	Empty	force per area along the axisThrust, positive means the axial force follows the same direction of “axisThrust”. It is non-dimensional, $= \frac{\text{thrustPerArea}(SI=N/m^2)}{\rho_{\infty} C_{\infty}^2}$
forcePerArea->circumferential (list)	Empty	force per area in circumferential direction, positive means the circumferential force follows the same direction of “axisThrust” based on right hand rule. It is non-dimensional, $= \frac{\text{circumferentialForcePerArea}(SI=N/m^2)}{\rho_{\infty} C_{\infty}^2}$

BETDisks (list)

Option	Default	Description
centerOfRotation	Empty	[3-array] center of the Blade Element Theory (BET) disk
axisOfRotation	Empty	[3-array] rotational axis of the BET disk
numberOfBlades	Empty	[int] number of blades to model
radius	Empty	[float] non-dimensional radius of the rotor disk, = $\text{Radius}_{\text{dimensional}} / L_{\text{gridUnit}}$
omega	Empty	[float] non-dimensional rotating speed, radians/nondim-unit-time, = $\Omega * L_{\text{gridUnit}} / C_{\infty}$, where the SI unit of Ω is rad/s.
chordRef	Empty	[float] non-dimensional reference chord used to compute sectional blade loadings.
nLoadingNodes	Empty	[float] Number of nodes used to compute the sectional thrust and torque coefficient C_t and C_q , defined in <i>BET Loading Output</i> . Recommended value is 20.
thickness	Empty	[float] non-dimensional thickness of the BET disk. Should be less than the thickness of the refined region of the disk mesh.
bladeLineChord	0.0	[float] non-dimensional chord to use if performing an unsteady blade-line (as opposed to steady blade-disk) simulation. Recommended value is 1-2x the physical mean aerodynamic chord (MAC) of the blade for blade line analysis. Default of 0.0 indicates to run blade-disk analysis instead of blade-line.
initialBladeDirection	Empty	[3-array]. Orientation of the first blade in the blade-line model. Must be specified if performing blade-line analysis.
twists	Empty	[list(dict)] A list of dictionary entries specifying the twist in degrees as a function of radial location. Entries in the list must already be sorted by radius. Example entry in the list would be {"radius" : 5.2, "twist" : 32.5}.
chords	Empty	[list(dict)] A list of dictionary entries specifying the blade chord as a function of the radial location. Entries in the list must already be sorted by radius. Example entry in the list would be {"radius" : 5.2, "chord" : 12.0}.
sectionalPolars	Empty	[list(dict)] A list of dictionaries for every radial location specified in <i>sectionalRadiuses</i> . Each dict has two entries, "liftCoeffs" and "dragCoeffs", both of which have the same data storage format: 3D arrays (implemented as nested lists). The first index of the array corresponds to the <i>MachNumbers</i> of the specified polar data. The second index of the array corresponds to the <i>ReynoldsNumbers</i> of the polar data. The third index corresponds to the <i>alphas</i> . The value specifies the lift or drag coefficient, respectively.
sectionalRadiuses	Empty	[list(float)] A list of the radial locations at which C_l and C_d are specified in <i>sectionalPolars</i>
alphas	Empty	[list(float)] alphas associated with airfoil polars provided in <i>sectionalPolars</i> in degrees.
MachNumbers	Empty	[list(float)] Mach numbers associated with airfoil polars provided in <i>sectionalPolars</i> .
ReynoldsNumbers	Empty	[list(float)] Reynolds numbers associated with the airfoil polars provided in <i>sectionalPolars</i> .

WEBINAR

6.1 Mesh and Run a High-Fidelity Aircraft Simulation in Minutes

17 November 2020 at 10AM CST (-6 UTC)

Intelligent automatic meshing meets super solver speed for unrivaled CFD efficiency.

<https://info.pointwise.com/mesh-run-a-high-fidelity-aircraft-simulation-in-minutes>

Register today for a live webinar that will help boost productivity while simultaneously ensuring mesh consistency and reliability across your organization. Increasingly complex geometry, larger simulations, and design optimization are just a few characteristics of advanced simulation and design that emphasize the need for more automation and higher throughput. Yet, automation in the absence of control and expertise often has limited applicability, including the inability to maintain consistency throughout the design and simulation process. This can result in an inaccurate, and sometimes unreliable, understanding of a design or the design space, slowing the turn-around time and delaying time to market.

With this in mind, at Pointwise we have been developing a new suite of features called Flashpoint that take the control and expertise we are known for and combines them with deeper automation. Flashpoint, and in particular, the automatic surface meshing component, will help ensure consistency and reliability throughout the meshing process while also providing a framework for encapsulating your organization best practices. We recently collaborated with FlexCompute, the developers of Flow360, a new cloud-based CFD solver, and using the NASA Common Research model as our benchmark, we are demonstrating the ability to mesh and run a high-fidelity aircraft simulation in minutes.

Discover how to:

- Automatically generate a high-quality surface mesh for an aircraft geometry.
- Construct and populate a hybrid-viscous, hex-core volume mesh in minutes.
- Reduce turnaround time and setup, deploy, and simulate using Flow360.

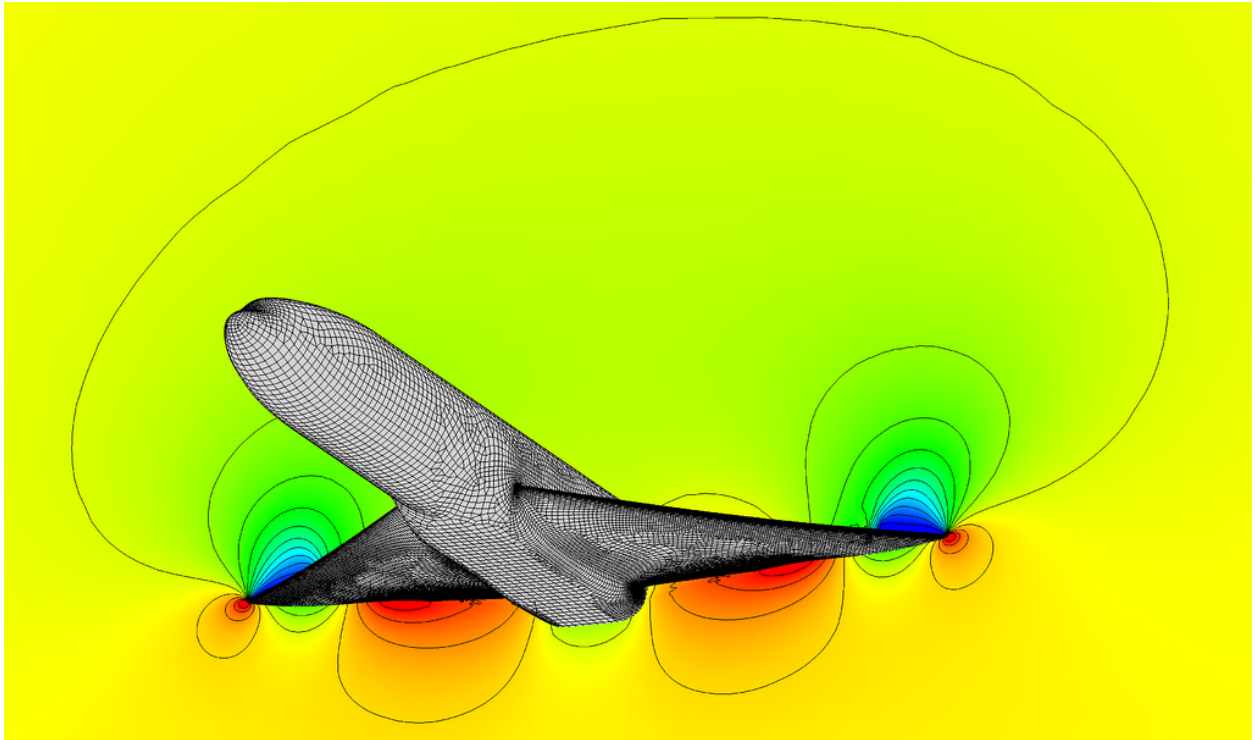


Fig. 6.1.1: A quad-dominant surface mesh and hex-core volume were generated for NASA Common Research Model using Pointwise latest automatic surface meshing and voxel meshing capabilities. The solution was then computed using FlexCompute Flow360 CFD solver with a runtime of just 105 seconds.

FREQUENTLY ASKED QUESTIONS

7.1 Where are my AWS credentials stored locally?

Your AWS credentials are encrypted and stored locally (if you hit Yes previously at authentication step) at

```
~/ .flow360/
```

For security, your password is stored as hashed value, so nobody can guess your password.

7.2 How to check my mesh processing status?

To list all your mesh files:

```
>>> flow360client.mesh.ListMeshes()
```

To view one particular mesh:

```
>>> flow360client.mesh.GetMeshInfo('')
```

7.3 My case is still running, but how can I check the current residual and surface force result?

```
>>> caseResid = flow360client.case.GetCaseResidual('')
```

7.4 How do I download or view a finished case result?

To download the surface data (surface distributions and slices):

```
>>> flow360client.case.DownloadSurfaceResults('', '/tmp/surfaces.tar.gz')
```

Replace the second parameter with your target location and output file name, ending with '.tar.gz'.

To download the entire flowfield:

```
>>> flow360client.case.DownloadVolumetricResults('', '/tmp/volume.tar.gz')
```

7.5 How can I delete my mesh or case?

To delete a mesh:

```
>>> flow360client.mesh.DeleteMesh('')
```

To delete a case:

```
>>> flow360client.case.DeleteCase('')
```

Caution: You won't be able to recover your deleted cases or mesh files including its results after your deletion.

WHITE PAPER

The White Paper for Flow360 can be found [here](#).